Jörg-Stefan Praßni

Interactive Feature Detection in Volumetric Data

Münster • 2012

Informatik

Interactive Feature Detection in Volumetric Data

Inaugural-Dissertation zur Erlangung des Doktorgrades der
Naturwissenschaften im Fachbereich Mathematik und Informatik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

vorgelegt von

Jörg-Stefan Praßni

aus Duisburg

2012

Dekan:	Prof. Dr. Matthias Löwe
Erster Gutacher:	Prof. Dr. Klaus H. Hinrichs
Zweiter Gutacher:	Prof. Dr. Xiaoyi Jiang
Tag der mündlichen Prüfung:	09.07.2012
Tag der Promotion:	09.07.2012

Abstract

With the increasing size and complexity of volumetric data sets, which arise from numerous applications, for example in the medical domain, the demand for efficient and reliable techniques for the extraction of relevant information from such data is growing. This dissertation presents three volumetric feature detection approaches that focus on an efficient interplay between user and system. The first technique exploits the LH transfer function space in order to enable the user to classify boundaries by directly marking them in the volume rendering image, without requiring interaction in the data domain. Second, we propose a shape-based feature detection approach that blurs the border between fast but limited classification and powerful but laborious segmentation techniques. Based on a rough pre-segmentation, the system decomposes the volume into a set of smaller structures and computes shape descriptors for them that serve as basis for an interactively defined shape transfer function. Third, we present a guided probabilistic volume segmentation workflow that focuses on the minimization of uncertainty in the resulting segmentation. In an iterative process, the system continuously assesses uncertainty of an intermediate random walker-based segmentation in order to detect regions with high ambiguity, to which the user's attention is directed to support the correction of potential segmentation errors.

Kurzfassung

Mit der zunehmenden Größe und Komplexität volumetrischer Datensätze, wie sie beispielsweise in der medizinischen Diagnose verwendet werden, steigt der Bedarf an effizienten und zuverlässigen Methoden zur Extraktion relevanter Informationen aus solcher Art von Daten. Im Rahmen dieser Dissertation wurden drei Techniken für die interaktive Merkmalsdetektion in Volumendaten entwickelt, die speziell auf ein effizientes Zusammenspiel zwischen Benutzer und Computersystem abzielen. Das erste Verfahren auf Basis des LH-Transferfunktionsraumes ermöglicht es dem Benutzer, Objekt-Oberflächen in einem Volumendatensatz durch direktes Markieren im gerenderten Bild zu identifizieren, wobei keine Interaktion im Datenraum des Volumens benötigt wird. Zweitens wird ein formbasiertes Klassifikationsverfahren vorgestellt, das ausgehend von einer groben Vorsegmentierung den Volumendatensatz in eine Menge von kleineren Regionen zerlegt, deren Form anschließend mit eigens entwickelten Klassifikatoren bestimmt wird. Dies versetzt den Benutzer in die Lage, lokale Merkmale des Volumens anhand ihrer Form zu unterscheiden und ihnen optische Eigenschaften mittels einer entsprechenden Transferfunktion zuzuweisen. Drittens wird ein interaktives Volumen-Segmentierungsverfahren auf Basis des Random Walker-Algorithmus beschrieben, das speziell auf die Verringerung von Fehlklassifizierungen in der resultierenden Segmentierung abzielt. In einem iterativen Vorgehen bestimmt das Computersystem kontinuierlich unsichere Regionen des aktuellen Zwischenergebnisses, um die Aufmerksamkeit des Benutzers auf diese Regionen zu lenken, so dass dieser potentielle Segmentierungsfehler zuverlässig korrigieren kann.

Contents

Preface

This dissertation represents the result of research that has been carried out from October 2008 till May 2012 at the Department of Computer Science at the University of Münster. First and foremost, I would like to thank my supervisor, Prof. Dr. Klaus Hinrichs, for giving me the opportunity to conduct this exciting research in his group and for his guidance and constant encouragement throughout the years. My special thanks go to Prof. Dr. Timo Ropinski for his valuable advice and our many fruitful discussions, which have shaped my dissertation.

Furthermore, I would like to express my thanks to all former and present members of the Voreen development team, including Dr. Jörg Mensmann, Dr. Jennis Meyer-Spradow, Stefan Diepenbrock, and Florian Lindemann, for all the hard work they have put into the construction of this powerful visualization framework, which has served me as basis for the implementation of the techniques presented in this thesis. Moreover, I want to thank all my further colleagues at the Visualization and Computer Graphics Research Group for the nice working atmosphere and the numerous inspiring on-topic as well as off-topic discussions.

This work was partly supported by grants from the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) through the Collaborative Research Centre 656 Molecular Cardiovascular Imaging (project Z1). I thank my colleagues there for the stimulating collaboration and the exciting insights into the medical domain.

Finally, I would like to express my deepest gratitude to my parents and grandparents, whose unconditional support and encouragement have been a constant throughout my life.

Münster, May 2012 *Jörg-Stefan Praßni*

Chapter 1

Introduction

With the increasingly widespread use of volumetric scanning techniques in medicine, biology, seismology, among many other fields, and the increasing size and complexity of the resulting data sets, the demand for sophisticated data analysis techniques for volume data is growing. Extensive progress has been made in the development of volume rendering algorithms, which are nowadays capable of generating 3D visualizations of multi-gigabyte data sets at interactive frame rates even on consumer hardware. The detection and analysis of features of interest embedded in volumetric data, however, remains a challenging task. Although fully-automated analysis techniques exist for specific purposes, in many application cases user interaction is crucial for gaining insight and understanding of the data. In interactive volume analysis, usually a trade-off between fast but imprecise *classification* schemes and accurate but time-consuming *segmentation* techniques has to be made.

Volume classification denotes the assignment of each voxel in the data set to a material class, which is defined in the data domain. A voxel's class membership is solely determined by its abstract data value, whereas spatial information is neglected. In volume visualization, classification is usually performed by the specification of a *transfer function* (TF) that maps abstract data values to optical properties, such as color and opacity, which are then used during the rendering process. The actual transfer function domain varies among volume visualization setups. Traditional transfer functions consider the data value only and are thus called one-dimensional, whereas multidimensional transfer functions also incorporate derived measures of the data value, such as the gradient magnitude or higher order derivatives. Multidimensional transfer functions allow for a more flexible classification compared to one-dimensional ones, in that they are able to distinguish boundaries within the data set from more homogeneous regions. In this work, we use the term volume classification as synonym for transfer function specification.

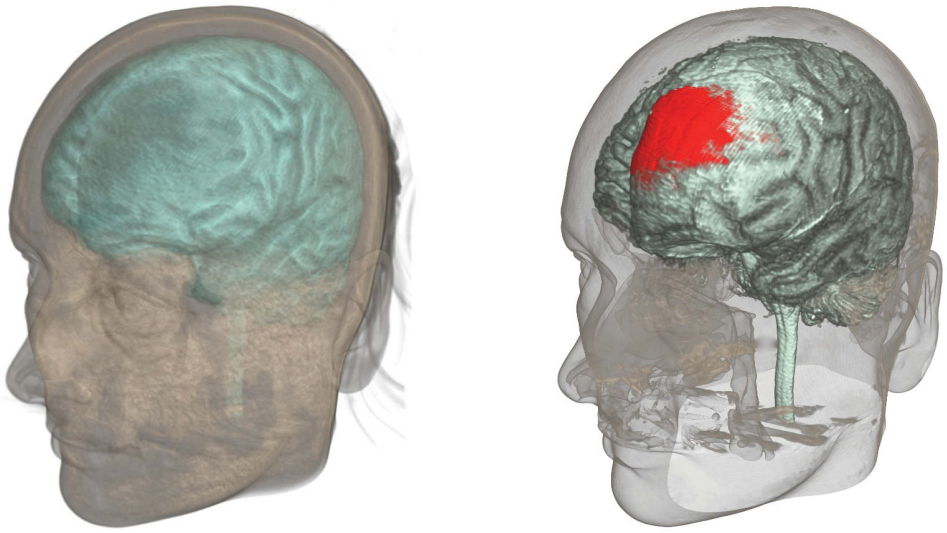
Interactive transfer function design is in many application cases essential for an effective exploration of a volume data set, because it enables the user to quickly adjust

the visualization by emphasizing certain materials and suppressing others. Manual transfer function setup, however, tends to be a time-consuming and frustrating process, as the relation between the transfer function and the resulting rendering is often very complex and unintuitive, i.e., seemingly minor transfer function changes may have a drastic impact on the volume rendering. As a consequence, several automatic and semi-automatic techniques that assist the user in finding an appropriate transfer function have been proposed by the research community. But despite these efforts, there does still not exist a widely accepted approach for transfer function definition and the challenge of volume classification seems to be far from being resolved.

Volume segmentation is the process of partitioning a volume data set into volumetric subregions by uniquely assigning each voxel to a segment. Since segmentation operates in the spatial domain and is therefore able to distinguish voxels belonging to the same material class, segmentation is more powerful than classification in terms of flexibility and precision. On the downside, however, segmentation has to be performed as a pre-processing step, whereas classification is an integral part of the volume visualization pipeline and therefore facilitates a more direct exploration of a volume data set. The amount of interactivity involved in the segmentation process heavily varies, ranging from fully automatic model-based approaches to almost fully manual techniques that require the user to directly paint or encircle regions of interest on the volume slices.

Figure 1.1 illustrates the differences in feature detection quality achievable with classification and segmentation on a medical brain tumor scan. While the transfer function defined by ad-hoc classification is able to discriminate the brain from surrounding tissue, it fails to clearly separate the tumor region, because tumor and brain tissue share a common intensity range. The segmented data set, in contrast, permits a clear visual distinction between brain, tumor, and surrounding tissue.

In this dissertation, we present three novel approaches for interactive feature detection in volumetric data. Though the proposed techniques are located at different positions within the spectrum between volume classification and segmentation and thus target different use cases, they have in common that they provide a high degree of interactivity while enhancing the efficiency of the data analysis by automatic computation. First, we introduce a classification scheme for the semi-automatic generation of so-called LH transfer functions, which classify boundary voxels by the intensities of their neighboring materials. The approach does not require any user interaction within the transfer function domain, but allows the user to visually emphasize features of interest by directly marking them in the volume rendered image. Second, we present a shape-based classification approach that enables the user



(a) transfer function **(b)** segmentation

Figure 1.1: Feature detection in a brain tumor scan: The rendering (a) has been created by applying an interactively defined transfer function, while in (b) an expert's manual segmentation has been used. The tumor can only be visually separated in the segmented data set.

to distinguish features by their 3D shape and to assign individual optical properties to these. Third, we develop a semi-automatic segmentation technique that supports the user in creating a highly reliable volume segmentation by analyzing and conveying uncertainty in an intermediate random walker-based segmentation. Furthermore, we describe how the three feature detection approaches have been implemented in the Voreen volume rendering framework.

This thesis is structured as follows: Before presenting the novel feature detection approaches, we introduce the basic concepts of volume rendering in Chapter 2 and give a brief overview of the Voreen framework (Chapter 3), which we have used for implementing our techniques. In Chapter 4 we discuss the challenges of volume classification in further detail and review related work in this field. After these preparative chapters, we describe our semi-automatic LH transfer function generation scheme in Chapter 5, followed by a presentation of the shape-based classification approach in Chapter 6. After covering related work regarding interactive volume segmentation in Chapter 7, we expose our uncertainty-guided volume segmentation approach in Chapter 8. Chapter 9 finally summarizes our work and indicates possible directions for future research.

The contributions presented in this thesis are based on the following publications: The LH transfer function generation scheme described in Chapter 5 has been presented at the Vision, Modeling, and Visualization Workshop (Praßni et al., 2009). The shape-based classification approach exposed in Chapter 6 has been presented at the IEEE Pacific Visualization Symposium (Praßni et al., 2010b). The uncertainty-aware volume segmentation system described in Chapter 8 has been presented at the IEEE Visualization Conference (Praßni et al., 2010a). Further work in the field of volume visualization has been conducted by contributing to the winning entry of the IEEE Visualization Contest 2010 addressing the pre-operative planning of brain tumor resections (Diepenbrock et al., 2010) and to the related publication in the IEEE Computer Graphics and Applications journal (Diepenbrock et al., 2011). A semi-automatic approach for the quantification of gas-filled microbubbles has been presented at the Eurographics Workshop on Visual Computing for Biology and Medicine (Praßni et al., 2010c). In addition to the scientific research carried out, extensive contributions have been made to the development of the Voreen framework, whose fundamental concepts have been presented in a tutorial at the IEEE Visualization Conference (Ropinski and Praßni, 2010).

Chapter 2

Concepts of Volume Rendering

This chapter introduces the theoretical foundations of volume rendering and presents its most popular implementations, with a special focus on GPU-based techniques. Furthermore, the role of classification in volume visualization is highlighted.

Volume rendering in its most general definition denotes the process of generating a 2D image of a 3D scalar field. Major sources of volume data sets are volumetric scanning techniques such as computed tomography (CT), magnetic resonance imaging (MRI), and positron emission tomography (PET), which provide three-dimensional scans by measuring a physical property at discrete positions in space. These procedures are widely used in medical diagnosis, as they allow physicians a non-invasive insight into the interior of a patient's body, but there also industrial applications mainly in material testing. Volume data sets also originate from geological and seismic scans, or are the result of scientific simulations. Usually, volumetric data is represented by a discrete, rectangular grid of scalar data points referred to as *voxels* (volumetric pixel).

Formally, volume rendering has the task of projecting a 3D scalar field

$$\Phi : \quad \mathbb{R}^3 \rightarrow \mathbb{R}$$

to an image plane representing the screen. This projection is done with regard to the position and orientation of a virtual camera. Techniques for visualizing volumetric data can be roughly grouped into direct and indirect approaches. *Indirect volume rendering* extracts isosurfaces, i.e., surfaces consisting of points with constant scalar value, from the volume data and displays them using common polygonal rendering techniques. The most prominent approach for isosurface extraction is the *Marching Cubes* algorithm (Lorensen and Cline, 1987), which determines for each voxel the polygon needed to represent the part of the isosurface that crosses that particular voxel. The resulting polygons are then combined to a surface mesh. The main advan-

tage of using such an intermediate geometric representation are the low hardware requirements, since rendering of geometric data is natively supported by virtually all graphics systems. However, indirect volume rendering depends on the assumption that the relevant structures of a volume data set can be conveyed to the viewer by displaying a set of isosurfaces. This is typically not the case for amorphous gas-like materials, which do not feature a well-defined boundary (Bartz and Meißner, 1999). A further drawback of isosurface rendering is the binary classification inherent in it, as it is limited to express whether an isosurface passes through a voxel or not and is therefore unsuitable for conveying surface quality or measurement uncertainty (Levoy, 1988). Especially image artifacts introduced by the limitations of the scanning process often hamper a reliable surface extraction. For these reasons, *direct volume rendering* (DVR) techniques, which directly render the sampled volume data and thus avoid the potentially lossy geometry extraction step, are nowadays preferred over isosurface rendering for the vast majority of application cases.

In the remainder of this chapter, we discuss the mathematical foundations of direct volume rendering and present its most prominent realizations, namely *texture slicing* and *raycasting*.

2.1 Direct Volume Rendering

Direct volume rendering is generally based on the simulation of physical light transport within the volume to be visualized. The resulting image is obtained by computing for each pixel on the view plane the accumulation of all light contributions actually reaching this pixel in consideration of light interaction with intermediate matter. For achieving correct physical results, a plethora of optical effects, such as diffraction and scattering, would need to be considered. In order to reduce the high computational effort, however, usually the much simpler emission-absorption model (Sabella, 1988) is employed, which assumes that light travels along a straight line at which each voxel potentially emits light as well as absorbs incident light. Under these assumptions, the radiance I of a single light ray entering the volume from the background at the starting point $s = s_0$ and leaving it in direction of the camera at $s = D$ can be described by the *volume rendering integral*:

$$I(D) = I_0\, e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} q(s)\, e^{-\int_{s}^{D} \kappa(t)dt}\, ds \tag{2.1}$$

The first term represents the background light I_0, which is attenuated during its way through the volume. The second term describes the accumulation of the integral

radiances $q(s)$ that are emitted along the ray and attenuated by the remaining medium between the emission point s and the exit point D. The local light absorption is expressed by $\kappa(t)$. By defining the transparency T of the material between s_1 and s_2 as

$$T(s_1, s_2) := e^{-\int_{s_1}^{s_2} \kappa(t)dt} ,$$

we can simplify Equation 2.1 to obtain the most common description of direct volume rendering:

$$I(D) = I_0\, T(s_0, D) + \int_{s_0}^{D} q(s)\, T(s, D)\, ds \tag{2.2}$$

Since volume data is sampled on a discrete grid, the volume rendering integral can typically not be evaluated analytically, but has to be approximated by a Riemann sum over n equidistant intervals $s_0 < s_1 < ... < s_n = D$. Assuming the accumulated radiance at the position s_{i-1} to be known, we can obtain an iterative formulation of Equation 2.2:

$$I(s_i) \;=\; I(s_{i-1})\, \underbrace{T(s_{i-1}, s_i)}_{:=T_i} + \underbrace{\int_{s_{i-1}}^{s_i} q(s)\, T(s, s_i)\, ds}_{:=c_i} \;=\; I(s_{i-1})\, T_i + c_i \tag{2.3}$$

where T_i and c_i denote the transparency and light emission of the ith interval, respectively. The radiance at the exit point of the light ray is then given by

$$I(D) = I(s_n) = I(s_{n-1})T_n + c_n = (I(s_{n-2})T_{n-1} + c_{n-1})T_n + c_n = ... ,$$

which can be re-written to the *discretized volume rendering integral*:

$$I(D) = \sum_{i=0}^{n} c_i \left(\prod_{j=i+1}^{n} T_j \right) , \quad \text{with } c_0 = I(s_0) \tag{2.4}$$

Although Equation 2.4 is already applicable for discrete volume data, directly evaluating it is computationally inefficient, since for each sampling point i along a ray the accumulated transparency of the remaining medium between i and the exit point D has to be determined. These redundant computations can be avoided by simultaneously accumulating color and opacity during the ray traversal, which is called *compositing*. In the most widely used *front-to-back compositing* scheme, which traverses the ray starting from the camera position into the volume, the color C_{src} at the current sampling point is attenuated by the so far accumulated opacity $(1 - \alpha_{dst})$

and then added to the accumulated ray color C_{dst}:

$$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst})\, C_{src} \tag{2.5}$$

$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst})\, \alpha_{src} \tag{2.6}$$

When traversing the ray in reverse direction from the back of the volume to the camera, the *back-to-front compositing* scheme has to be applied:

$$C_{dst} \leftarrow (1 - \alpha_{src})\, C_{dst} + C_{src} \tag{2.7}$$

Back-to-front compositing does not require an iterative update of the accumulated opacity, because the opacity of the back part of the ray does not influence the color contribution of the current sample. Nevertheless, front-to-back compositing is usually preferred, as it allows for an early termination of the ray traversal when the ray is saturated, i.e., the accumulated opacity has reached a value of 1.0, and therefore subsequent samples have no contribution to the final color of the ray.

2.1.1 Transfer Functions

So far we have implicitly assumed that light emission and absorption coefficients are known for each voxel in the volume. However, the scalar values stored in a volumetric data set usually represent a physical quantity with no relation to light interaction. Therefore, the rendering system itself or, in most cases, the user has to decide how certain structures should look by assigning optical properties to them. This assignment is typically done using a *transfer function* (TF) that maps the abstract data values to color and opacity. Transfer functions can be grouped into different categories according to their domain. *Intensity-based* transfer functions consider only the intensity value for assigning optical properties and are therefore called *one-dimensional* (1D TF). *Multi-dimensional* transfer functions additionally take derived measures of the intensity into account. Prominent representatives of this class are *intensity-gradient-based* transfer functions (2D TF), whose domain is the two-dimensional space of intensity and gradient magnitude. Their main advantage over 1D transfer functions is their ability to distinguish boundaries from more homogeneous regions.

While the optical property mapping is mathematically and technically rather simple, the challenge in many applications lies in the design of an appropriate transfer function that does not only highlight the desired features of interest, but also effectively suppresses unwanted structures in order to reduce occlusions. Although pre-defined or automatically generated transfer functions can be applied in some cases, the usual volume visualization workflow requires the user to specify the

transfer function manually. Throughout this work we refer to the process of finding an appropriate transfer function as *classification*. Chapter 3.2.1 presents graphical editors for 1D and 2D transfer functions provided by the Voreen framework.

Besides the classical transfer function types described so far, a variety of more specialized transfer function domains has been proposed by the research community (see Chapter 4). In the two-dimensional LH transfer function space, which we exploit for the boundary classification scheme presented in Chapter 5, boundary voxels are classified by the lower and higher intensities of the two neighboring materials that form the boundary. In Chapter 6, we introduce a novel transfer function domain based on the shape of volumetric features.

2.1.2 Volume Illumination

Traditional geometry-based computer graphics makes extensive use of simulated illumination in order to enhance realism. Real-time applications are usually restricted to local illumination models, which compute light reflection exclusively from local surface information while neglecting global effects such as shadowing. The most frequently used local illumination model is Phong lighting (Phong, 1975). It models the light reflected at a surface point as a combination of *ambient*, *diffuse*, and *specular* reflections, expressed as RGB colors:

$$I = k_a I_a + k_d I_d (\vec{n} \cdot \vec{L}) + k_s I_s \left(\vec{r} \cdot \vec{v}\right)^n \tag{2.8}$$

The coefficients k_a, k_d, and k_s are RGB triplets that specify the respective ambient, diffuse, and specular reflectances of the material (material colors). The RGB color values I_a, I_d, and I_s define the corresponding emissions of the light source. Note that the multiplication of RGB triplets is performed component-wise. The first term of the Phong equation describes the ambient reflection, which is independent of both viewer position and light direction and serves as rough approximation of indirect illumination in the scene. The second term represents the diffuse, or Lambertian, reflection, which depends on the incidence angle between the direction $\vec{L}$ of the incoming light and the surface normal $\vec{n}$. The third term models specular highlights as occurring on shiny surfaces. The amount of specular reflection is determined by the angle between the perfect reflection direction $\vec{r}$ and the viewing direction $\vec{v}$. The shininess coefficient n modulates the size of specular highlights.

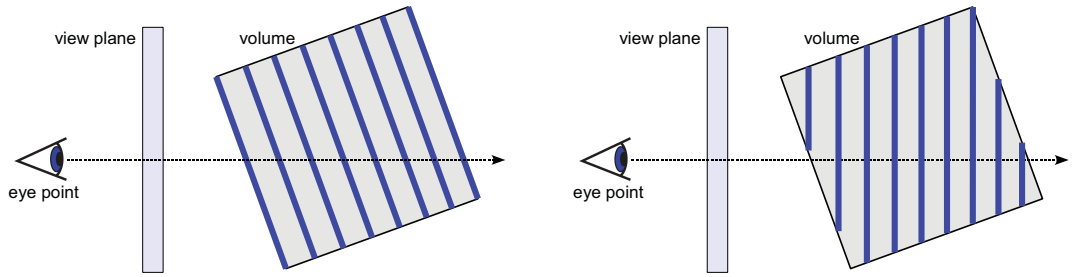
Although direct volume rendering is based on the simulation of light transport, in its basic formulation the light emission of a single voxel is solely determined by its scalar value. Yet many volume rendering applications can benefit from incorporating illumination effects caused by an external light source, since illumination not only

helps the user in recognizing certain surface properties, such as orientation and roughness, but also adds additional depth cues and thus enhances spatial comprehension of the rendered data set. In principle, Phong lighting can be integrated into direct volume rendering by using the voxel color provided by the transfer function as material coefficients in Equation 2.8. The computed light reflection is then used as light emission during compositing. However, in contrast to geometric scenes where a surface normal is either directly assigned to each polygon or can be easily computed from its vertices, volume data sets do not contain explicit surface information. Instead, the gradient at a voxel position is often used as estimate for the surface normal. While the gradient quality is usually sufficient for the illumination of data sets with clear object boundaries, such as CT scans, gradient-based lighting is less applicable to data sets with high levels of noise, such as MRI and PET scans.

2.2 Texture Slicing

2D texture slicing was among the earliest implementations of direct volume rendering that were able to provide interactive frame rates. In its initial formulation, as described by Hibbard and Santek (1989), the volume is decomposed into a stack of axis-aligned 2D slices, which are then projected onto the image plane either in front-to-back or back-to-front order and thereby combined using the according compositing scheme (see Figure 2.1a). The slicing direction, parallel to one of the volume's major axes, is chosen such that the angle between the slice normal and the viewing rays is minimized, in order to avoid that rays pass between two slices without intersecting them. As a consequence, three orthogonal slice stacks are created, which can be switched between frames depending on the current camera position. Besides its simplicity, the main benefit of 2D slice rendering are its low hardware requirements, as it can be implemented on any graphics system that supports 2D texture mapping and per-pixel blending operations. The use of fixed axis-aligned slice stacks, however, has severe drawbacks. Most obvious, the sampling rate is determined by the number of slices and therefore cannot be changed at rendering time. Furthermore, the effective sampling rate decreases with the angle between the viewing direction and the slice normal, which may result in rendering artifacts, especially when looking along a diagonal of the volume. Finally, changing between different slice stacks often causes a disturbing flickering.

3D texture slicing (Cullip and Neumann, 1994) effectively solves these problems by exploiting the 3D texture mapping capabilities of modern graphics hardware in order to slice the volume in the current viewing direction (see Figure 2.1b). Since the viewing direction may change between frames, a static decomposition of the volume



(a) 2D texture slicing **(b)** 3D texture slicing

Figure 2.1: Comparison of texture slicing approaches: 2D texture slicing decomposes the volume in a pre-processing step into axis-aligned slices and projects them onto the view plane, either in front-to-back or back-to-front order. The 3D approach slices the volume perpendicularly to the current viewing direction.

into 2D slices is then no longer possible. Instead, the volume is directly sampled in 3D texture space by assigning 3D texture coordinates to the vertices of the polygons that have been obtained by intersecting the volume's bounding box with a set of equidistant planes parallel to the current view plane. The slice compositing remains the same as with 2D texture slicing.

While the use of 2D slice rendering is nowadays mainly confined to outdated graphics systems without 3D texturing support, 3D texture slicing still enjoys a high popularity as it provides a decent image quality at interactive frame rates even on commodity graphics hardware.

2.3 Raycasting

With the ever-increasing prevalence of programmable graphics hardware, *GPU-based raycasting* has become one of the most popular direct volume rendering techniques. First described by Röttger et al. (2003), it is arguably the most straight-forward implementation of the direct volume rendering integral. Its basic idea is to cast a single ray from the eye point through each pixel on the image plane into the volume and then to sample the volume data at discrete positions along this ray. The color and opacity values obtained by the application of the transfer function to the sampled data values are accumulated according to the front-to-back compositing scheme. The resulting color is then assigned to the pixel the ray was cast through. Figure 2.2 illustrates this proceeding. The ray traversal is implemented on the GPU in a fragment shader, which is executed for each image pixel by rendering a *proxy geometry*, i.e., a

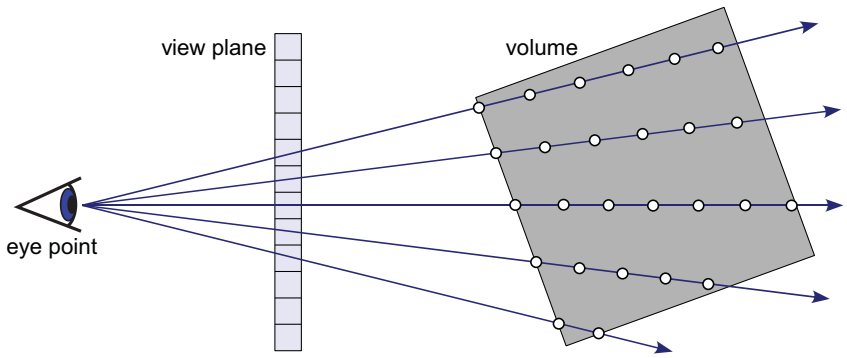
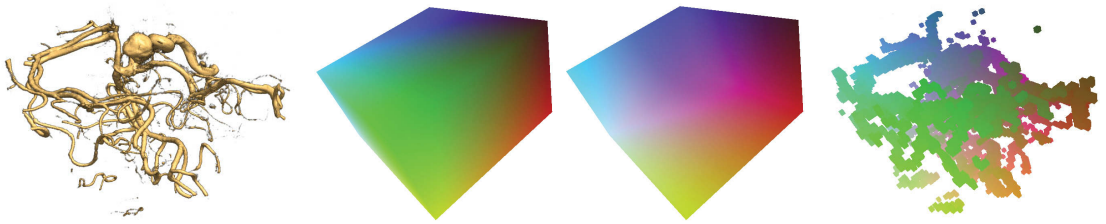


Figure 2.2: Principle of volume raycasting: Through each pixel on the view plane a ray is cast from the camera position into the volume. The pixel colors are obtained by accumulating the voxel emissions at discrete sampling points along each ray according to the front-to-back compositing scheme.

polygonal representation of the volume in space, with regard to the current viewing parameters. Typically the data set's bounding box is used as proxy geometry. In order to be accessible by the fragment shader, the volume data needs to be stored in a 3D texture in the graphics memory.

Krüger and Westermann (2003) have extended the basic raycasting approach by performance optimizations that exclude those parts of the volume from the ray traversal that are not visible on the screen, either due to occlusion or transparency. *Early ray termination* stops the ray traversal when the accumulated opacity has reached a value of 1.0, since in this case the remaining samples along the ray do not contribute to the resulting ray color (compare Equation 2.5). The opacity threshold might even be chosen slightly below 1.0, because further contributions are usually not visually perceivable. Early ray termination is most effective for dense data sets, where rays have a high probability of hitting an opaque surface shortly after entering the volume. However, the performance gain heavily depends on the transfer function, as rays crossing only semi-transparent objects never become fully saturated and therefore need to be traversed till their end point.

A further modification of the Krüger-Westermann approach regards the determination of ray parameters, i.e., the ray start and end points. Instead of computing the ray parameters analytically in the fragment shader, it utilizes the geometry pipeline in order to generate two 2D textures that store the (x, y, z) texture coordinates of the ray start and end points encoded as RGB colors. These ray parameter textures, often called *entry-exit points* (EEP), are obtained by rendering the front and the back faces of the volume proxy geometry, respectively. The fragment shader is then able to retrieve



(a) raycasted image **(b)** entry points **(c)** exit points **(d)** optimized entry points

Figure 2.3: Ray parameter textures encoding ray start and end points for GPU-based raycasting. (b) and (c) are generated by rendering a cubic volume proxy geometry. (d) is the result of an optimized geometry excluding invisible parts of the data set.

its ray parameters by performing lookups in both textures at the fragment position. While the intermediate texture generation step might appear more complex than the direct computation of the intersection points between the viewing ray and the volume bounding box in the shader, it opens up further optimization possibilities. For example, *empty space skipping* can be implemented by rendering an optimized proxy geometry that leaves out invisible parts of the volume, i.e., regions that are assigned zero opacity by the current transfer function. Figure 2.3 shows ray parameter textures generated from a simple cubic proxy geometry, along with optimized entry points.

Chapter 3

Voreen - The Volume Rendering Engine

This chapter provides an overview of the Voreen framework, which has served us as basis for the realization of the proposed feature detection approaches. In addition to the central data-flow paradigm, the main user interface components are presented. The extendibility of the framework is demonstrated with an example implementation of GPU-based raycasting.

The effective analysis of volumetric data often requires the flexible combination of multiple rendering and data processing techniques. For example, a user examining a data set with direct volume rendering might realize that a certain pre-processing step needs to be applied to the volume data in order to obtain a meaningful visualization. The resulting rendering might then be further enhanced by adding image processing filters. The Voreen (Volume Rendering Engine) project (Meyer-Spradow, 2009; Meyer-Spradow et al., 2009) provides a C++-based rapid-prototyping framework that supports such dynamic volume analysis workflows. It employs the visual programming paradigm in order to allow both domain users and visualization developers to efficiently combine built-in components with custom techniques to create data processing pipelines specifically tailored to certain use-cases.

3.1 Data-flow Concept

The design of the Voreen framework revolves around the concept of *data-flow networks*. These networks consist of modular units, called *processors*, which encapsulate rendering and data processing algorithms. A processor operates on input data it receives from its *inports* and outputs the processed results via its *outports*. The data-flow is established by unidirectional connections from each of the outports in the network

to one or multiple inports of processors that are to consume the respective output. Processor ports are typed, i.e., each port transmits a certain type of data such as a 2D image, a 3D volume, geometric data, or a collection of objects of these basic types. Processors furthermore have *properties* that are used for parameterizing the encapsulated algorithms. The Voreen framework offers various types of properties ranging from primitive numeric types to complex rendering parameters such as camera position and transfer function. Properties of different processors can be linked in order to synchronize their values.

The main advantage of the data-flow concept is that processors are autonomous building blocks, which can be arbitrarily combined as long as their port types match. Processors do especially not need any knowledge about the network topology, e.g., which predecessors and successors they are connected to. Instead, a processor's behavior and thus its output data is fully determined by its input data and property configuration. The only exception from this rule of locality are *coprocessor ports* that enable processors to directly communicate with each other via function calls instead of data-flow. A coprocessor port is usually used when a processor requires certain functionality whose concrete implementation should be interchangeable at runtime, similar to the strategy design pattern in object-oriented programming.

The generic data-flow concept facilitates the extension of the framework with custom functionality, as developers can easily implement a new rendering or data processing algorithm as a Voreen processor that communicates with the framework via a clean and stable interface composed of data ports and properties.

3.2 User Interface

The VoreenVE (visualization environment) application provides a graphical user interface (GUI) to the Voreen framework (see Figure 3.1). Its main component is a graphical network editor that enables the user to interactively manipulate the network topology by adding or removing processors, establishing or deleting port connections, and manipulating property links. For each `Canvas` processor in the current network a corresponding canvas widget is shown. Properties can be accessed via automatically constructed GUI components. Additionally, tools for the inspection of intermediate data processing results, performance analysis, and animation recording are available.

3.2.1 Transfer Function Editor

A GUI component specifically relevant in the context of this work is the interface for the manual specification of transfer functions. Voreen provides separate visual

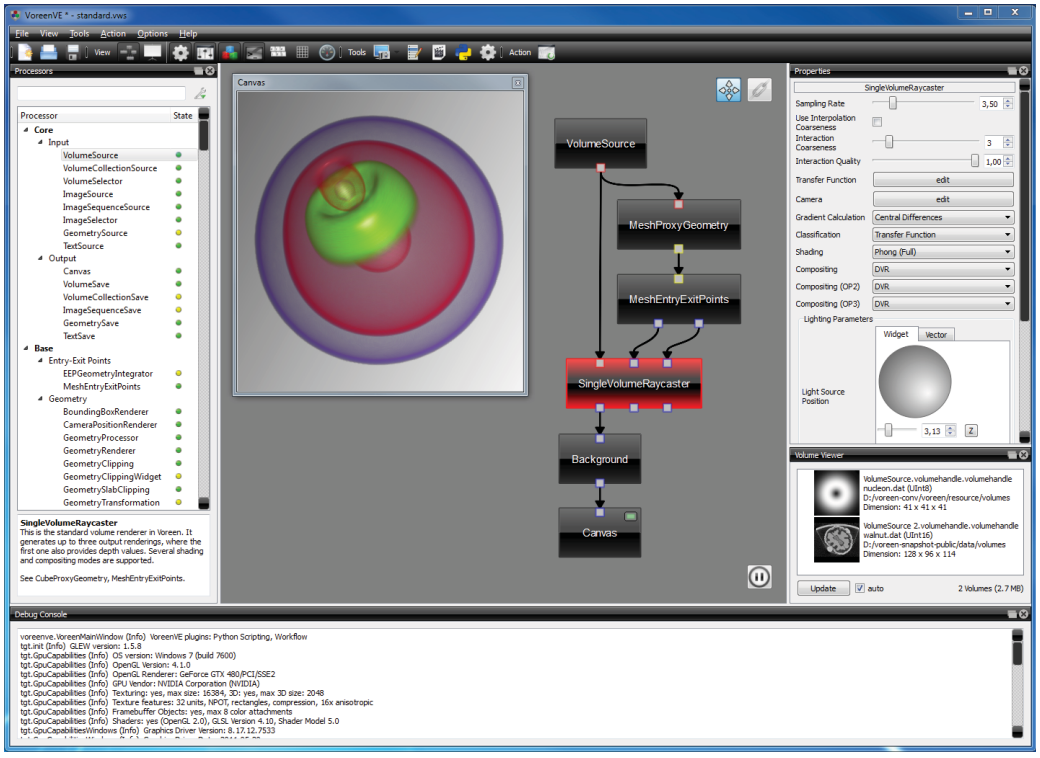
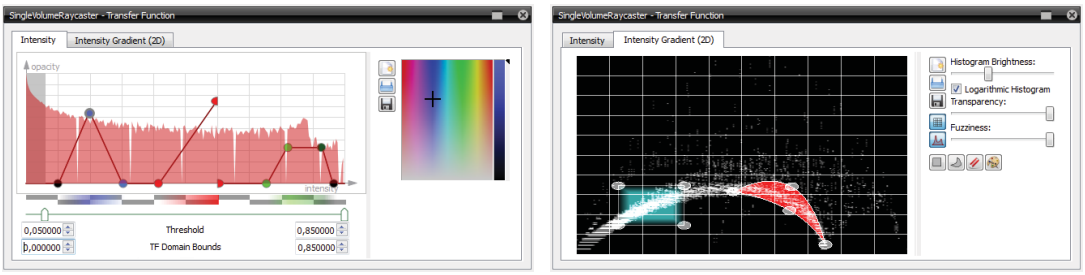


Figure 3.1: Screenshot of the VoreenVE application for the manipulation and parametrization of data-flow networks. Left of the central network editor a list of available processors is shown. The properties of the selected processor are presented by the property list widget on the right. At the bottom a debugging console is visible. The free-floating canvas widget displays the input image of the Canvas processor.



(a) 1D transfer function editor **(b)** 2D transfer function editor

Figure 3.2: Voreen transfer function interfaces. The 1D editor allows the user to define an intensity transfer function by manipulating its graph using control points. The 2D editor is based on color-assigned widgets that are placed within a rectangle representing the intensity-gradient transfer function space.

editors for intensity transfer functions (1D), which map from voxel intensity to an RGBA color value, and intensity-gradient transfer functions (2D), which additionally take the gradient magnitude at a voxel position into account.

As presented in Figure 3.2a, an intensity transfer function is defined as a piecewise linear mapping over the data set's intensity range. The user manipulates the function's graph through a control point-based interface. Each control point is assigned a RGB color as well as an opacity value, which is expressed by the point's vertical position. Function values between control points are obtained by linear interpolation. Split control points can be used for modeling discontinuities in the mapping. The transfer function graph is underlayed by an intensity histogram of the data set.

The 2D transfer function editor shown in Figure 3.2b is based on widgets that are placed within a rectangle representing the transfer function's domain: the horizontal direction corresponds to the voxel intensity and the vertical axis represents the gradient magnitude. The subset of the intensity-gradient space that is covered by a widget is mapped to the widget's color and opacity, while the subspace between widgets is assigned zero opacity. A 2D histogram of intensity and gradient magnitude facilitates the identification of features within the transfer function space.

3.3 Integration of GPU-based Raycasting

As a practical example, we describe the integration of GPU-based raycasting (compare Chapter 2.3) into the Voreen framework. Figure 3.3 shows a basic data-flow network implementing raycasting and the resulting rendering of a medical CT scan. The `VolumeSource` processor at the top of the network loads the volume data from disk and transfers it via its outport to the connected consumer processors. The three processors constituting the actual volume raycasting pipeline are marked in red: The `CubeProxyGeometry` processor creates an axis-aligned bounding box of its input volume as proxy geometry and delivers it to the `MeshEntryExitPoints` processor, which generates two 2D OpenGL textures storing the ray entry and exit points, respectively, according to the Krüger-Westermann approach. The ray parameter textures are then sent via two separate image ports to the `SingleVolumeRaycaster`, which implements the actual ray traversal in a GLSL fragment shader and writes the resulting rendering to its first outport. The subsequent `GeometryProcessor` enriches the image with a wireframe rendering of the volume's bounding box by calling the `BoundingBoxRenderer` as a coprocessor. Finally, the `Background` processor adds a color gradient background and sends the resulting image to the `Canvas` processor for display in the associated output window.

We would like to emphasize that the presented raycasting implementation exhibits

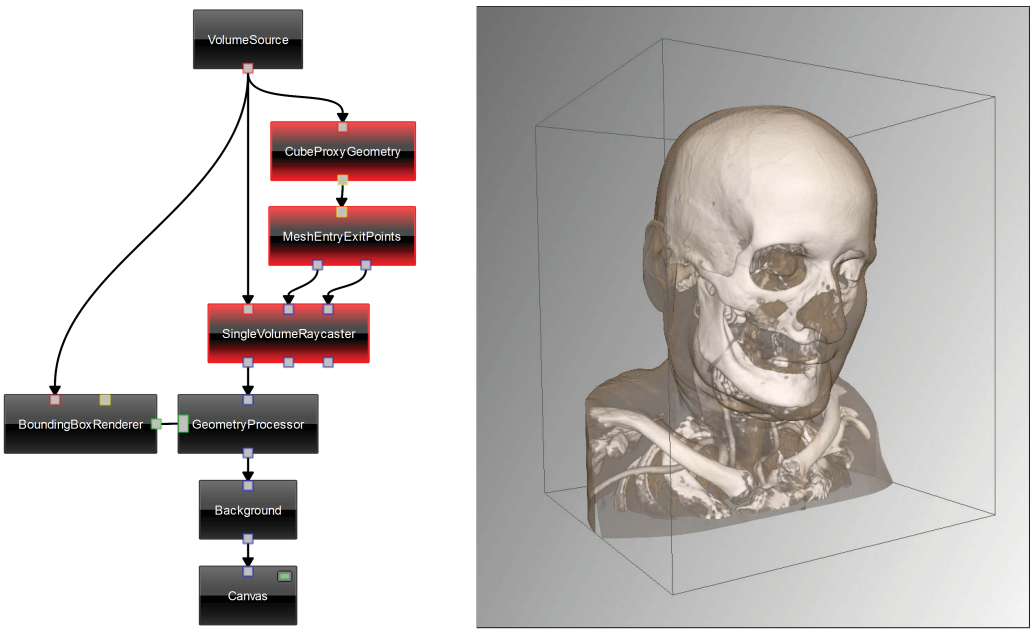


Figure 3.3: Voreen data-flow network implementing GPU-based raycasting, along with the resulting rendering of a medical CT scan.

a significantly higher degree of granularity than found in most other volume visualization systems, which typically encapsulate the entire rendering pipeline in one monolithic block (Meyer-Spradow et al., 2009, p. 7). The separation of proxy geometry creation, ray parameter computation, and ray traversal provides a high flexibility. For example, the basic `CubeProxyGeometry` can be exchanged at run time by a more advanced variant supporting empty space skipping. A spherical projection can be easily integrated by replacing the default `MeshEntryExitPoints` processor with a custom implementation that computes the appropriate ray parameters. In both cases the ray traversal code in the `SingleVolumeRaycaster`'s fragment shader can remain unchanged.

3.4 Extending the Framework

The Voreen framework has been designed with special focus on extendibility. Developers can conveniently integrate new techniques by adding *modules*, which primarily contain custom implementations of the three main components of a Voreen data-flow network: processors, ports, and properties. A processor encapsulates a rendering or data processing algorithm and is implemented by a class inheriting

from the `Processor` base class. The actual functionality of a processor is located in its `process()` method, while its `initialize()` method is used for initializations such as shader loading. There are no limitations imposed on the kind of operations a processor may perform, especially the OpenGL API and the filesystem may be accessed directly. Usually, a custom processor uses the standard port types provided by the framework for accessing and storing input and output data, but developers may also add new port classes for transferring data structures not directly supported by the framework. Similarly, a processor can be parametrized by an arbitrary combination of standard and custom property types. A custom property type, however, must be accompanied by a corresponding GUI widget that can be used by the framework for representing the property in the user interface. Besides these network components, Voreen modules may also add volume readers and writers for supporting further data formats.

The components provided by a custom module have to be integrated into the framework on two different levels. First, the C++ code files need to be incorporated into the build process. Voreen uses the CMake build system (Martin and Hoffman, 2004) for generating native makefiles for a specific build environment from compiler-independent configuration files. Accordingly, each module is required to provide a CMake file that lists all of its code files. Second, module resources have to be registered at runtime, which is done by a subclass of `VoreenModule` that instantiates the module's classes using a factory pattern.

Chapter 4

Volume Classification

Classification is an integral part of interactive volume visualization. In this chapter we review related work in the field of transfer function specification with special focus on semi-automatic classification techniques. Furthermore, we reason the need for higher-dimensional transfer functions, which additionally take derived measures of the intensity value into account.

A transfer function maps the abstract scalar values of a volume data set to optical properties, such as color and opacity, which are then used in the volume rendering process. When designing a transfer function manually in a trial-and-error process, the user modifies the graph of the transfer function directly. A pure intensity-based transfer function (1D) is usually constrained to a piecewise linear function, which is edited by manipulating control points or placing primitives like peaks and ramps above value ranges. An example of such a control-point-based transfer function editor has been presented in Chapter 3.2.1.

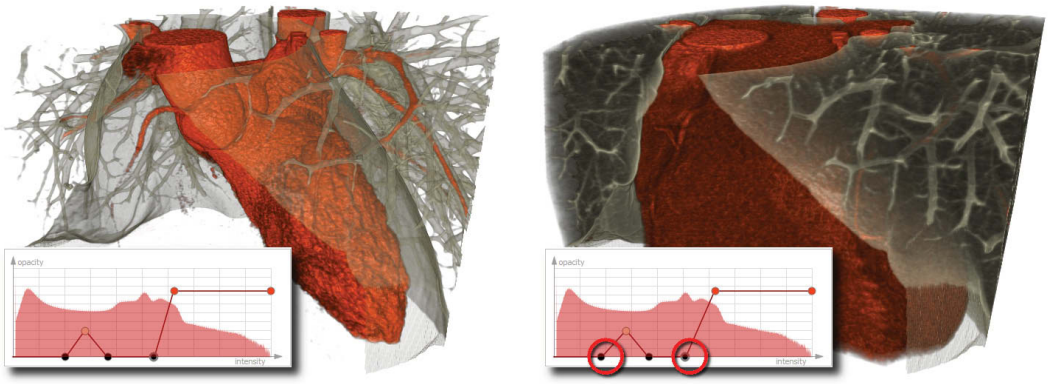
Although manual transfer function setup can certainly be regarded as "a vehicle of exploration by which the observer comes to understand the data" as stated by Pfister et al. (2001) in the Transfer Function Bake-Off, this approach also suffers from serious drawbacks. Its main problem is the often complex and unintuitive relation between the transfer function and the rendered image, as subtle transfer function changes may have a drastic and unanticipated impact on the resulting visualization (see Figure 4.1). Thus creating appropriate transfer functions tends to be a time-consuming and often frustrating process. The use of higher-dimensional transfer functions further complicates the user interaction, e.g., manually editing a three-dimensional function appears to be impractical. As a consequence, multiple automatic and semi-automatic methods that assist the user in finding transfer functions have been proposed. But before giving an overview of the most relevant classification techniques, we illuminate the inherent limitations of 1D transfer functions and thus motivate the use of higher-dimensional transfer functions in the following section.

4.1 Limitations of One-dimensional Transfer Functions

Volume data sets usually consist of homogeneous regions, i.e., regions of (nearly) constant intensity, and boundaries between them. We call these homogeneous regions *materials*. Due to limitations in the scanning process, boundaries in non-artificial data sets are never sharp but are characterized by a smooth transition between the two intensities of the neighboring materials, which is known as *partial volume effect*. Therefore, a boundary is represented by an intensity interval that is bounded by the neighboring materials' intensities. As a consequence, a material cannot be classified in isolation with a 1D transfer function when a boundary's intensity range spans the material's intensity. In that case, a 1D transfer function inevitably assigns the same optical properties to the material as well as to a part of the respective boundary. Classifying boundaries instead of materials is often also not possible because of the potential overlap of their intensity intervals.

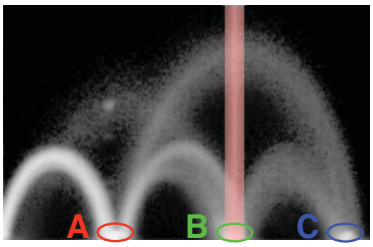
Since boundaries are regions of changing intensity, they are characterized by high gradient magnitudes, whereas the gradient magnitude within materials is near zero. Hence, a *2D histogram* whose axes correspond to the intensity and the gradient magnitude, respectively, is often used to facilitate the analysis of a volume data set. While materials are simply represented by blobs in such a histogram, boundaries usually appear as arcs (Kniss et al., 2002). Figure 4.2 shows a 2D histogram of the CT scan of a human tooth and illustrates that it is impossible to isolate the dentin in this data set with a 1D transfer function, since a part of the enamel-background boundary has the same intensity range as the dentin.

In order to circumvent the demonstrated limitations of 1D transfer functions, one might be tempted to simply suppress boundaries during the rendering by ignoring voxels with a gradient magnitude above a certain threshold. This approach, however, implicates several disadvantages. First of all, finding a proper global threshold turns out to be difficult for data sets containing much noise, since noisy materials have higher gradient magnitudes by themselves and can thus be hard to distinguish from boundaries. Furthermore, boundaries appear in a volume rendering as surfaces, which are very important for conveying the shape of volumetric regions to the user. Surface information is also essential for the application of lighting models as described in Chapter 2.1.2. The fact that the suppression of boundaries discards important information is the main reason why, to our knowledge, no volume visualization system follows this approach in order to cope with the partial volume effect. Instead, the application of multi-dimensional transfer functions, which incorporate not only the intensity value but also derived data such as the gradient magnitude at a voxel position, is nowadays considered best practice.

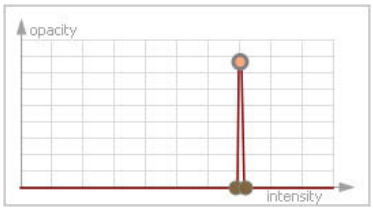


(a) (b)

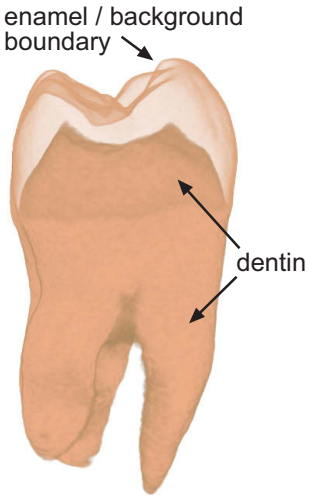
Figure 4.1: The challenge of volume classification demonstrated on a CT scan of a human heart: In (a), the blood stream and the semi-transparently rendered vessel walls/heart surface are cleanly classified by the overlayed 1D transfer function. In (b), a minor variation of the transfer function in form of two small shifts of the marked control points to the left introduces strong occlusions.



(a) 2D histogram



(b) 1D transfer function



(c) resulting rendering

Figure 4.2: The 2D histogram (a) of the tooth data set contains several arcs corresponding to boundaries between materials, while the materials are represented by blobs at the bottom of the histogram (low gradient magnitude). The dentin's blob (B) is covered by the arc of the boundary between the background (A) and the enamel (C). Therefore, the 1D transfer function (b) fails to cleanly classify the dentin in the resulting rendering (c), where the undesired enamel-background boundary is visible.

4.2 Image-centric Classification

Image-centric volume classification techniques support the user in finding an appropriate transfer function by analyzing the volume rendered image. One important representative of this class is the *Design Gallery* approach proposed by Marks et al. (1997). They sample the space of possible transfer functions and generate a rendering for each sample. The user evaluates the quality of the offered renderings in an interactive process in order to find the best one. The strength of this method is the intuitive, image-based user interface shielding the user completely from the complexity of transfer function space. It is, however, difficult to automatically sample the entire transfer function space without missing any significant subspace, especially for higher dimensional transfer functions. Additionally, the number of samples has to be small, since the user is required to assess the quality of all presented visualizations.

While design galleries were originally designed as a general approach for parameter setting in the field of computer graphics, König and Gröller (2001) further adapted them to the task of transfer function specification. They treat each specification domain such as data value, opacity, and color separately and arrange thumbnail renderings according to the value ranges they represent. When the user has specified a value range for each domain, these ranges are combined to the final transfer function. This technique can certainly help the user getting an overview of the transfer function space, but it is restricted to separable transfer functions, i.e., each dimension of the transfer function domain has to be specified independently from the others.

He et al. (1996) introduced the breeding of transfer functions with genetic algorithms. They define the fitness of intermediate transfer function generations as the quality of the resulting visualization. The image quality is evaluated either by the user, or by applying some predefined criteria such as entropy or variance. Choosing suitable criteria for an automatic assessment, however, is a difficult task by itself.

Wu and Qu (2007) presented a framework for combining and editing existing transfer functions based on genetic algorithms and user-drawn sketches. The user operates on volume renderings, obtained from pre-generated transfer functions, by sketching the features that are desired to be visible in the combined rendering. In a similar way, it is possible to remove unwanted structures from an input rendering. The system employs a contour-based image similarity measure focusing on the user-sketched structures in order to evaluate the fitness of candidate transfer functions. While achieving respectable results in combination with an intuitive user interaction, this technique heavily depends on the quality of the pre-generated transfer functions.

A further sketch-based interface for the design of 1D transfer functions has been proposed by Ropinski et al. (2008). After the user has identified a feature of interest by

drawing strokes close to its silhouette, their system generates a suitable component transfer function based on a histogram analysis.

4.3 Data-centric Classification

In contrast to evaluating the rendered image, data-centric classification approaches assist the user by analyzing the volume data directly. The *Contour Spectrum* proposed by Bajaj et al. (1997) is such an approach. It displays characteristics of the volume data set in compact form in order to support the user by finding appropriate isovalues for an informative visualization. Generally spoken, the contour spectrum consists of computed metrics over the range of data values and is presented to the user as one or multiple functions of data value. The authors suggest the area of an isocontour as well as the integral of gradient length over an isocontour as effective metrics. For instance, when the user is interested in visualizing boundaries, he may look for local maxima in the plot of the gradient integral function and choose the corresponding data values as isovalues. Even though this method can provide the user with helpful information about a data set, it suffers from its restriction to very simple transfer functions consisting of a set of isovalues only.

The classification techniques mentioned so far use only 1D transfer functions. Levoy (1988) was the first to propose taking into account the gradient length in volume rendering. In their seminal paper on semi-automatic transfer function specification, Kindlmann and Durkin (1998) show how to additionally exploit the second order derivative in order to semi-automatically extract boundaries from volumetric data. Although this probably forms the basis for most multidimensional transfer function approaches, the transfer function itself was still 1D and the gradient magnitude was only considered in order to modify the opacity. Their method has been extended by Kniss et al. (2002) who introduced the nowadays widely used two-dimensional space of data value and gradient magnitude as transfer function domain (see Chapter 4.1). They also proposed a dual-domain interaction that eases the setup of higher-dimensional transfer functions by data probing within the volume.

In the recent years, a variety of alternative transfer function domains has been proposed. Šereda et al. (2006a) employ the LH space for classification. They construct a two-dimensional LH histogram by mapping each boundary voxel to its LH coordinates, which are determined by the intensities of the two neighboring materials that form the boundary. LH histograms are well-suited for boundary detection, since boundaries appear as separate blobs in the LH space. We employ the LH space for a semi-automatic classification approach that enables the user to define a transfer function by directly interacting with the volume rendered image (see Chapter 5).

Caban and Rheingans (2008) use textural properties of volume regions instead of intensity and gradient values to control optical properties. Hadwiger et al. (2008) use a pre-computed feature volume to store the results of a region growing process over a parameter domain. This allows visualization of different feature sizes without a costly re-computation of the segmentation. Correa and Ma (2008) describe size transfer functions that map the relative size of features to color and opacity by utilizing scale fields for continuous representation of size. Patel et al. (2009) use mean and variance of voxel intensities for a transfer function specification that is robust to noise. Hladuvka et al. (2000) and Kindlmann et al. (2003) employ the surface curvature for classification. They use the two-dimensional space of the two principal curvatures as transfer function domain in order to visualize the shape of surfaces. Since the data value at a sample point is completely ignored, curvature-based transfer functions are not capable of distinguishing between materials.

The so far presented techniques focus on the specification of transfer functions for data sets the user has no or little previous knowledge about. In practice, however, users are often confronted with a set of similar data sets from a specific domain, e.g., physicians regularly examining cardiac MRI data sets. Rezk-Salama et al. (2006) support such domain users by adapting a semantic model for a selected application case. They integrate expert knowledge into the user interface in order to reduce the complexity of optical property assignment for non-expert users.

Despite the numerousness of promising approaches in the field of transfer function generation, there still does not exist any widely accepted technique for this central task of volume visualization.

Chapter 5

Efficient Boundary Detection and LH Transfer Function Generation

In this chapter we present an efficient technique for the construction of LH histograms which, in contrast to previous work, does not require an expensive tracking of intensity profiles across boundaries and therefore allows an LH classification in real time. Furthermore, we propose a volume exploration system for the semi-automatic generation of LH transfer functions, which does not require any user interaction within the transfer function domain. During an iterative process the user extracts features by marking them directly in the volume rendered image. The system automatically detects a marked feature's boundary by exploiting our novel LH technique and generates a suitable component transfer function that associates user-specified optical properties with the region representing the boundary in the transfer function space. The component functions thus generated are automatically combined to produce an LH transfer function to be used for rendering.

Šereda et al. (2006a) introduced the two-dimensional LH space as transfer function domain. They assume that every voxel lies either inside a material or on the boundary between two materials with lower intensity F_L and higher intensity F_H, respectively. The LH histogram (see Figure 5.4), whose axes correspond to F_L and F_H, is built from the data set by accumulating boundary voxels with the same (F_L,F_H) coordinates, which are retrieved by analyzing the intensity profile across a boundary. The authors show that the LH histogram conveys information about a data set's boundaries in a more compact and robust way than common 2D histograms incorporating the intensity and gradient magnitude, because in LH histograms boundaries appear as blobs instead of arcs (compare Chapter 4.1). A further significant advantage of the LH space is that it allows an unambiguous classification of boundaries with

distinct LH values, which is not the case for the intensity-gradient space due to arch overlaps as shown by Kniss et al. (2002). These properties make the LH space a very attractive transfer function domain, especially for the semi-automatic classification of boundaries. Šereda et al. (2006b) further described a projection of the LH space to a 1D transfer function domain that allows the classification of complete objects instead of single boundaries.

A major drawback of the LH technique, however, are the complex computations required for determining the LH values, since for each boundary voxel the intensity profile across the boundary has to be analyzed by integrating the gradient field until either a constant area, a local extremum, or an inflex point is reached. Therefore, the LH classification cannot be performed in real time during the rendering process. Instead, an additional volume storing the LH values of all voxels has to be generated. Besides the required pre-computation time, this approach triples the memory consumption of the volumetric data to display, since the LH volume contains two additional channels per voxel and must match the original volume in resolution and bit depth. This is especially problematic for GPU-based volume rendering.

One could argue that post-interpolative classification is still possible with pre-computed LH values, since the stored LH values can be interpolated before actually applying the transfer function. However, we consider the LH value computation to be part of the classification process. Therefore, in this work the term "LH post-classification" denotes the computation of LH values from interpolated volume data.

In this chapter, we propose an efficient technique for the computation of LH values, which is fast enough to allow post-classification at interactive frame rates. Furthermore, we present an intuitive and efficient mechanism for specifying LH transfer functions that does not require any user interaction within the transfer function domain. We employ a sketching metaphor that allows the user to mark the features of interest in image space in order to assign the desired optical properties. Thus the user is able to sequentially identify the features of interest and to make them visible by the matter of a mouse click later on. This approach also enables less experienced users to interactively explore complex volumetric data sets without requiring training.

5.1 Efficient Construction of LH Histograms

In this section, we present an efficient way to compute a boundary voxel's LH values by only considering its intensity, gradient magnitude and second directional derivative along the gradient direction. In order to compute LH values without the expensive tracking of a path to the neighboring materials, it is necessary to make

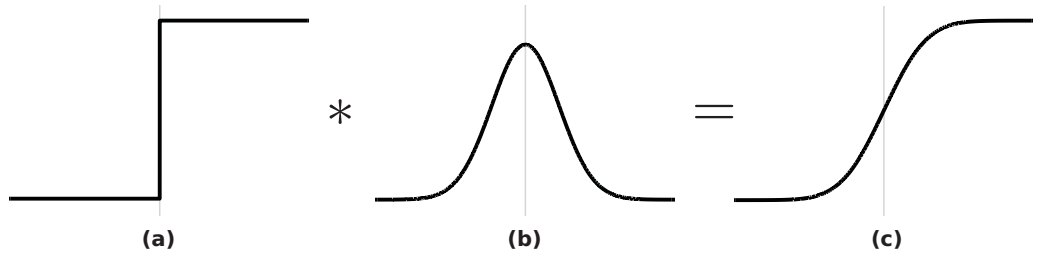


Figure 5.1: Assumed boundary model: ideal boundaries (c) in volume data sets are step functions (a) blurred by a Gaussian (b).

assumptions about the characteristics of boundaries in volume data sets. Since actual scanning devices are band-limited, they are unable to exactly reproduce point objects or sharp boundaries. In general, the band-limiting characteristics of an imaging system are described by its *point spread function (PSF)*, which specifies the system's impulse response to a point source. For the following considerations we employ the boundary model proposed by Kindlmann and Durkin (1998). They assume that real objects have sharp edges, i.e., discontinuous changes in the measured physical property, and model the PSF by a Gaussian function that is isotropic and constant over a data set. Although these assumptions might appear rather strong, the authors have shown that they match well the characteristics of CT and to some extent of MRI data sets. From a more intuitive point-of-view, one can think of boundaries as step functions that are blurred by a Gaussian as depicted in Figure 5.1.

5.1.1 Boundary Function

We start our derivation by considering a path that continuously follows the gradient direction through a boundary between the two materials with intensities F_L and F_H. Since the gradient vector at a position within a scalar field is always perpendicular to the isosurface through this point, this path also penetrates the boundary perpendicularly and thus constitutes the shortest path. In the following, we refer to this as the *boundary path*. For a mathematical description, we introduce the *boundary function* $f(x)$, which maps a position x along the boundary path to the intensity v at the respective sampling point. Figure 5.2 shows a slice of an artificial data set containing two regions with intensities F_L and F_H and a path cutting through their boundary.

According to the boundary model, $f(x)$ is a result of the convolution of a step function describing the physical boundary and a Gaussian function with the standard

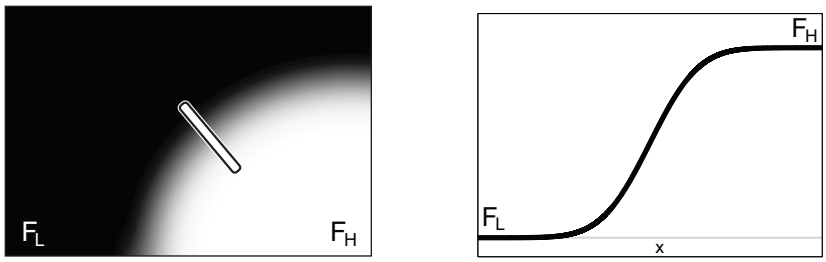


Figure 5.2: Slice with boundary path (a) and corresponding boundary function (b).

deviation σ. Therefore, the boundary function can be defined as:

$$f(x) = F_L + (F_H - F_L)\ \Phi\left(\frac{x}{\sigma}\right) \tag{5.1}$$

with $\Phi(x)$ being the cumulative distribution function (CDF) of the standard normal distribution. The center of the boundary is defined to be located at $x = 0$. Equation 5.1 tells us that each boundary features its own boundary function, which is parametrized by the intensities F_L and F_H of the neighboring materials as well as a parameter σ that specifies the amount of blurring that happened to the boundary. However, under the assumption that the boundary blurring is an attribute of the scanning device and is therefore uniform over a data set, the *blurring parameter* σ can be considered to be a constant. For the determination of σ as well as a derivation of the boundary function refer to Kindlmann and Durkin (1998). The first and second derivatives of $f(x)$ are as follows:

$$f'(x) = \frac{F_H - F_L}{\sigma\sqrt{2\pi}}\ e^{-\frac{x^2}{2\sigma^2}} \tag{5.2}$$

$$f''(x) = -x\ \frac{F_H - F_L}{\sigma^3\sqrt{2\pi}}\ e^{-\frac{x^2}{2\sigma^2}} \tag{5.3}$$

Apparently, $f'(x)$ has the form of a Gaussian function. Since the Gaussian function has inflection points at $x = \pm\sigma$, these are also the positions where $f''(x)$ attains its extrema.

Since the CDF of the standard normal distribution $\Phi(x)$ is bijective and thus invertible within its range, we can directly conclude from Equation 5.1 that the same property applies to $f(x)$. Therefore, we can define the inverse of the boundary function $f^{-1}(v)$, which maps an intensity $v \in]F_L, F_H[$ at a sampling point within the

boundary between two materials F_L and F_H to a position x along the corresponding boundary path:

$$\begin{aligned} & f(x) \;=\; v \\ \Leftrightarrow\quad & F_L + (F_H - F_L)\;\Phi\left(\frac{x}{\sigma}\right) \;=\; v \\ \Leftrightarrow\quad & x \;=\; \sigma\,\Phi^{-1}\left(\frac{v - F_L}{F_H - F_L}\right) \\ \Leftrightarrow\quad & f^{-1}(v) \;=\; \sigma\,\Phi^{-1}\left(\frac{v - F_L}{F_H - F_L}\right) \end{aligned} \tag{5.4}$$

$f^{-1}(v)$ enables us to express the boundary function's derivatives as functions of intensity instead of position:

$$g(v) := f'\left(f^{-1}(v)\right) = \frac{F_H - F_L}{\sigma\sqrt{2\pi}}\exp\left(\overline{\Phi}\right) \tag{5.5}$$

$$\begin{aligned} h(v) &:= f''\left(f^{-1}(v)\right) \\ &= -\frac{F_H - F_L}{\sigma^2\sqrt{2\pi}}\,\Phi^{-1}\left(\bar{v}\right)\exp\left(\overline{\Phi}\right) \end{aligned} \tag{5.6}$$

$$\text{with}\quad \bar{v} := \frac{v - F_L}{F_H - F_L} \quad\text{and}\quad \overline{\Phi} := -\frac{1}{2}\Phi^{-1}\left(\frac{v - F_L}{F_H - F_L}\right)^2$$

The functions $g(v)$ and $h(v)$ specify the first and second derivative of the boundary function at the position x with $f(x) = v$. Plots of them are shown in Figure 5.3. $g(v)$ can be used to map the LH space to the conventional intensity-gradient transfer function space. Though this mapping is not injective and therefore suffers from information loss, it allows a basic integration of the proposed classification technique into volume rendering systems without the need for a LH post-classification as described in Subsection 5.2.3.

5.1.2 Computation of LH Values

With these preparations, we now return to our goal of calculating the LH values at an arbitrary sampling point within a boundary. We consider the boundary path that runs through the sampling point. The position along this path that corresponds to the sampling point is labeled x_p. Furthermore, we assume the function values $f(x_p)$, $f'(x_p)$, and $f''(x_p)$ to be known. For clarity of presentation, these substitutes are used in the following derivation:

$$v := f(x_p) \qquad g := f'(x_p) \qquad h := f''(x_p)$$

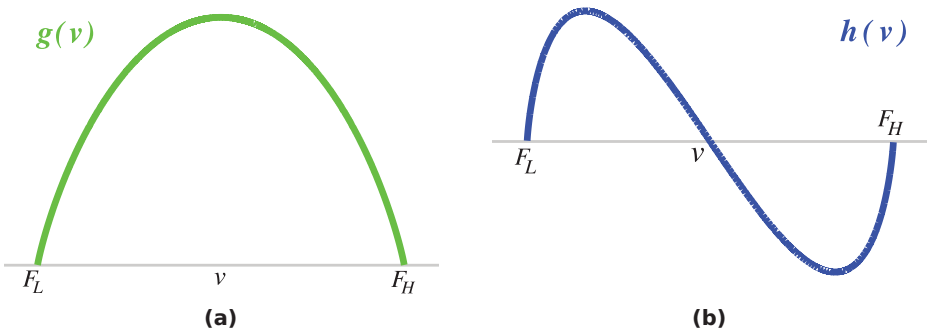


Figure 5.3: First (a) and second (b) derivative of the boundary function as functions of intensity.

First of all, we can recover x_p from the ratio of g and h. Dividing Equation 5.2 by Equation 5.3 yields:

$$\frac{g}{h} = -\frac{\sigma^2}{x_p} \quad \Leftrightarrow \quad x_p = -\sigma^2 \frac{h}{g} \tag{5.7}$$

By additionally considering Equation 5.1 it is now possible to determine F_L:

$$\begin{aligned}
&\textbf{I (Eq. 5.1)} && v = F_L + (F_H - F_L)\,\Phi\left(\frac{x_p}{\sigma}\right) \\
&\textbf{II (Eq. 5.2)} && g = \frac{F_H - F_L}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x_p^2}{2\sigma^2}\right) \\
&\Leftrightarrow && F_H = \frac{\sigma\sqrt{2\pi}\,g}{\exp\left(-\frac{x_p^2}{2\sigma^2}\right)} + F_L \\
&\textbf{II in I} && v = F_L + \frac{\sigma\sqrt{2\pi}\,g}{\exp\left(-\frac{x_p^2}{2\sigma^2}\right)}\,\Phi\left(\frac{x_p}{\sigma}\right) \\
&\Leftrightarrow && F_L = v - \frac{\sigma\sqrt{2\pi}\,g}{\exp\left(-\frac{x_p^2}{2\sigma^2}\right)}\,\Phi\left(\frac{x_p}{\sigma}\right) \\
&\underset{(5.7)}{\Rightarrow} && F_L = v - \frac{\sigma\sqrt{2\pi}\,g}{\exp\left(-\frac{h^2}{g^2}\frac{\sigma^2}{2}\right)}\,\Phi\left(-\sigma\frac{h}{g}\right)
\end{aligned} \tag{5.8}$$

F_H can be derived by inserting F_L and x_p into Equation 5.1:

$$F_H = \frac{v - F_L}{\Phi\left(-\sigma \frac{h}{g}\right)} + F_L \tag{5.9}$$

The preceding analysis is based on the assumption that the boundary function $f(x)$ and its derivatives are known at all positions within boundaries. This is true for $f(x)$ itself, as it simply equals the intensity value at a sampling point; the determination of the boundary function's derivatives, however, needs further consideration. Recalling that the boundary path is defined to continuously follow the gradient direction, $f'(x)$ and $f''(x)$ turn out to be the data set's first and second directional derivatives along the gradient direction. According to vector calculus (Marsden and Tromba, 1996), the directional derivative $\mathbf{D}_{\vec{v}}$ along the direction $\vec{v}$ of a scalar field $s(\vec{r})$ is the scalar product of the gradient of s and the vector $\vec{v}$ in normalized form:

$$\mathbf{D}_{\vec{v}}\, s = \nabla s \cdot \frac{\vec{v}}{||\vec{v}||}$$

Hence, the first derivative along the gradient direction is just the gradient magnitude:

$$\mathbf{D}_{\nabla s}\, s = \nabla s \cdot \frac{\nabla s}{||\nabla s||} = ||\nabla s||$$

In a similar way, we obtain the second directional derivative along the gradient direction:

$$\begin{aligned} \mathbf{D}^2_{\nabla s}\, s &= \mathbf{D}_{\nabla s}\left(\mathbf{D}_{\nabla s}\, s\right) = \mathbf{D}_{\nabla s}\left(||\nabla s||\right) \\ &= \frac{\nabla\left(||\nabla s||\right) \cdot \nabla s}{||\nabla s||} \end{aligned}$$

5.1.3 Comparison with Šereda's Method

In order to compare the proposed technique to the original one in terms of speed and accuracy, we applied it to a selection of data sets used by Šereda et al. (2006a).

Computation Time

Table 5.1 shows performance comparisons for CT scans of a human tooth and a human hand. The measurements were conducted on an Intel Core 2 Duo 2.2 GHZ machine with an NVIDIA GeForce 8800 GTX graphics board. We determined the

		LH Hist. Construction on CPU		Rendering Speed (FPS)	
data set	size	Šereda et al.	our technique	2D TF	LH TF
tooth	256^2 x 161	1:12 min	7.2 / 15.3 s	35.9	26.9 / 20.5
hand	256^2 x 232	1:36 min	10.7 / 19.4 s	32.2	24.5 / 17.3

Table 5.1: Performance of the LH classification of two CT data sets. Columns 3 and 4 compare the construction time of LH histograms on the CPU by the original technique and by ours. For our technique, the first figure indicates the performance with pre-computed derivatives, the second without. Columns 5 and 6 specify frame rates for the LH post-classification with and without pre-computed derivatives compared to the application of a conventional 2D transfer function with a viewport size of 512^2.

construction time of an LH histogram of the entire data set on the CPU with a single-threaded implementation as well as the rendering speed when performing an LH post-classification in combination with Phong shading in a GPU-based raycaster. Both measurements were performed with and without pre-computed gradient magnitudes and second derivatives. Šereda et al. used pre-computed derivatives for their benchmark.

For the single-threaded LH histogram generation on the CPU we achieved a significant speed up of about one order of magnitude with pre-calculated derivatives and still a speed up of about factor five when generating the derivatives on the fly. Since the LH values can be computed independently for each voxel position, a nearly linear speedup can be expected for a multi-threaded implementation. It should be considered, however, that Šereda et al. did not specify their hardware configuration, which hampers the comparability of the results.

As the authors admitted, the original LH technique is not fast enough to allow post-classification, and therefore they had to store the LH classification in an additional pre-computed volume. In contrast, our method allows LH post-classification at interactive frame rates, as can be seen from Table 5.1. For an estimation of the additional effort caused by the LH value calculation in the shader we also determined frame rates for a classification with a conventional intensity-gradient based transfer function, which was set up to produce a rendering result resembling the one achieved with the LH classification as closely as possible: with pre-computed derivatives we noticed a frame rate drop of only about 30 %, while an on-the-fly calculation of the derivatives slows down the rendering by about 40 % compared to the 2D transfer function.

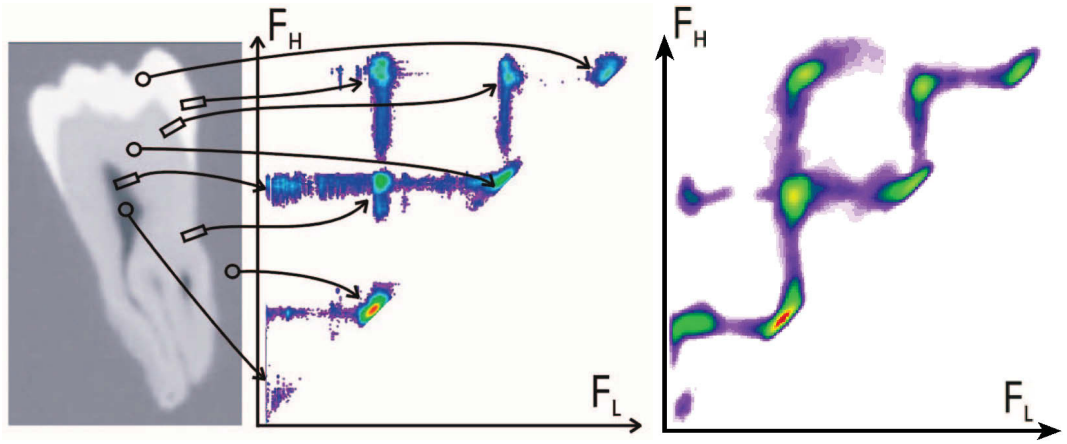


Figure 5.4: LH histograms of the tooth data set generated with the original method (a) (courtesy of Šereda et al.) and with our technique (b). The contribution is shown in logarithmic scale: red is highest, magenta is lowest.

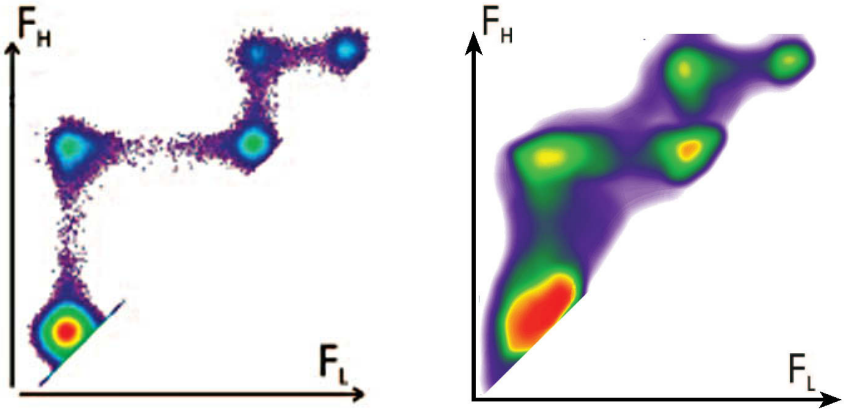


Figure 5.5: LH histograms of an artificial data set of two noisy spheres computed with Šereda's (a) and our technique (b) (logarithmic scale).

Accuracy

The ability to calculate sufficiently accurate LH values is crucial for the proposed technique. Figure 5.4 compares an LH histogram of the tooth data set computed by our technique to the result presented by Šereda et al. (2006a). Although our LH histogram appears to be slightly more blurry and especially the horizontal and vertical bars between the boundary blobs are more pronounced, it clearly exhibits the same structure as the original LH histogram. Note that not only the blobs representing boundaries are existent, but also the blobs on the diagonal of the histogram that represent homogeneous regions.

In order to investigate to what extent our LH calculation is prone to noise, we generated an artificial data set of two spheres blurred by a Gaussian and added 0.1 percent of Gaussian noise to it, similar to the data set used by Šereda et al. The LH histogram we derived from the volume (see Figure 5.5) still shows the expected blobs, but also exhibits a broader distribution of misclassified voxels around these blobs than in the original histogram. As this seems to be a consequence of the noise sensitivity of the standard derivative estimators, we believe that an application of more advanced derivative reconstruction schemes could significantly improve the noise robustness of our method.

5.2 Volume Exploration

In this section, we describe a semi-automatic volume exploration system that exploits the LH technique in order to enable the user to interactively classify features of interest. In an iterative process the user extracts boundaries by directly marking them in the volume rendered image. The system then detects LH values of the respective boundaries and generates a suitable LH transfer function incorporating user-specified optical properties.

5.2.1 User Interaction

In our system, the user is expected to mark a desired feature of interest by drawing a free-form patch onto the region covered by the respective feature in image space. It is neither necessary to precisely sketch a feature's silhouette nor has the user to mark the whole object. Furthermore, during the exploration no interaction within the transfer function domain is required, rather the user can apply the commonly used windowing approach to change the visibility of parts of the data set.

In order to offer the user the possibility to assign optical properties to features

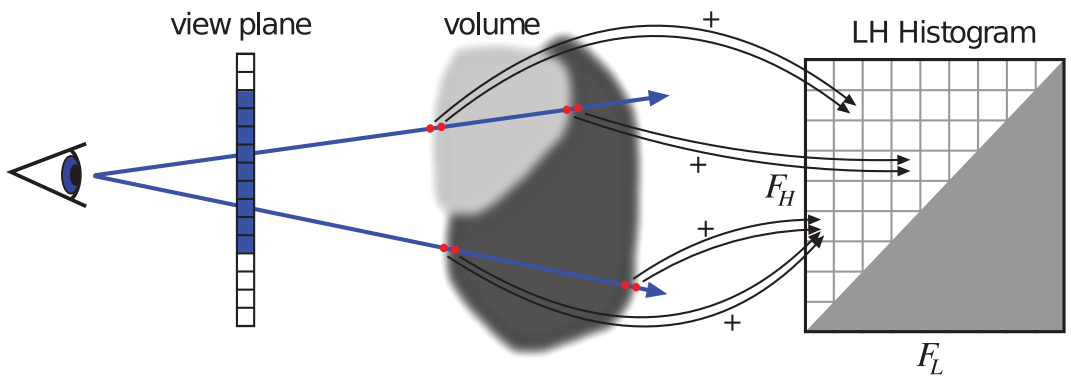


Figure 5.6: Generation of the LH histogram. A ray is cast through each view plane pixel covered by the user-drawn patch (blue). Each visible boundary voxel hit by one of these rays is mapped to the histogram bin (F_L, F_H) representing its boundary.

independently, we use a layer interface similar in spirit to the one presented by Ropinski et al. (2008). Each extracted boundary is represented by a layer through which the user can specify the boundary's color and opacity.

5.2.2 Boundary Extraction

After the user has drawn a patch, the system analyzes the subvolume determined by the extrusion of the patch in the viewing direction in order to detect all visible boundaries contained within this subvolume. The boundary detection is performed by an analysis of the LH histogram of this subvolume. We create the LH histogram by casting a ray through each pixel of the user-drawn patch in the view plane. At each visible boundary voxel that is hit by one of these rays, the F_L and F_H values are calculated according to Equations 5.8 and 5.9, and the corresponding histogram cell with coordinates (F_L, F_H) is incremented. Figure 5.6 depicts this proceeding.

As we are interested in the detection of features that have been marked by the user, it is necessary to consider only those voxels for the construction of the LH histogram that are visible in the current rendering. For that purpose, we weight each voxel's contribution by its opacity, which is determined by the current windowing. In order to prevent opaque but completely occluded voxels from contributing to the histogram, we stop the traversal of a ray when it is saturated, i.e., when the ray has reached an alpha value of 1.0 during the front-to-back compositing.

Since we aim at the extraction of boundaries, it is necessary to consider only boundary voxels for the LH histogram. The straightforward approach to incorporate only voxels with a gradient magnitude above a certain threshold, however, has some

disadvantages:

- The gradient magnitude is boundary-dependent, i.e., the gradient magnitude distribution within a boundary is proportional to the intensity range that is spanned by the boundary. This makes it difficult to define an adequate threshold for the entire data set.

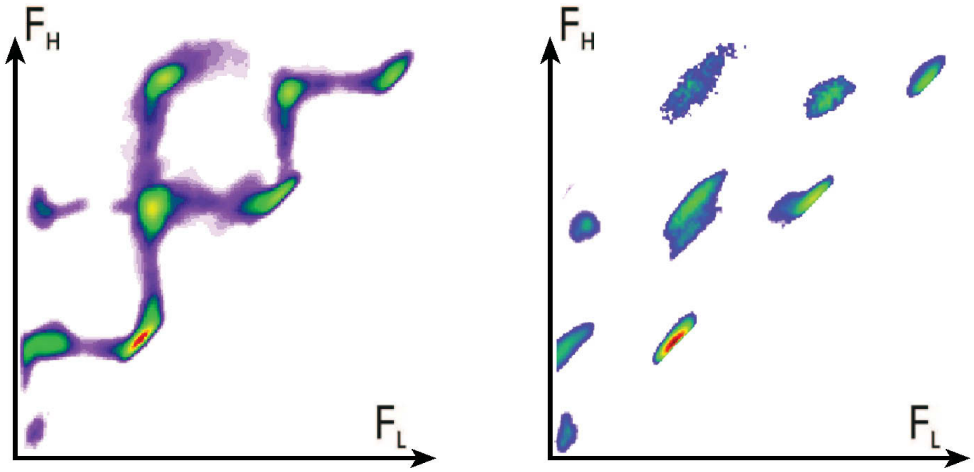
- Noisy regions exhibit a significantly high gradient magnitude and may thus accidentally contribute to the histogram.

Instead, we use a voxel's position x_p along its boundary path (see Equation 5.7) for weighting its contribution. The weighting factor w is given by:

$$w := \exp\left(-x_p^2\right) = \exp\left(-\sigma^4 \frac{h^2}{g^2}\right) \tag{5.10}$$

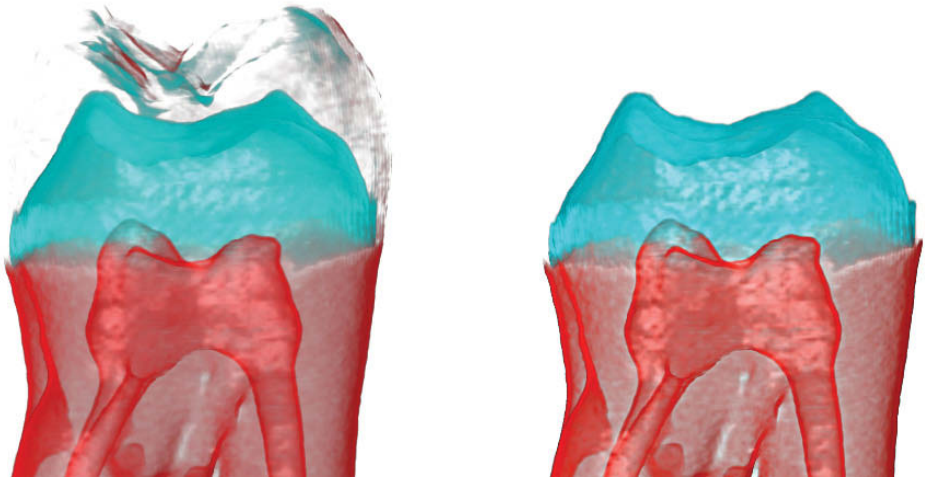
As we defined voxels with $x_p = 0$ to be located at the center of a boundary, the contribution of these center voxels is maximal while an increasing distance to the center reduces the influence on the LH histogram. Furthermore, this weighting diminishes the influence of noisy regions, since such regions usually exhibit high second order derivatives in relation to the gradient magnitude. Figure 5.7b demonstrates the effect of the distance weighting for the tooth data set: The boundary as well as the material blobs are significantly more localized and the connecting bars almost vanish. We also noticed a substantial reduction of rendering artifacts when applying the distance weighting. Figure 5.8 illustrates this for the tooth data set. Both renderings have been generated with the same LH transfer function consisting of two blobs that are located at the respective coordinates of the dentin-background (red) and dentin-enamel (blue) boundaries in LH space. Without distance weighting, there are several misclassified voxels at the enamel-background boundary visible, whereas distance weighting eliminates these artifacts without perceptibly effecting the boundaries themselves. Apparently, the LH classification is most reliable at the center of a boundary. Therefore, we use the distance weighting not only for the boundary extraction but also as part of the classification.

After the construction of the LH histogram, the detection of boundaries is now reduced to a local maxima search in the histogram space, as each blob in the LH histogram represents a boundary within the analyzed subvolume. In order to cope with noise and discretization artifacts, we apply a slight blurring before the maxima detection, e.g. by a 3x3 Gaussian kernel for an 8 bit histogram.



(a) no distance weighting **(b)** distance weighting applied

Figure 5.7: LH histograms of the tooth data set generated by our technique with and without distance weighting (logarithmic scale).



(a) no distance weighting **(b)** distance weighting applied

Figure 5.8: LH classified renderings of the tooth data set obtained using the same transfer function with and without distance weighting.

5.2.3 Transfer Function Generation

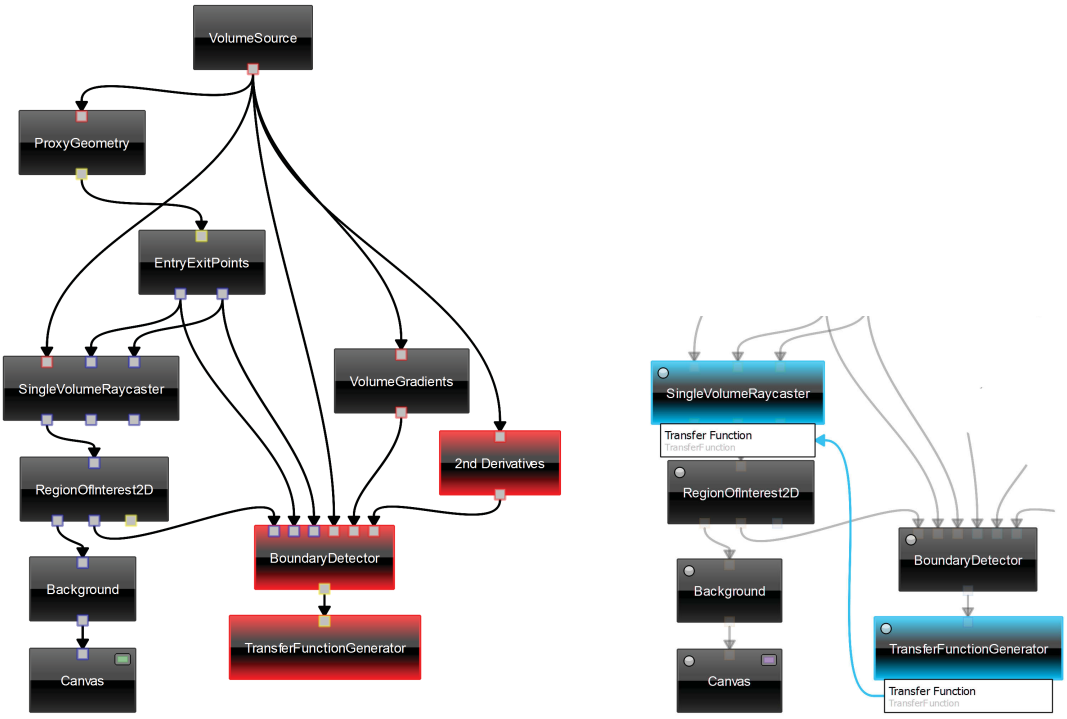
The feature extraction yields a list of tuples (F_L, F_H) representing the extracted boundaries. This boundary information can be used in a straightforward way to generate a suitable LH transfer function for the visualization of these boundaries. In our setup, each boundary is represented by a single layer in the user interface and associated with an LH component function that contains a Gaussian bell curve centered around the boundary's (F_L, F_H) coordinates. Besides a boundary's optical properties, the user can control its bell curve's variance. We call this parameter "fuzziness" as it specifies the size of the bell curve in LH space and therefore determines to what extent voxels with slightly deviating LH values are incorporated by the component function. In order to produce an LH transfer function that can actually be used during the rendering process, the boundaries' component functions are combined by calculating their weighted average based on opacity. This yields a transfer function containing the LH blobs of all classified boundaries.

We want to stress that the proposed volume exploration technique does not necessarily require an LH classification during the rendering process. Instead, the extracted boundary information can be used to generate a 2D transfer function based on the intensity and gradient magnitude, which is nowadays widely used by volume rendering systems. The region that represents a boundary in this transfer function space can be determined by evaluating Equation 5.5, which provides us with the gradient magnitude as a function of intensity. This mapping, however, is not injective since distinct LH coordinates may be mapped to intersecting arcs in the intensity-gradient space.

5.3 Integration into Voreen

Integrating the proposed LH classification approach into the Voreen framework enabled us to focus on implementing the boundary detection and transfer function generation, while using standard components provided by the framework for rendering and interaction. Figure 5.9a shows the Voreen data-flow network that implements the volume exploration system described in Subsection 5.2. The processors highlighted in red have been specifically developed for this system, the remaining ones are standard components.

There are three central tasks the processors can be attributed to: rendering, user interaction, and boundary detection. The volume rendering is performed by the sub-network on the left, which resembles the standard raycasting pipeline presented in Chapter 3.3. The only difference lies in the `RegionOfInterest2D` processor in-



(a) data-flow network **(b)** property linking layer

Figure 5.9: (a) shows the data-flow network implementing the proposed LH classification technique in Voreen. The standard raycasting pipeline on the left is complemented with custom processors (highlighted) performing the boundary detection and LH transfer function generation. (b) is a cutout of the network's linking layer containing a property link between the transfer function generator and the raycaster.

serted between the `SingleVolumeRaycaster` and the `Background` processor. It allows the user to draw a free-form patch over the volume rendered image in order to mark the feature whose boundaries are to be detected. The `RegionOfInterest2D` processor's first (left-most) outport passes the rendered image overlayed with the painted region of interest to the `Background` processor, while its second outport puts out a mask image representing the region of interest, which in turn is used by the `BoundaryDetector`.

5.3.1 Volume Raycasting with LH Classification

Since the Voreen framework had no built-in support for LH transfer functions, we needed to extend the `SingleVolumeRaycaster` to support this transfer function space.

For the calculation of LH values by the Equations 5.8 and 5.9 the gradient magnitude g and the second derivative h have to be known at each sampling point. We compute approximations of these quantities by central differences and the discrete Laplacian operator on-the-fly in the raycasting shader. Since the CDF of the standard normal distribution Φ is not available on the GPU, we sample it into a one-dimensional texture that can be accessed by the shader. Due to the low-frequency character of Φ a resolution between 32 and 64 samples seems to be sufficient. The LH transfer function table itself is stored in a two-dimensional OpenGL texture.

5.3.2 Boundary Detection and Transfer Function Generation

The `BoundaryDetector` computes the LH histogram of the subvolume determined by the extrusion of the user-drawn patch in viewing direction, as described in Subsection 5.2.2. Its image inports (blue) receive a mask image of the user-drawn patch as well as the ray parameter textures encoding the ray start and end points of the current frame (compare Chapter 2.3). Applying the mask image to the ray parameter textures then yields the entry and exit points of the rays that need to be cast through the subvolume of interest. The `BoundaryDetector`'s volume inports (red) receive the volume's scalar data and its first and second derivatives along the gradient direction, which are used for evaluating the LH computation functions 5.8 and 5.9. Information about the detected boundaries is then submitted via a geometry port to the `TransferFunctionGenerator`, which creates the LH transfer function as described in Subsection 5.2.3. In order to apply the generated transfer function during the volume rendering, it needs to be passed back to the `SingleVolumeRaycaster`. This is done by a property link that synchronizes the transfer functions of the two processors as depicted in Figure 5.9b.

5.4 Results

We applied the proposed volume exploration technique to three medical CT scans as well as an electron microscopy scan. All renderings have been generated by interaction with the described user interface only, without any direct manipulation of the LH transfer function.

Figure 5.10 shows the classification of a renal angio CT scan. We extracted five features: the bone, the kidneys, the blood vessels (semi-transparent), the skin, and a synthetic stick. The exploration of this data set took about five minutes, including the assignment of optical properties. In Figure 5.11 an electron microscopy scan is explored. We managed to classify four features in this data set with relatively little effort, which indicates that the assumed boundary model fits this type of data.

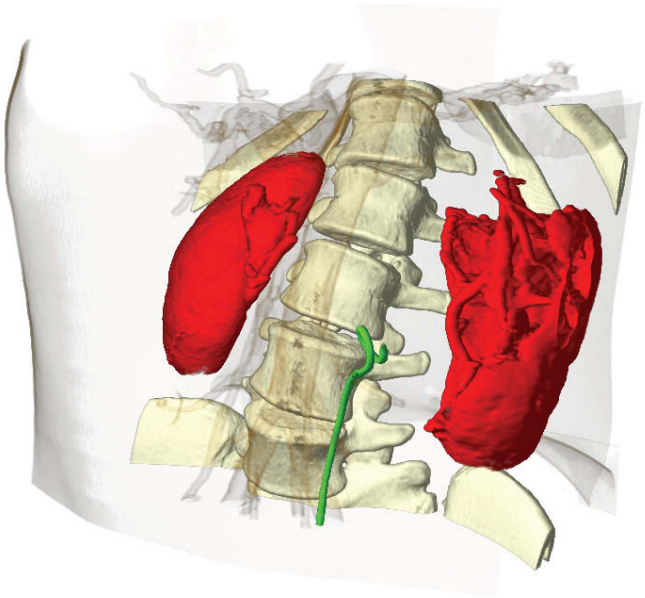


Figure 5.10: Application of the proposed technique to a renal angio CT scan. We extracted the bone, the kidneys, the blood vessels, the skin, and a synthetic stick (green).

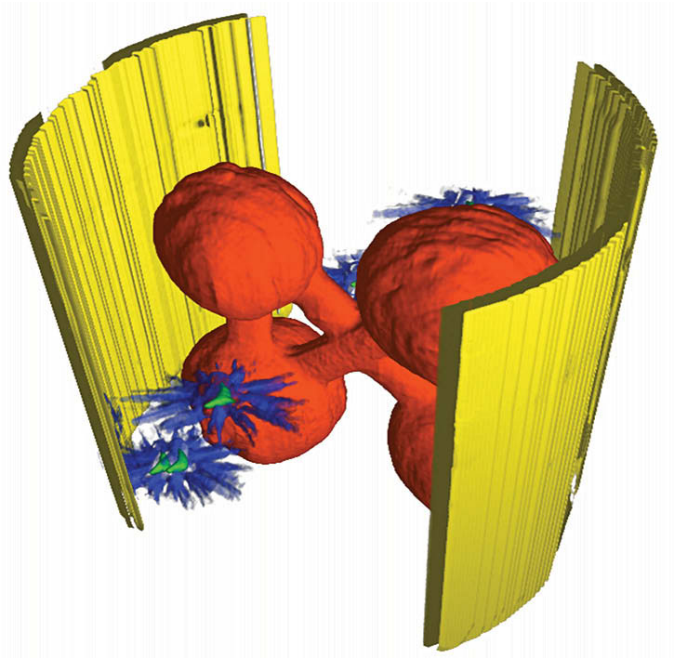


Figure 5.11: Electron microscopy scan with four extracted features.

Figure 5.12 shows the result of the application of the proposed technique to the tooth data set. Figure 5.13 presents a classification of a hand CT scan. Besides the tissue-background boundary, which appears as the skin in the rendering, we extracted the bone-tissue boundary as well as the bone-marrow boundary. The extraction process took just about one minute. However, we were not able to separate the blood vessels from the bone, as they share a common footprint in the LH space.

Figure 5.12: Rendering of the tooth data set (CT). The dentin-background (red), the dentin-enamel (yellow), and the enamel-background (blue) boundaries have been classified.

5.5 Summary

In this chapter, we have presented an efficient method for the calculation of LH values that does not require an expensive tracking of boundary intensity profiles, but exclusively uses local measures. We have shown that it allows post-classification at interactive frame rates whereby the pre-computation of an LH volume is not necessary. By comparing our results to the work of Šereda et al. (2006a), we have demonstrated that the proposed technique is sufficiently robust when applied to CT data. Also due to the relatively easy implementation, we believe that our novel LH classification has the potential to boost the use of the LH space as transfer function domain.

Moreover, we have proposed a system for the semi-automatic design of LH transfer functions, which completely shields the user from the transfer function domain and allows him/her to extract features of interest by direct interaction with the rendered volume and to conveniently assign optical properties to these features. Furthermore, we have pointed out a possibility to exploit our system for the generation of conventional 2D transfer functions based on the intensity and gradient magnitude, which eases the integration into existing volume rendering systems.

In the future, we would like to improve the applicability of our LH classification to MRI data. We believe that this could be achieved by exploiting more advanced derivative reconstruction schemes or an adaptation of the boundary model.

Figure 5.13: Classification of the hand data set (CT). The tissue-background boundary (skin), the bone-tissue (blue), and the bone-marrow boundaries (red) are visible.

Chapter 6

Shape-based Transfer Functions

We propose a volume classification technique that takes the shape of volumetric features into account. The presented technique enables the user to distinguish features based on their 3D shape and to assign individual optical properties to these. Based on a rough pre-segmentation, which can be done by windowing, we exploit the curve-skeleton of each volumetric structure in order to derive a shape descriptor similar to those used in current shape recognition algorithms. The shape descriptor distinguishes three main shape classes: longitudinal, surface-like, and blobby shapes. In contrast to previous approaches, the classification is not performed on a per-voxel level, but assigns a uniform shape descriptor to each feature and therefore allows for a more intuitive user interface for the assignment of optical properties. By using the proposed technique, it becomes for instance possible to distinguish blobby heart structures filled with contrast agents from potentially occluding vessels and rib bones. After introducing the basic concepts, we show how the presented technique performs on real world data, and we discuss current limitations.

A major drawback of conventional classification techniques, which consider only the data value and its derivatives, is their inability to distinguish features that belong to the same material class. For example, intensity-based transfer functions do not allow the user to discriminate between different kinds of bones or to separate blood vessels from heart tissue in a contrast-enhanced CT scan. Such tasks usually require an elaborate segmentation of the data set.

In this chapter we present an addition to the transfer function concept along with a corresponding user interface. By applying methods already used in shape classification, we are able to define a multidimensional transfer function that takes into account the shape of a desired feature for assigning optical properties. We perform a rough, threshold-based pre-segmentation in a pre-processing step and compute the

curve-skeletons for each of the resulting volumetric structures. Since curve-skeletons are well known to sufficiently describe the shape properties of 3D objects (Macrini et al., 2008), we are able to derive some shape metrics for the objects, specifying the degree of membership in some predefined shape classes. These shape metrics are given by a shape descriptor, i. e., a triple (*tubiness*, *surfaceness*, *blobbiness*). Thus the user is able to distinguish features that are similar in terms of intensity and gradient magnitude, but do have different shapes. In order to avoid the need for intensive user interaction during the pre-processing step, we do not rely on high-quality manual or semi-automatic segmentations, but focus on those that can be achieved by simple windowing. As a consequence, a major challenge in the classification process for real-world data is to handle imperfections in the pre-segmentation, which may result in instabilities in the shape-skeletons and thereby disturbing shape classification. Hence, we use data pre-processing and robust classifiers to handle these issues, which would not be necessary for voxelized geometry data often used for testing skeletonization algorithms.

To make our concept easily accessible, we also propose a corresponding user interface for specifying shape-based transfer functions, which is based on a continuous triangle-shaped plot showing the occurrence of the detected shape classes. The user can directly assign optical properties to these shape classes. Benefits of the presented approach are that it allows the user to visually separate objects that previously could only be separated by applying complex segmentation techniques, therefore requiring much less manual intervention. Furthermore, the notion of shape is very intuitive and thus the concept can also be applied without profound knowledge of the underlying algorithms, e. g., for use by domain experts.

6.1 Related Work

In addition to the general overview of volume classification techniques given in Chapter 4, we briefly review here related work in the fields of shape-based classification and volumetric skeletonization.

Our approach is related to the one described by Sato et al. (2000). They also support a volume classification based on shapes. However, in contrast to our technique they detect shapes, such as linear structures and blobs, by measuring multi-scale responses to 3D filters. Thus, their classification is performed on a per-voxel basis, whereas our approach classifies complete volumetric features, thereby allowing more intuitive user interfaces and a better understanding by domain experts. Jiawan et al. (2003) use inertia tensors for the classification of shapes in volumetric data sets. Their approach, however, is limited to classifying structures of a pre-defined size.

Shape Classification based on skeleton structures has a long history. In 3D, Binford's generalized cylinders (Binford, 1971) decompose an object into a set of elongated parts defined by sweeping a 2D cross-section along a 3D space curve. The concept of an axial description of shape was proposed even earlier in 2D through Blum's medial axis transform, or skeleton (Blum, 1973). Pizer et al. (2003) proposed a framework of stable medial representation for segmentation of objects, registration, and statistical 3D shape analysis.

Cornea et al. (2005, 2007) examined existing algorithms and introduced the concept of the hierarchical curve-skeleton as a robust method to compute increasingly detailed skeletons. Their algorithm uses a repulsive force field to extract curve-skeletons of general 3D objects from volumetric data sets, using topological characteristics such as critical points found in the resulting vector field. As the potential field is generated by charging the object's boundary, the algorithm only requires information about the object's surface voxels.

Skeletons were previously used for shape-matching, e. g., by Hilaga et al. (2001), but also for volume visualization. Takahashi et al. (2004) automate transfer function design by extracting the topological structure of a volume data set using a skeletonization process. Reniers et al. (2008) classify voxel surfaces using a 3D skeletonization method. Correa and Silver (2005) use skeletons to manipulate transfer functions while they are moving along features. Cornea et al. (2005) list a multitude of further uses for curve-skeletons.

6.2 Classifying Shapes in Volume Data

The goal of our approach is to construct a fuzzy shape-classification. In contrast to shape-matching techniques which *compare* shapes, this classification results in several membership scores that specify how much the object matches each of the predefined shape classes. In order to allow the design of an intuitive user interface for shape classification especially for non-visualization experts, we decided to avoid the per-voxel classification performed by previous approaches (Sato et al., 2000; Jiawan et al., 2003), since these require the user to analyze multidimensional histograms of similar complexity as the intensity-gradient transfer function space. Instead, we aimed at presenting a manageable set of shape-classified structures to the user for the assignment of optical properties. Therefore, we have chosen a statistically motivated approach, based on curve-skeletons, that classifies structures obtained from a decomposition of the input volume. These curve-skeletons have several benefits for shape-classification: they are invariant to translation, rotation, and scaling, and they can cope with moderate amounts of within-class deformation.

From analyzing typical volume data sets and their corresponding shape-skeletons, we have derived three independent shape classes: longitudinal/tubular, surface-like, and blobby shapes. The goal of the classification process is to assign to each voxel in the data set a score for each shape class, specifying how much the object part corresponding to the voxel matches the shape. No binary decisions are made, because volume data may contain many ambiguous shapes and the task of mapping the results of shape classification to optical properties is better left to the user in an interactive process. The overall workflow of our approach is a follows (see Figure 6.1):

1. Creation of a *pre-segmentation* to obtain surface information about the objects of interest. In order to minimize the amount of user interaction that is necessary during the pre-processing step, we focus on segmentations that can be achieved by assigning intensity thresholds (windowing). In case the intensity range of the desired structures is known in advance, e.g. Hounsfield scale for CT scans, the pre-segmentation can be obtained automatically.

2. Computation of the curve-skeleton, resulting in several polygonal chains of skeleton points, the *curve-skeleton segments*. Additionally, we perform a normalization of the skeleton for reducing artifacts often caused by imperfect segmentations.

3. Decomposition of the pre-segmentation based on the curve-skeleton segments: The volume is further divided into *skeleton regions*, with each region corresponding to a segment of the curve-skeleton.

4. Merging of skeleton regions. Neither the computation of the curve-skeleton nor the skeleton normalization take into account the membership of structures to the shape classes. Therefore, the skeleton regions resulting from the ad-hoc decomposition of the pre-segmentation cannot be expected to have a meaningful shape. We merge skeleton regions based on certain criteria in order to improve their classifiability.

5. Shape analysis, incorporating both curve-skeleton and volume data to assign shape scores to the merged skeleton regions, thereby constructing the shape descriptors.

In the following, we first describe the shape classification scheme and then in more detail the individual steps necessary to generate the shape classification.

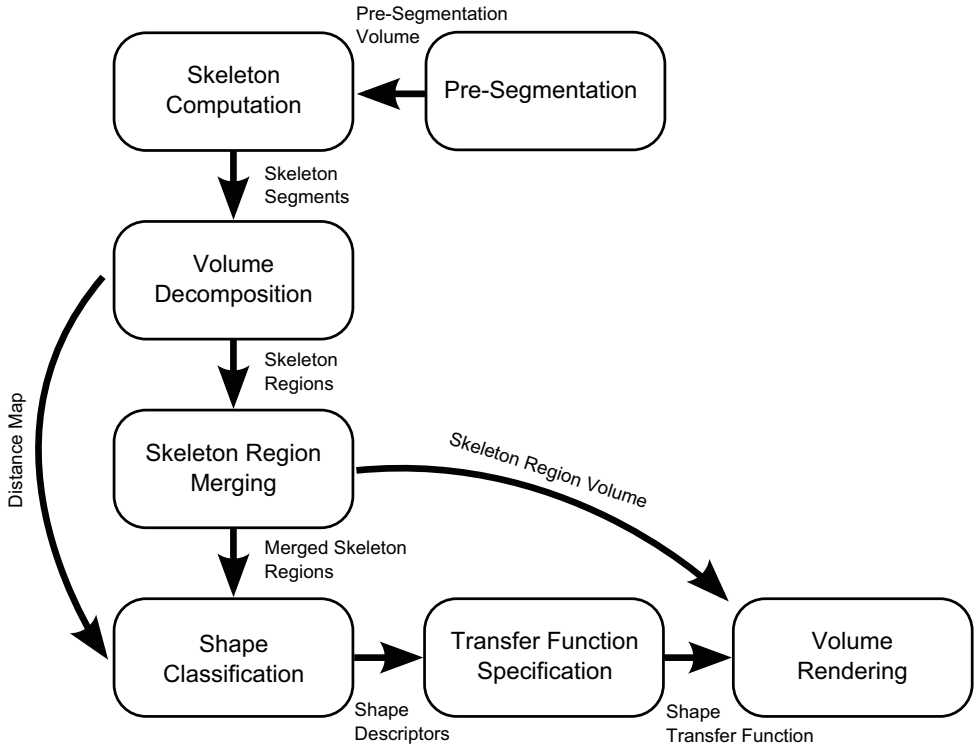


Figure 6.1: Workflow of our shape classification approach.

6.2.1 Shape Classification

The goal of the shape classification is to assign to each skeleton region generated by the previous volume decomposition a degree of membership in each of the supported shape classes. Only voxels specified in the pre-segmentation are considered, while all other voxels are considered as background voxels and ignored, setting their shape descriptor to all zero. Shape class membership is computed independently for each class, therefore further classes can be added easily. Our system supports three shape classes, namely tubiness, surfaceness, and blobbiness. In a medical context tubiness would be associated with blood vessels or elongated bones, blobbiness would be found with organs such as the heart or the kidneys. Structures with a high surfaceness could include the skullcap or a blade-bone. The three shape classes can also be interpreted as a measure of dimensionality, i. e., a shape with a one-dimensional elongation gets a larger tubiness value, whereas a more three-dimensional shape gets a larger blobbiness value.

Tubiness

We define an elongated region having a circular cross-section of constant diameter as a perfectly tubular structure. In theory, such structures feature a curve-skeleton that runs through the center of each cross-section, and each of the region's surface points has the same minimal distance to the curve-skeleton: the radius of the tube. Therefore, we define the tubiness τ of a volumetric region R as the inverse of the standard deviation of its surface voxels' minimal skeleton distances, i. e.,

$$\tau(R) := \frac{1}{\max\left(\sqrt{\frac{1}{|\partial R|}\sum_{\vec{v}\in\partial R}\left(D(\vec{v})-\bar{d}\right)^2},\ 1\right)} \in [0;1]\,, \tag{6.1}$$

where ∂R is the set of surface voxels of R, D is the distance map computed during the skeleton region growing, and $\bar{d}$ is the mean skeleton distance of all $v \in \partial R$. In contrast, both surfaces and blobby regions usually feature a significantly higher standard deviation, as Figure 6.2 illustrates. Note that the tubiness classifier is not restricted to cylindric shapes but yields similar classification results for curved longitudinal shapes such as rib bones.
The reliability of the tubiness classifier depends to some degree on the quality of the computed curve-skeletons, since especially an incomplete, discontinuous skeleton segment boosts the standard deviation of a tubinal region's surface-to-skeleton distances. By connecting adjacent skeleton segments during the skeleton normalization we were able to handle this issue in most cases. Only highly degenerated tubular regions, where only a small fraction is represented correctly by the skeleton, may be misclassified.

Surfaceness

A surface can be either planar or folded in space. In the first case, the object's minimal oriented bounding box has a very small extent in one space direction compared to the extent in the two other space directions. We compute the minimal oriented bounding box for each region by applying Har-Peled's technique (Har-Peled, 2001). The planarity of a region is then proportional to the ratio of the bounding box's second shortest side length b_m and its shortest side length b_s. According to our notion of planarity, we define regions with $b_m \geq 10 \cdot b_s$ to be maximal flat and regions with $b_m \leq 5 \cdot b_s$ to be not flat at all and apply a linear transition between these extrema:

$$plan(R) := \text{clamp}\left(\frac{1}{5}\frac{b_m(R)}{b_s(R)} - 1,\ 0,\ 1\right) \in [0;1] \tag{6.2}$$

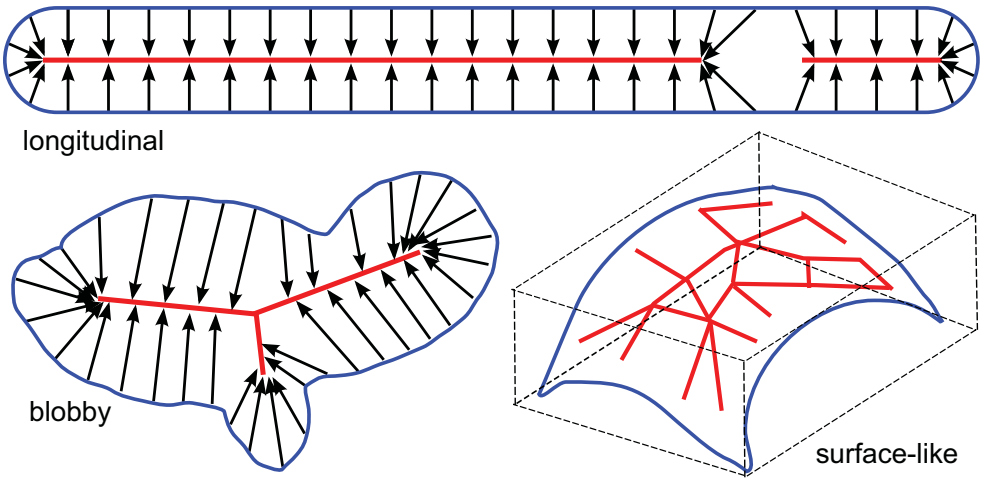


Figure 6.2: Typical cases for the three shape classes. The arrows illustrate the distance of each surface voxel (blue) to the skeleton segment (red) . For the longitudinal shape a gap in the skeleton is displayed. The surface shape is shown along with its bounding box.

In case of a curved surface, the bounding box criterion does not work. Therefore, we additionally compute the *convexity* of each shape by selecting an arbitrary set of pairs of surface points and determining for each line segment formed by a point pair the fraction that runs inside the shape. The convexity $conv(R)$ is then the average of these fractions. Since curved surfaces occupy only a small part of their bounding box's volume as depicted in Figure 6.2, most parts of the line segments between surface points are lying outside the region. Note that the sum of the planarity measure and the convexity measure in isolation is not an appropriate surfaceness classifier, since curved longitudinal shapes might have both planar bounding boxes as well as a low convexity. Therefore, their normalized sum has to be weighted by the inverse tubiness score:

$$surf(R) := (1 - \tau(R)) \frac{plan(R) + conv(R)}{2} \in [0;1] \tag{6.3}$$

Blobbiness

The goal of the blobbiness classifier is to detect volumetric regions that humans intuitively regard as "compact". Though it is hard to give a precise definition of blobbiness, one can certainly say that a sphere or a cube are blobby objects, whereas tubes or planar structures are less so. Furthermore, a suitable blobbiness classifier

should classify an ellipsoid or a cuboid less blobby than a sphere or a cube. One possibility could be the examination of the ratio of surface to volume of a given region, since it can be shown that a sphere has minimum surface for a given volume and that a cube has a smaller surface than any non-cubic cuboid of the same volume. But although the surface-to-volume ratio seems to be an appropriate indicator for the blobbiness of *analytical* structures, it is less useful for the classification of *volumetric* data sets, which often contain regions with rough, imperfect boundaries. These jagged regions exhibit a larger surface (i. e., a greater number of surface voxels) than an analytical object of the same global shape, resulting in a misleading surface-to-volume ratio.

Therefore, we follow a statistically motivated approach focusing on the spatial distribution of a region's voxels instead of its surface. We interpret the volumetric region R as a probability distribution and compute its *second central moment about the mean*, also called *variance*, i. e.,

$$\sigma^2(R) := \sum_{\vec{r} \in R} |\vec{r} - \overline{m}|^2 \tag{6.4}$$

where $\vec{r}$ are the coordinates of a voxel $\in R$ and $\overline{m}$ is the region's center of mass, i. e.,

$$\overline{m} = \frac{1}{|R|} \sum_{\vec{r} \in R} \vec{r} \tag{6.5}$$

From an intuitive point of view, the variance measures to what extent a region's mass is centered around its center of mass, in other words the variance determines the compactness of an object. More precisely, a sphere is the shape with minimal variance for a given volume, since no volume element can be moved any closer to the center of mass in order to reduce the variance. Therefore, we consider the variance of a volumetric region an appropriate measure of its blobbiness. However, the variance is heavily influenced by the size of an object, as an object with larger volume needs to occupy a larger region, which increases its variance. In contrast, we are interested in a size-independent measure of blobbiness, because we generally consider shape and size as independent concepts. We achieve this by expressing a region's variance relatively to the variance of a sphere with the same volume. First, we express a sphere's radius r as a function of its volume vol:

$$vol(r) = \frac{4}{3}\pi r^3 \quad \Leftrightarrow \quad r(vol) = \left(\frac{3}{4\pi} vol\right)^{1/3} \tag{6.6}$$

Second, we calculate the variance $\sigma^2_{r_0}$ of a sphere with radius r_0, which is centered

around the origin, by integrating over all sphere-surfaces with radius $\leq r_0$:

$$\sigma_{r_0}^2 = \int_0^{r_0} (4\pi r^2) r^2 dr = \frac{4}{5}\pi r_0^5 \tag{6.7}$$

Inserting Equation 6.6 into Equation 6.7 yields the variance of a sphere with volume vol:

$$\sigma_{sph}^2(vol) = \frac{4}{5}\pi \left(\frac{3}{4\pi} vol\right)^{5/3} \tag{6.8}$$

Now, we are able to express a region's variance in terms of the variance of a sphere of the same size. Hence, we define the blobbiness of a region R as:

$$blob(R) := \frac{\sigma_{sph}^2(|R|)}{\sigma^2(R)} \in [0;1] \tag{6.9}$$

6.2.2 Pre-segmentation

As such, volumetric data does not have a notion of shape. Some semantic information needs to be added to the intensity values associated with the voxels in order to be able to define shapes, mainly by specifying where object boundaries are located, i. e., which voxels are part of an object's surface. For typical volumetric data sets, this decision can be easy, as it is the case with the sharp contrast between a metallic object and surrounding air in an industrial CT scan, or more complicated such as for an inner organ surrounded by tissue of similar intensity in a medical context. Note, however, that for shape analysis not a full segmentation is required, only surfaces need to be specified. For example, two or more objects which would be placed in individual segments for a full segmentation can be placed in the same segment when only their surface is important. Splitting up this *pre-segmentation* into multiple sub-segments corresponding to individual objects is left to the subsequent shape classification. Hence, instead of a full-blown segmentation technique a much simpler method such as windowing can be applied to acquire a rough approximation of object surfaces inside the volume data.

6.2.3 Curve-Skeleton Computation

A major issue with skeleton computation is stability. Even small changes in the object data, as caused by noise, can have a great influence on the resulting curve-skeleton. This is less of a problem for our classification scheme, as we are analyzing each skeleton segment by itself, and therefore we incorporate more information with a statistical approach that is less likely to be influenced by local stability issues. Cornea

et al. (2007) have evaluated several methods for computing curve-skeletons, noting that the potential field method results in the cleanest and smoothest curve-skeletons. We were especially interested in smooth skeletons of low complexity, because heavily branching skeletons would cause problems for the volume decomposition and merging step. Therefore, we have chosen the potential field method, although it is the slowest of the examined techniques, but running time of the pre-processing is not a main issue for our use case.

Our setup uses the *pfSkel* application (Cornea et al., 2005) that computes the potential-based vector field of the object specified in the input pre-segmentation to construct the segments of the 3D curve-skeleton. The algorithm requires a binary volume as input to specify surface voxels. It outputs the generated curve-skeleton as multiple skeleton segments, each consisting of a polygonal chain of seed points. Additionally, information about critical points and high divergence points are given, but these are not used for our technique. To prevent cavities inside the pre-segmentation, caused by noise or other artifacts, from disturbing computation of the skeletons, we add extra layers of voxels at the surface of objects. While this can have the effect of smoothing the object and may lead to unwanted merging of adjacent features, it worked out to remove some noise with the relatively high resolution data sets we used. The only further relevant parameters are field strength, for which we chose a low value of 4 to get a minimal complexity skeleton, and percentage of high divergence points to use, for which we stayed with the default 20% for all examined data sets.

Though our statistically motivated classification approach is relatively robust towards imperfections of the curve-skeleton, there are still some computation artifacts and properties inherent to curve-skeletons that cause problems during the subsequent steps. Therefore, we perform three basic post-processing steps on the computed skeleton, which we call *skeleton normalization*:

1. Thin longitudinal structures such as vessels often exhibit gaps in the skeleton as depicted in Figure 6.3. Since these gaps hamper the applicability of the tubiness classifier, we close them by connecting each end node of the skeleton graph with its next neighbored skeleton point outside the end node's skeleton segment, if the connecting line lies completely within the pre-segmented object.

2. When a blobby shape is passing into a tubular structure, the skeleton segment running through both is not necessarily split up at the border region between the two shapes. It would be impossible to classify the blobby shape in isolation as the skeleton region corresponding to the skeleton segment also contains the respective part of the tubular structure. Moreover, in case the blobby part of the region is dominated by the tubular one, the region would not even be recognized as blob. Therefore, we split the

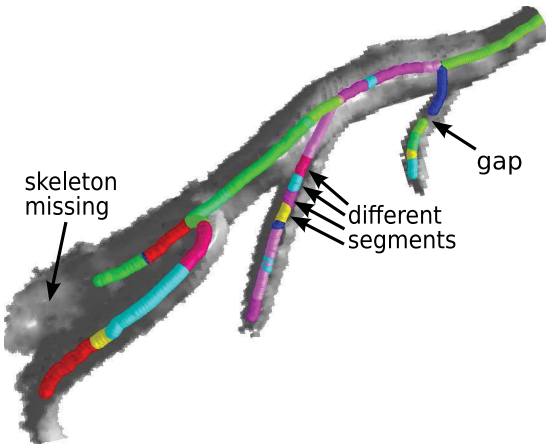
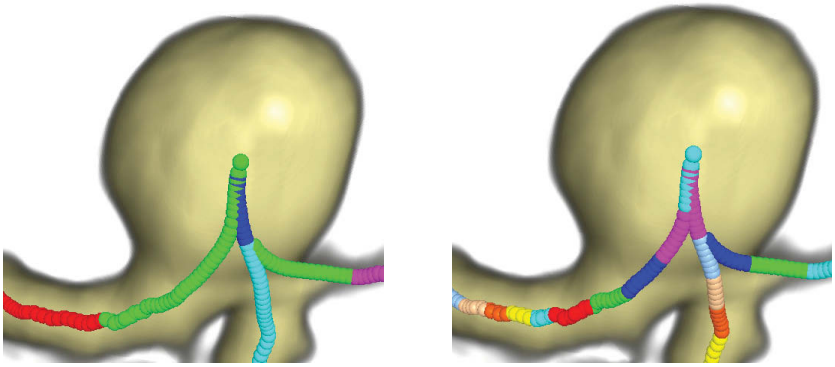


Figure 6.3: Close-up of curve-skeleton in an angiography data set prior to normalization. Pre-segmentation errors can lead to gaps in the skeleton, as well as missing skeletons for entire parts of the volume. The skeleton segments are shown with alternating colors.

curve-skeleton into segments not exceeding a certain length as shown in Figure 6.4. For the data sets with sizes of 256^3 to 512^3 we examined, length thresholds of five to ten voxels gave good results. Due to the curve-skeleton decomposition, however, the subsequent region merging step gains in importance.
3. Degenerated skeleton branches shorter than the length threshold are discarded (pruning) as these may cause problems during the region merging.

6.2.4 Volume Decomposition

The rough pre-segmentation consisting of a few very large segments is not suitable for any shape classification. Therefore, we have to further decompose the pre-segmented volume into smaller segments, which more likely exhibit pronounced shapes, before applying the shape classifiers. This decomposition is based on the previously computed skeleton. In a curve-skeleton as returned by the potential field algorithm, however, there is no connection between the skeleton segments and individual voxels in the data set. Thus, we perform a distance transform in order to compute the distance of each voxel to its nearest skeleton segment and assign the voxel to this segment. The distance transform is done by simultaneous region growing, where each of the skeleton segments is chosen as a separate seed. In contrast to the hierarchical mesh decomposition presented by Cornea et al. (2005), this includes not only surface voxels but all voxels associated with a skeleton segment, forming the *skeleton*



(a) (b)

Figure 6.4: Comparison of the curve-skeleton of an aneurysm blob before (a) and after (b) the splitting operation. Without splitting the left green segment ranges from the blob deep into the left vessel preventing a proper classification of the blob.

region. The region growing process assigns each voxel that was initially part of the pre-segmentation to such a skeleton region. The distance map containing the minimal distance of each voxel to its next skeleton segment is used in subsequent steps.

6.2.5 Merging of Skeleton Regions

The volume decomposition outputs a heavily over-segmented data set, since each segment of the normalized curve-skeleton is assigned a skeleton region. Although the over-segmentation is necessary in order to make sure that no region of the decomposed volume contains structures of different shape classes, it also causes the skeleton regions to loose their initial shape property. For instance, the splitting of a vessel causes an originally tubular region to become blobby. Therefore, the fusion of skeleton regions into classifiable units is a crucial step of the classification process. One might get the impression that volume decomposition and region merging are inverse operations. However, while the volume decomposition is based on the geometry of the curve-skeleton, the fusion process also takes into account the shape properties of skeleton regions. The merging steps are as follows:

- First of all, very small regions are merged with the neighbor region with which they share the largest part of their surface, since those regions cannot be expected to have any meaningful shape. In order to avoid the necessity to manually specify a minimal segment size, we define each segment, whose

Figure 6.5: Blob merging demonstrated on one lung of the NCAT phantom (Segars, 2001). (a) shows the curve-skeleton after normalization, (b) the corresponding skeleton regions, (c) the regions after fusion.

size deviates down from the mean segment size by more than two times the standard deviation, as too small.

- Tube merging. In order to allow a reliable classification of tubes, split tubular structures have to be merged as far as possible, while preventing erroneous fusions with blobs or surface-like structures. We have chosen the tubiness score as merge criterion. More precisely, we fuse all pairs of neighbored regions that both as well as the merge result have a tubiness score above a certain threshold. Note that the tubiness classifier itself is not affected by the splitting of tubular structures. Instead, the misclassification is caused by the fact that the splitting significantly increases the blobbiness score. Therefore, it is reasonable to base the tube merging on the tubiness classifier. In addition, the merging stops at skeleton branches, i.e. two regions are not merged, if the connection point of their skeleton segments is the origin of a third segment. In our examinations, a tubiness threshold of around 0.8 turned out to be the best choice.

- Blob merging. The curve-skeleton of blobby structures is usually heavily branching, which in combination with the skeleton splitting leads to many thin regions that would not be classified as blobby as illustrated for one lung of the NCAT phantom in Figure 6.5a. Such regions have in common that they share a much larger border with other regions (inner surface) than with the background (outer surface). We merge regions with a high inner-surface-to-outer-surface

ratio. As a result, split up blobby structures are usually not fused into a single region, but the remaining regions have regained a blobby shape and can thus be classified correctly, as visible in Figure 6.5c.

- Quality-based merging. Since one main objective of the previous fusion steps is not to erroneously merge regions of different shape classes, these steps might fail on imperfect skeleton segments or object surfaces. For instance, a vessel region with a rough surface due to artifacts in the pre-segmentation may have a low tubiness score and might therefore not be merged with neighbored vessel regions. The last fusion step copes with these situations by trying to improve the clarity of the classification: two regions are merged when the classification of the resulting shape is less ambiguous than the input regions' classifications. We define the ambiguity of a region R as the quotient of its lowest and highest classification score:

$$amb(R) := \frac{\min\left(\tau(R), surf(R), blob(R)\right)}{\max\left(\tau(R), surf(R), blob(R)\right)} \tag{6.10}$$

6.3 User Interface for Shape Transfer Functions

To be able to intuitively assign optical properties to certain shape classes as identified by our approach, a sufficient user interface is required. In this section, we briefly describe such a user interface concept. It allows us to represent the 4-dimensional transfer function space, given by the classifiers tubiness, surfaceness and blobbiness as well as the intensity values, in a comprehensible way. For simplicity, we consider the shape-based transfer function assignment as a two-stage process, where the user first constrains the shapes to be visualized before defining the desired intensity range. The normalization of the shape classifiers makes them comparable and allows the user to intuitively constrain the visualization to certain shape classes. This normalization gives us the opportunity to use an equilateral triangular shape selection widget based on barycentric coordinates (see Figure 6.6), similar to the one proposed by Kindlmann and Weinstein (1999) for DTI data. Each corner of the triangle represents one of the three shape classes. For each computed shape segment we place a marker inside the triangle in such a way that the marker's position indicates the likelihood that the shape class is best represented by one of the three basic shapes. For the placement of each marker we use the barycentric coordinates defined by the three normalized shape classifiers. Then a marker is placed closer to those corners of the triangle that represent the shape classes the identified segment is best classified as. Thus, in the first processing step, the user is able to select an arbitrary number of markers to

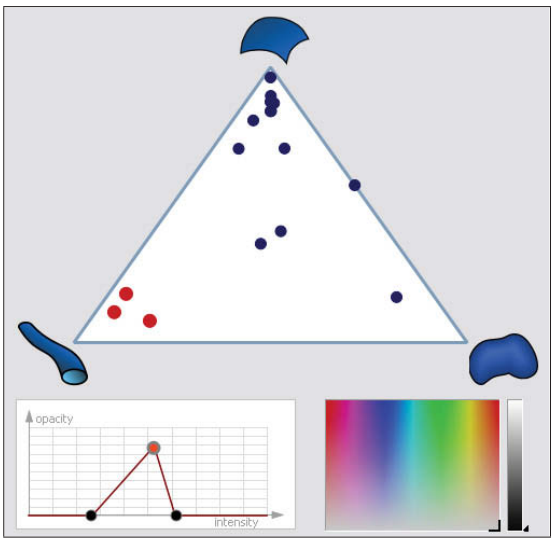


Figure 6.6: Shape-selection widget. Each shape is represented by a marker whose position depicts its degree of membership in the three shape classes. The user has selected three tubinal shapes (red).

constrain the shape classes to be visualized. In the second step, a conventional 1D transfer function is used to assign optical properties to the selected shape classes. For the examination of data sets with no planar structures, the surfaceness class can be omitted collapsing the triangle to a line.

6.4 Integration into Voreen

Figure 6.7 shows the Voreen data-flow network implementing the presented shape classification approach. The sub-network on the left is responsible for the actual shape classification and roughly reproduces the abstract workflow depicted in Figure 6.1. The `VolumeThreshold` processor is a standard component that performs a basic background removal on the input data set by applying lower and upper intensity thresholds (pre-segmentation). The curve-skeleton of the pre-segmented data set is computed by the `PFSkel` processor, which is a thin wrapper around the *pfSkel* application (see Section 6.2.3). The resulting curve-skeleton is transmitted to the `RegionGrowing` processor, which uses it to decompose the pre-segmented input volume by applying simultaneous region growing as described in Section 6.2.4. The `RegionGrowing` processor provides two volumes containing the resulting skeleton regions and the corresponding distance map, respectively. Since the following

three steps of the classification workflow, namely skeleton region merging (see Section 6.2.5), shape classification (see Section 6.2.1), and transfer function specification (see Section 6.3), all require the computation of shape descriptors for volumetric regions, we decided to concentrate these tasks in the single `ShapeClassifier` processor. The `ShapeClassifier` does not only perform the computations regarding the shape classification, but also has a shape-selection widget attached that enables the user to assign optical properties to the detected shapes (see Figure 6.6).

The rendering of the shape-classified data set is performed by the modified raycasting pipeline on the right of the data-flow network. The commonly used `SingleVolumeRaycaster` has been replaced by the `SegmentationRaycaster`, which offers the possibility to assign a separate transfer function to each segment of a data set. Accordingly, it has an additional inport expecting a segmentation volume that assigns a segment ID to each voxel in the volume to be rendered. In our case, this segmentation volume is the result of the region merging step and is provided by the `ShapeClassifier`. The user-defined shape transfer function is transmitted from the `ShapeClassifier` to the `SegmentationRaycaster` via a property link.

6.5 Results

We have tested our technique with several synthetic as well as real-world data sets, mostly from the medical domain. All pre-segmentations of the data sets presented in this section are a result of windowing. We stayed with our default merging parameters of 0.8 for the tubiness threshold and 0.1 for the maximum inner-surface-to-outer-surface ratio, while adjusting the maximum skeleton segment length to the sizes of the data sets. User interaction was only required for the specification of the intensity range used for the pre-segmentation and for the assignment of transfer functions to the shapes, while the intermediate classification process runs automatically. Preprocessing times for the different data sets are given in Table 6.1. All tests were conducted on a system equipped with an Intel Core 2 Quad Q9550 CPU and an NVIDIA GeForce GTX 280 graphics board.

Figure 6.8 depicts the single steps of the classification process for an angiography data set. The volume decomposition based on the normalized curve-skeleton yields a heavily over-segmented volume (see Figure 6.8b). Therefore, the initial skeleton regions mostly lack pronounced shapes and exhibit rather random shape scores, which are distributed almost uniformly over the whole spectrum. The region merging fuses the more than 200 initial regions to nine final features with pronounced shapes (see Figure 6.8c): The aneurysm is correctly classified as an isolated blob, while the three large vessel regions (blue, green, ocher) are represented correctly by the

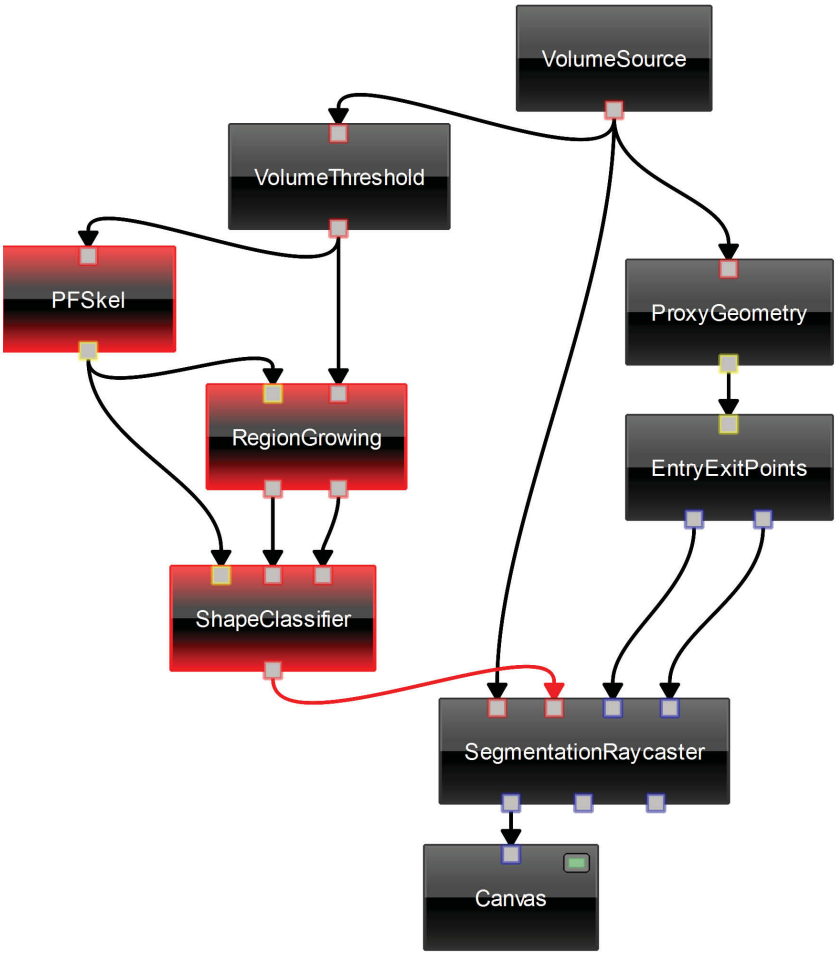


Figure 6.7: Voreen data-flow network implementing the presented shape classification approach. The sub-network on the left performs the classification of the input volume, while the raycasting pipeline on the right renders the resulting segmentation. Custom processors are highlighted in red.

data set	resolution	t_{skel}	t_{class}
angiography 1 / 2	256^3	210 / 176	13 / 9
mouse (cardiac)	$225 \times 178 \times 256$	512	7
mouse (torso)	$360 \times 290 \times 400$	1930	27

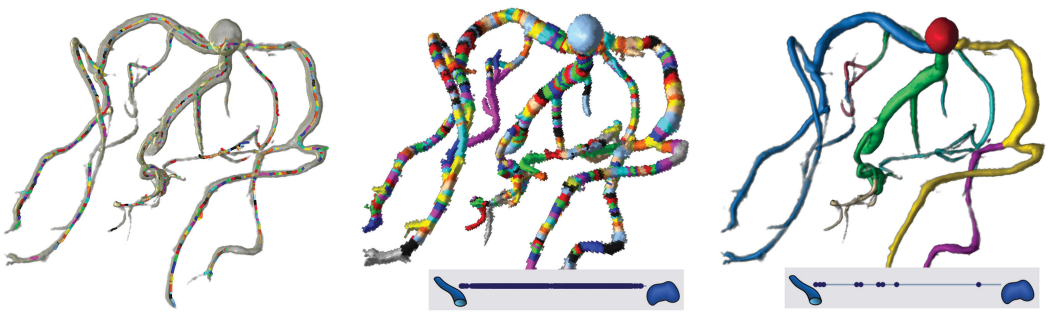
Table 6.1: Statistics for the data sets presented in the results section. Given are the data set resolution, the times for computation of the curve-skeleton and for the classification (in seconds).

left-most shape cluster in the user interface. The ambiguity of the five remaining vessel shapes is caused by imperfect skeleton segments, which lower the tubiness score of their regions. However, these regions are still clearly distinguishable from the aneurysm.

Figure 6.9 shows the classification results of two angiography data sets. In both cases the blobby aneurysms could be easily isolated from the tubular vessels. We omitted the surfaceness classifier for all angiography data sets, since they do not contain such structures.

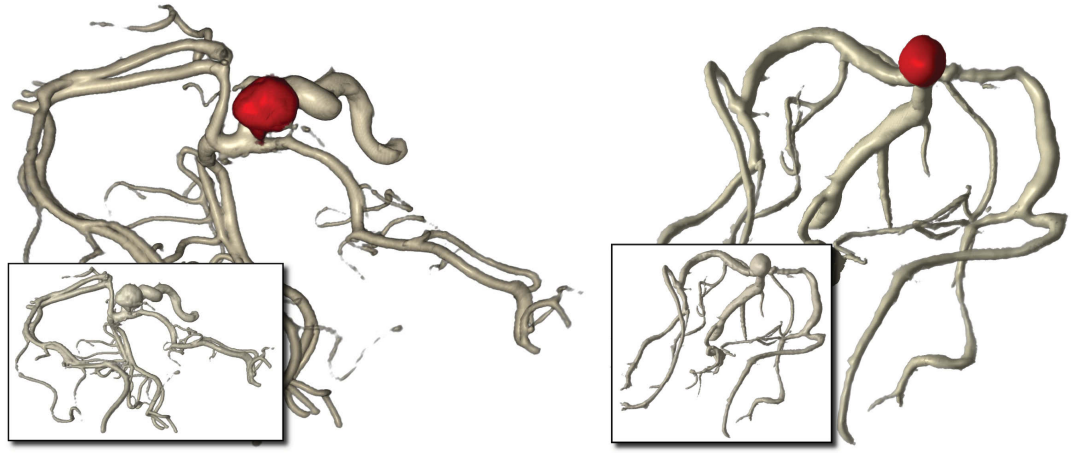
In Figure 6.10 we applied our technique to the CT scan of a mouse heart. Though the shape classification is less clear than for the angiographies, the two shape clusters corresponding to the blobby heart structures and the vessels, respectively, can be easily identified in the user interface. Furthermore, the vessels are mostly correctly separated from the heart, though a slight misclassification is visible in the third rendering where parts of the vessels have been classified as blobby. This is due to the fact that these vessel regions touch the heart structures and are merged with them.

Figure 6.11 shows the shape-classification of a CT scan of a mouse lying on a bed. This case is interesting for several reasons. Since the heart and vessels are filled with contrast agents, their intensity range overlaps with the bones' range and can therefore not be separated by a conventional 1D transfer function, a typical situation in contrast-enhanced CT scans. Furthermore, the bed as a surface-like structure also shares this intensity range as visible in Figure 6.11a. The shape-classification shown in Figure 6.11b allows a separation of the heart from the bones and vessels as well as the bed. We consider the classification of the elbows to be correct, since though they are part of a longitudinal structure the elbows themselves are of blobby shape. The blobby classification of parts of the shoulder bones, however, is an error that is caused by gaps in the skeleton in these regions. In Figure 6.11c the classification has been manually refined through the user interface by removing the blobby structures outside the heart.



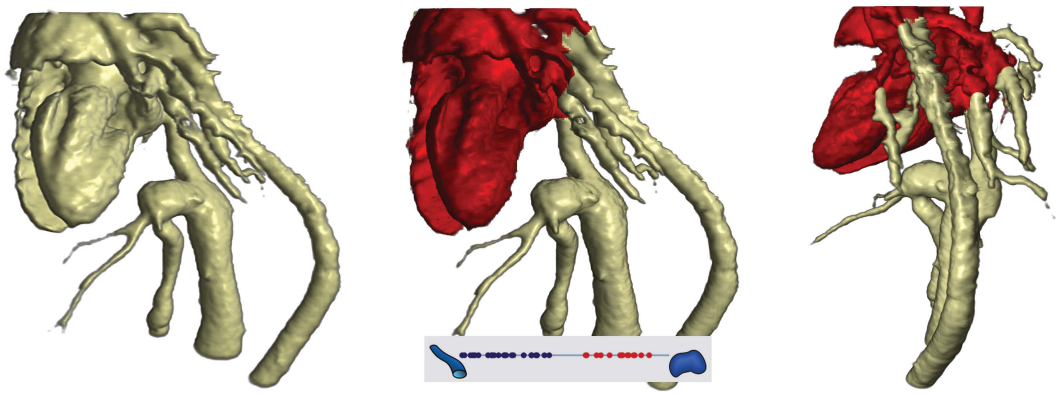
(a) (b) (c)

Figure 6.8: Classification workflow for the angiography data set shown in Figure 6.9b. Subfigure (a) displays the normalized curve-skeleton, while (b) and (c) present the skeleton regions before and after merging along with the corresponding shape distributions.



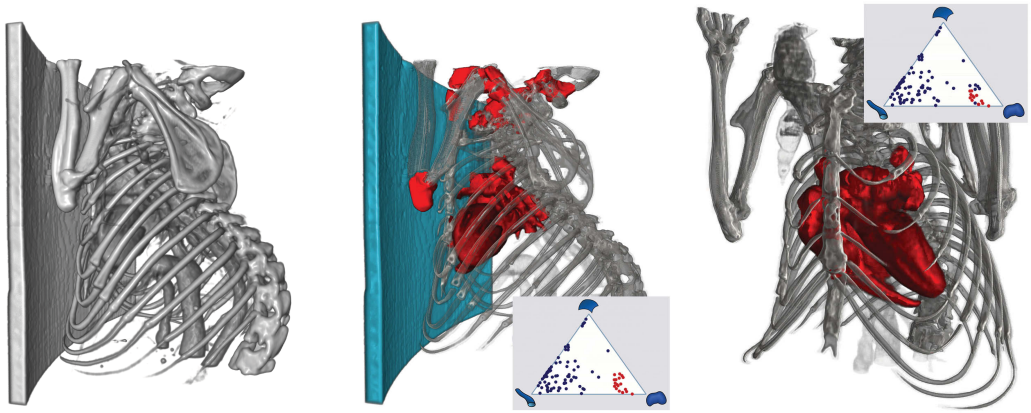
(a) (b)

Figure 6.9: Shape-classified volume renderings of two angiography data sets, each containing an aneurysm. The thumbnails show renderings generated with a conventional 1D transfer function, for comparison.



(a) (b) (c)

Figure 6.10: Application of the proposed classification technique to a CT scan of a mouse heart. (b) and (c) show renderings of the classification result from two perspectives along with the corresponding shape distribution. The red-labeled heart structures correspond to the shapes selected in the user interface. The rendering in (a) was generated with a conventional 1D transfer function.



(a) (b) (c)

Figure 6.11: Shape-classified CT scan of a mouse lying on a bed. (a) was rendered with a conventional 1D transfer function. In (b) the bones/vessels are rendered semi-transparently, while the heart and the bed have been colored independently. The blobby structures (red) have been selected in the user interface. (c) shows a further refined classification where the blobby features outside the heart have been selected in the user interface in order to assign the same optical properties that have been chosen for the bones/vessels.

6.6 Current Limitations and Future Work

A principal limitation of our approach is the dependency on a proper masking of the volumetric structures that are to be shape-classified. While such a masking might be tedious in the general case, creating a sufficiently precise pre-segmentation for volumetric scans with contrast agents, which are a typical use-case for volume visualization in the medical domain, is possible with little effort by windowing.

If an adequate masking is provided, errors or ambiguities in the shape classification are mainly caused by incomplete merging operations, since small regions are less likely to exhibit a pronounced shape than larger ones. On the other hand, lowering the merge barrier increases the risk of erroneously fusing shapes of different classes. We want to investigate whether a more global merge strategy, which does not focus on the local neighborhood of regions but rather tries to increase the overall clarity of the classification, can improve this issue. Furthermore, we currently only use information from level 0 of the skeleton hierarchy, the "core skeleton", while incorporating low divergence points and high curvature points added in levels 1 and 2 might help to refine the region merging.

A further issue is the significant time consumption of the shape classification and especially the skeleton computation, which do currently not allow an interactive parameter tuning. While we are positive that a more efficient implementation can achieve interactive results for the merging and classification steps, it should also be investigated whether a simpler skeletonization algorithm could possibly give similar results while having less demanding runtime requirements compared to the potential field technique.

6.7 Summary

In this chapter, we have introduced shape-based transfer functions in order to blur the border between simple but limited classification and powerful but costly segmentation techniques. In contrast to previous approaches towards shape classification in volume data, which perform a voxel level classification, the proposed technique does not require the user to interpret complex histograms but provides him/her with a manageable set of shape-classified volumetric features and offers an intuitive interface for the assignment of optical properties. Since we consider shape, texture and size as independent properties of volumetric features, we believe that a combination of our approach with texture-based (see Caban and Rheingans, 2008) and size-based (see Correa and Ma, 2008; Hadwiger et al., 2008) classification schemes might be worth investigating.

Chapter 7

Volume Segmentation

In this chapter we review related work in the field of semi-automatic, non-model-based volume segmentation. Besides the basic segmentation algorithms, we present volume segmentation systems that particularly focus on the efficient integration of user interaction and feedback into the segmentation process. Furthermore, approaches for dealing with segmentation uncertainty are covered.

An accurate segmentation of a volume data set does not only facilitate the generation of high-quality visualizations, but is also a prerequisite for quantitative volume analysis. For example, the determination of the volume of a tumor in a medical scan is not possible without a precise segmentation. While fully automatic segmentation approaches exist, they are based on models of the structures to segment and cannot be used for other purposes or when a certain degree of abnormality is exceeded, which often applies to pathological medical cases, such as tumors or ruptures. In most application cases, a reliable volume segmentation can only be obtained by incorporating expert knowledge into the segmentation process. In the remainder of this chapter, we give an overview of semi-automatic volume segmentation approaches, which do not assume a priori knowledge about the data set and are therefore universally applicable.

7.1 Semi-automatic Volume Segmentation

Based on the supported user interaction, supervised volume segmentation techniques can be categorized into two types (Olabarriaga and Smeulders, 2001): *Boundary-based* methods aim at detecting the boundary between the desired structures and the background. They either take a small set of boundary voxels as user input, or require the user to provide a nearly complete boundary. *Region-based* techniques consider the

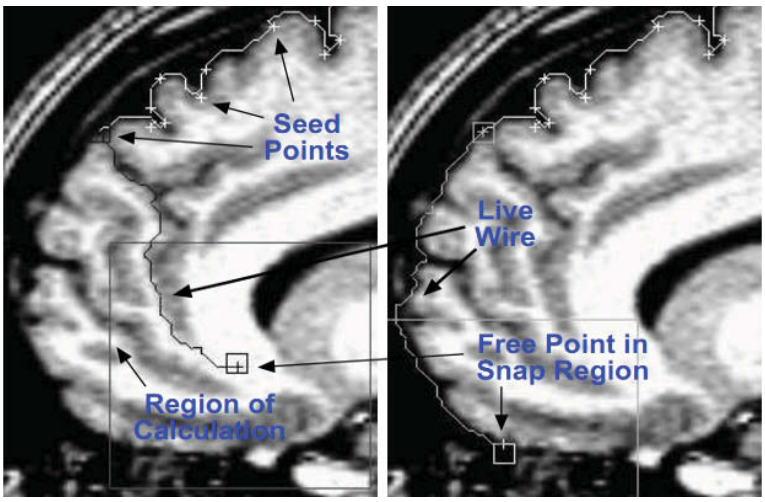


Figure 7.1: Livewire: The system computes minimal paths connecting user-specified control points placed along the object's boundary (Hastreiter and Ertl, 1998).

region to be extracted as a continuous set of similar voxels, and they expect the user to specify an initial set of *seed* voxels belonging to the desired object.

7.1.1 Boundary-based Approaches

The most prominent group of boundary-based segmentation approaches are active contours and level sets (Sethian, 1999). All these techniques require that the user places a contour near the desired boundary, which is then evolved to a local energy minimum. The algorithms mainly vary in the design of the energy functional to be minimized. The complex setup and the limited user interaction have traditionally been considered as severe drawbacks of the level set technique hampering its application in interactive workflows. However, recent presentations of interactive, level set based segmentation tools by Ben-Zadok et al. (2009) and Cremers et al. (2007) might indicate a change in that respect.

The intelligent scissors algorithm (Mortensen and Barrett, 1995) is a boundary-based segmentation technique for 2D images, also known as livewire. It computes a minimum-cost path between user-specified boundary points via Dijkstra's shortest-path algorithm. Hastreiter and Ertl (1998) have extended it to volume segmentation by considering inter-slice relations and a 3D filter to compute the local cost. Figure 7.1 shows the user interface of their system.

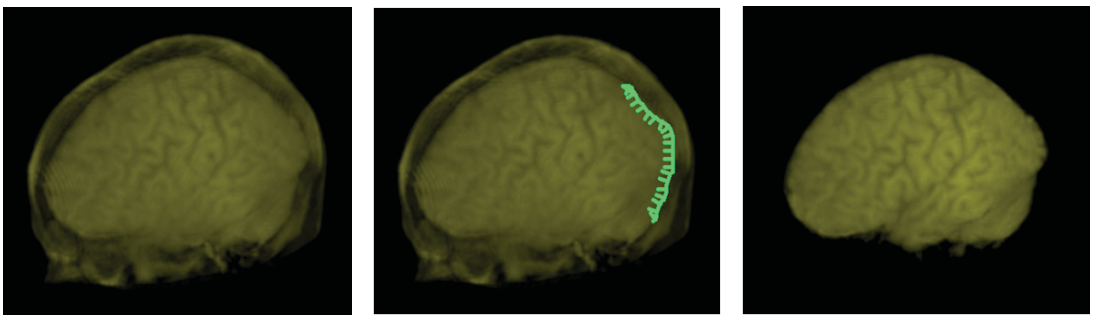
7.1.2 Region-based Approaches

Region growing is a basic region-based segmentation method that starts from an initial set of seed points and iteratively adds neighboring voxels when they satisfy a similarity criterion, which is usually based on intensity thresholds (Huang and Ma, 2003) or a diffusion metric (Sherbondy et al., 2003). Chen et al. (2006) supplemented the basic approach with a sketch-based interface for the efficient specification of seed points, which is based on the volumetric extrusion of user-drawn 2D sketches. Although region growing in many cases achieves fast segmentation results that correspond well to the perceived boundaries, the choice of the homogeneity criterion becomes difficult for data sets with weak boundaries or a high level of noise (Zhu and Yuille, 1996).

The graph cuts technique proposed by Boykov and Kolmogorov (2001) interprets the image as a graph, weighted to reflect intensity changes. While the algorithm can be easily generalized to 3D, it suffers from the ”small cut” problem: since the algorithm returns the smallest cut separating the seeds, a small number of seeds might not be adequate. Grady (2006) introduced the random walker technique as a probabilistic, seeded image segmentation approach. Like graph cuts, it views the volume as a weighted graph, whose nodes correspond to the voxels. The basic idea is to determine for each voxel the probability that a random walker starting from there first reaches one of the user-specified foreground seeds. The probability with which a certain neighbor voxel is chosen as next step is defined by edge weights, which have to reflect the characteristics of the volume and are usually derived from intensity differences between the respective voxels. A crisp segmentation may finally be obtained by assigning each voxel to the label for which the highest probability was calculated. In Chapter 8 we present an uncertainty-guided volume segmentation system based on the random walker technique.

7.2 Volume Segmentation Systems

In addition to the segmentation algorithms outlined above, the research community has presented several volume segmentation systems, which can be classified based on accuracy, repeatability and interaction efficiency (Olabarriaga and Smeulders, 2001). Bartz et al. (2003) have proposed a hybrid approach for the segmentation of the tracheo-bronchial tree of the lungs, where they limit the user interaction to the specification of seed points. Gu and Peters (2004) have introduced a system for interactive organ segmentation based on the combination of an erosion operation and the fast marching method. Boykov and Kolmogorov (2000) employed the graph



(a) initial rendering **(b)** boundary stroke **(c)** resulting segmentation

Figure 7.2: 2D-to-3D stroke elevation metaphor used by the Volume Catcher system: the user marks the boundary of the object to segment in the volume rendered image (Owada et al., 2005).

cuts technique for interactive organ segmentation. Ben-Zadok et al. (2009) recently presented a volume segmentation tool based on level sets that focuses on image-guided therapy. Tzeng et al. (2005) have presented a system which is based on a combination of machine learning and interactive painting. By interactively painting on slices, the system can be trained and the whole volume can be classified. The Volume Catcher system presented by Owada et al. (2005) lets the user segment a data set by directly marking features of interest in the volume rendered image. It employs a 2D-to-3D stroke elevation algorithm in order to identify volume boundaries from 2D sketches drawn onto the image plane (see Figure 7.2). The generated 3D sketch is then used to parametrize a subsequent segmentation algorithm. Since the stroke elevation is based on gradient analysis, however, the technique is limited to data sets with clear boundaries and a low level of noise.

7.2.1 Incorporating Uncertainty

The systems named so far do not take into account the issue of segmentation uncertainty. In fact, though some fully-automatic systems incorporating or providing probabilistic segmentation information exist, *guided* uncertainty-aware volume segmentation systems are rarely found. Cremers et al. (2007) propose a probabilistic level set formulation for allowing the integration of user input into the segmentation process. Kontos et al. (2006) presented a system for segmentation of medical data sets with respect to uncertainty. By combining manual thresholding and automated boundary correction, they achieve accurate segmentations for 3D SPECT data. Uncertainty is not conveyed to the user, but only used for automatic improvements.

More recently, Saad et al. (2010) proposed an exploration and editing system for probabilistic segmentations of medical data sets. They analyze a given probabilistic segmentation in order to detect ambiguous regions and to allow the user to correct potential misclassifications. The main characteristic of the system is the decoupling of the segmentation process and the uncertainty handling, which makes it applicable to any segmentation algorithm providing a probabilistic segmentation. On the downside, however, error corrections can only be made based on the input segmentation, whereas the underlying volume data cannot be considered.

Chapter 8

Uncertainty-Aware Guided Volume Segmentation

Reliability of a volume segmentation is of critical importance in many application cases. In this chapter we present a guided probabilistic volume segmentation approach that focuses on the minimization of uncertainty. In an iterative process, our system continuously assesses uncertainty of a random walker-based segmentation in order to detect regions with high ambiguity, to which the user's attention is directed to support the correction of potential misclassifications. This reduces the risk of critical segmentation errors and ensures that information about the segmentation's reliability is conveyed to the user in a dependable way. In order to improve the efficiency of the segmentation process, our technique does not only take into account the volume data to be segmented, but also enables the user to incorporate classification information. An interactive workflow has been achieved by implementing the presented system on the GPU using the OpenCL API. Results obtained for several medical data sets of different modalities, including brain MRI and abdominal CT, demonstrate the reliability and efficiency of our approach.

Especially when used in the context of 3D volume visualization, conventional classification is not only a cumbersome task, but is often also regarded as unreliable by users. In contrast to traditional slice-wise examination of volume data sets, 3D visualization inherently suffers from occlusion and therefore requires masking through classification and/or segmentation in order to get an occlusion-free view to the structures of interest. This, however, poses the risk of unintentionally masking parts of the structure of interest or erroneously assigning parts of the background to the feature of interest. Both of these effects might lead to critical misinterpretations during visual inspection and to an erroneous quantitative analysis, e. g., when measuring a tumor's volume. Given the well-accepted importance of uncertainty visualization as one of

the top visualization research challenges (Johnson, 2004), surprisingly few research papers have actually tackled the issue of uncertainty arising through the volume classification process.

In this chapter we propose a guided segmentation approach, which focuses on uncertainty and is conducted by an interplay between the user and the system. The underlying workflow has been designed as a tight interplay, where each subtask is delegated to the optimal interaction partner. While the human user is rather strong in visual pattern recognition and can reliably estimate an appropriately visualized situation, the strength of the system is vast processing power, which can be exploited to support expensive computations as well as a rough analysis. As shown in Figure 8.1, in the first step of the workflow, we exploit the probabilistic random walker segmentation algorithm (Grady, 2006) in combination with the high processing power of the GPU to generate segmentations with known uncertainty. In the second step, we analyze this uncertainty information. In order to allow the user to judge and modify the uncertain regions, our system directs the user's attention to regions with high uncertainty in step 3. Finally, within step 4 the user can inspect the results and refine the starting parameters of the previous segmentation, and the workflow starts over again through the feedback loop. This workflow ensures, that the user is always aware of the currently involved uncertainty, which is essential to judge the reliability of the results.

The main contribution of this chapter is an uncertainty-guided volume segmentation workflow, which is based on a GPU implementation of the random walker segmentation algorithm. The user is able to generate a segmentation through the interactive workflow, within which uncertain regions are automatically extracted and presented to the user in the order of their importance. Thus, the user can quickly segment unambiguous regions, which usually constitute the largest part of a data set. The system then directs the user's attention to the uncertain parts of the segmentation and allows the user to locally refine the segmentation, and it ensures that the uncertainty information is conveyed in a reliable way. Since transfer functions usually allow one to obtain a rough classification quickly, incorporating classification information in the segmentation process is exploited to further improve its efficiency. To our knowledge, the presented system is the first that integrates direct visual feedback of segmentation uncertainty into an interactive segmentation approach.

8.1 Related Work

Since uncertainty visualization plays a key role within our system, we briefly review the most relevant approaches. A general overview of the field of guided volume

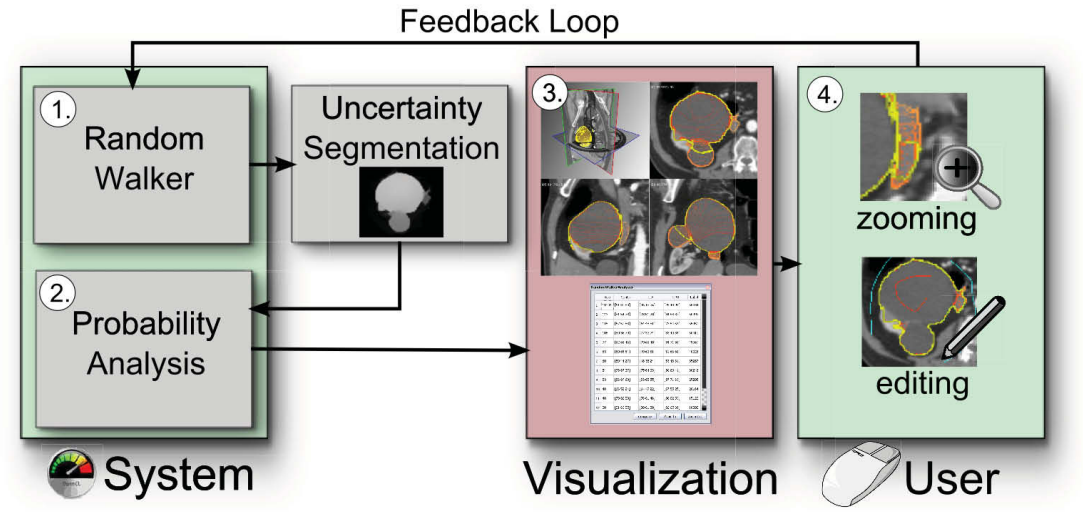


Figure 8.1: Proposed workflow: The system generates a random walker solution using seeds that have been defined in previous iterations, analyzes the probabilistic field in order to detect uncertain regions, conveys this uncertainty information to the user by using a table widget as well as integrating it into the 2D and 3D views, thus allowing the user to select and modify uncertain regions.

segmentation has been given in Chapter 7.

In their work originally targeted towards uncertainty visualization in a geospatial context, MacEachren et al. (2005) classify uncertainty visualization based on data types and data quality. Although due to the context no volumetric uncertainty visualization is discussed, the principle of crispness and transparency can be transferred to our approach. In the same context, Pang (2001) describes illustrative techniques for multivariate uncertainty data. Grigoryan and Rheingans (2004) exploit uncertainty-based surface displacement. In their paper they also demonstrate the application to segmentation results. However, it requires a surface with an appropriate normal for which uncertainty information is present, which we cannot directly derive from our uncertain segmentation results. Pang et al. (1996) describe alternative concepts for the visualization of surface uncertainty by means of bumps, glyphs and so-called fat surfaces. Even more relevant to our approach are the color, opacity and texture mapping techniques proposed by Rhodes et al. (2003). While this technique is also tailored towards surface representations, the speckle, texture and noise approach presented by Djurcilov et al. (2001) has been developed for volumetric data sets. The same is true for the probabilistic animation approach presented by Lundström et al. (2007).

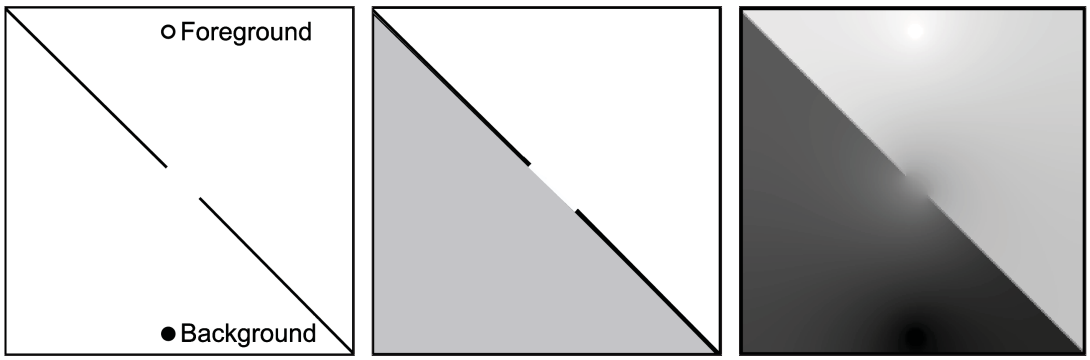
8.2 System Design

In order to be able to provide an efficient guided segmentation workflow, the selection of an appropriate probabilistic segmentation algorithm was crucial for our system. In particular, we surveyed existing semi-automatic approaches with respect to the following requirements:

1. Availability of information about local reliability of an obtained segmentation.
2. Efficient and reliable user interaction, i.e., small changes of the user-provided information should not be prone to cause large, unanticipated changes in the resulting segmentation.
3. Fast computation.

The first requirement already excludes the majority of techniques, including region-growing (Huang and Ma, 2003) and intelligent scissors (Mortensen and Barrett, 1995), which provide binary segmentations. While there exist some fuzzy formulations of the level-set concept, which might allow one to derive uncertainty information, few of them have actually been applied to volumetric data, and to our knowledge all of these approaches are tailored towards specific use-cases, mainly brain MRI (Chen et al., 2008). Additionally, due to the limited amount of interactivity, especially regarding the correction of intermediate results, level-sets do not seem to be well-suited for a guided workflow. While Ben-Zadok et al. (2009) and Cremers et al. (2007) presented promising advances towards interactive level set segmentation, their approaches do not provide probabilistic results.

Although the graph cuts technique itself is also limited to provide a crisp segmentation, Kohli and Torr (2008) have shown recently how a confidence measure can be derived from a graph cut solution based on min-marginals. So we had the choice between graph cuts and the random walker approach, which is inherently probabilistic, since it provides each voxel's probability for the membership of all segments. The user interaction is very similar with both techniques: the user specifies two (small) sets of foreground and background seed voxels to sketch the desired segments. Thus, when dealing with well-shaped data sets, both algorithms are adequate. However, the "small cut" phenomenon of the graph cut algorithm hampers the reliability of the user interaction as illustrated in Figure 8.2: since the surface area of the provided seeds is smaller than the weak (missing) part of the boundary, the smallest cut surrounds the seeds assigning all pixels to the background (a), which is most likely not the result the user expected. Additionally, adding a rather small set of further seeds, so that the seed surface becomes larger than the weak boundary, would cause a drastic change in



(a) Image with seeds **(b)** segmentation **(c)** probability map

Figure 8.2: Weak border detection property of the random walker: in contrast to the graph cut approach, the random walker is able to segment the objects although an interrupted boundary is present (adapted from Grady (2006)).

the segmentation towards the ”correct" result. For 3D segmentation, the requirement to draw seed regions whose surface area exceeds the weak boundary area becomes even more problematic, since the user typically defines less seeds in comparison to the number of voxels than for 2D image segmentation. Figure 8.2b, in contrast, demonstrates the weak border detection property of the random walker, which is accompanied by the reflection of the weak boundary in the surrounding probabilities as shown in Figure 8.2c. Another problem of the graph cuts approach is its tendency to produce blocky artifacts at the object boundary (Boykov and Kolmogorov, 2003), which especially becomes a problem when voxel-precise segmentation is desired.

For these reasons, we have selected the random walker algorithm to be the foundation of the proposed system. However, this decision is to some extent contrary to the requirement for a fast computation, since the graph cuts approach is known to run in real-time even for 3D data, whereas the time necessary for the computation of a random walker solution of medical data sets typically ranges from seconds to several minutes. To overcome this limitation, we exploit the features of current GPUs and provide an OpenCL implementation of a conjugate gradient solver used for the random walker calculations as described in Subsection 8.5.2. Thus, we achieve computation times which support the interactive and iterative proceeding of our workflow.

8.3 Guided Volume Segmentation

Although uncertainty information is present in the probabilistic segmentation generated by the random walker algorithm, exploiting this information in a user-centric workflow is not trivial. In case of a multi-label segmentation, i. e., more than two segments are considered, the probabilities constitute a vector field, and even for two-label segmentation a combined 3D visualization of the data set and the probability volume seems to be hardly practical. In contrast, a manual slice-wise examination of probabilities and volume data is a cumbersome process for large data sets, posing the risk of missing critical misclassifications. The low-frequency character of the probability field further amplifies the risk of overlooking areas of uncertainty.

Throughout this chapter, we adopt the commonly used medical data model, which assumes that a volume is composed of rather homogeneous regions representing the scanned materials, which intermix at their borders constituting smooth transitions (Kindlmann and Durkin, 1998). Under this assumption, the probabilistic segmentation can be expected to exhibit thin uncertainty margins at the boundaries. Therefore, large areas of ambiguous probabilities can be interpreted as indicators for potential misclassifications. We exploit this observation in order to derive information about the uncertainty of the segmentation result from the probability distribution.

The proposed system supports the user during the iterative refinement of probabilistic segmentations by analyzing the probability field to detect pronounced ambiguous regions and directing the user's attention to these. Since reliability is the main focus of our work, we have decided to employ a slice-based interface for user input (see Figure 8.3). The three axis-aligned slices are used for seed point definition as well as for displaying the current segmentation result along with ambiguous regions. The 3D multiplanar view provides context information about the current position of the slices in the data set and shows an iso-surface rendering of the segmentation with highlighted uncertain regions. Thus, we simultaneously convey the shape of the segmentation as well as the spatial relations of uncertain areas, which are both aspects that are difficult to perceive from a pure slice-wise representation. Although the random walker algorithm is applicable to K-way segmentation with an arbitrary number of labels, we decided to confine our system to 2-way segmentation, since the extraction of a single feature of interest from the background is the most common application for medical image segmentation and the simultaneous classification of multiple features would inevitably increase the complexity of the user interface. If necessary, multiple features can still be extracted iteratively.

The proposed segmentation system differs from previous work dealing with uncertainty in volume data in a significant way: Instead of just conveying the uncertainty

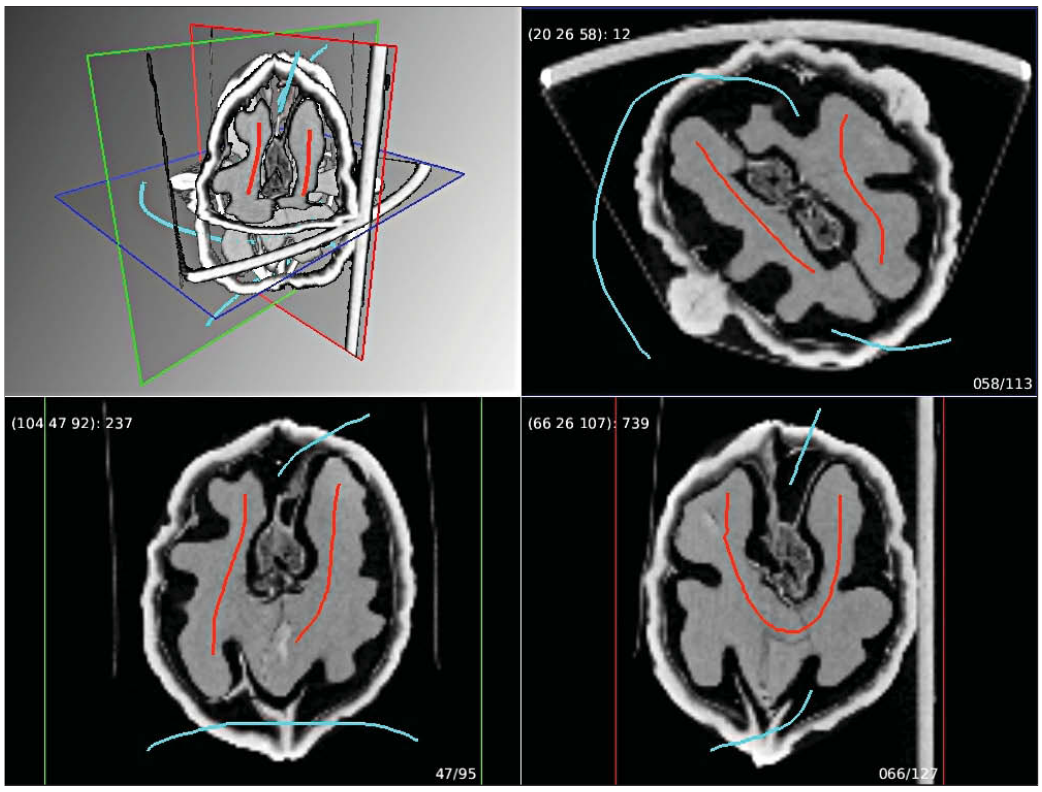


Figure 8.3: User interface of the proposed system: The upper right, lower right and lower left views present axis-aligned slices of the data set to be segmented. The upper left multiplanar view conveys contextual information about the current slice positions. The user has drawn foreground seeds (red) and background seeds (blue) onto the 2D views.

of a given segmentation to the user or allowing the subsequent modification of the volume, as for instance proposed by Kniss et al. (2005) in the form of a Bayesian risk minimization framework, we consider uncertainty already in the segmentation stage and propose a combination of interaction and visualization techniques developed for allowing the user to reliably and efficiently minimize the uncertainty. When conducting first tests with the random walker technique, we noticed that for many use cases, as for instance organ segmentation, usually large parts of the desired object could be properly segmented quite easily, while ambiguous parts of the segmentation where rather small and scattered across the surface of the object, where they were sometimes hard to detect. Therefore, we based the design of our system on Shneiderman's visual information seeking mantra (Shneiderman, 1996): "Overview

first, zoom and filter, then details-on-demand“. Therefore, we not only exploit the system for actual computation of the random walker, but also for a preclassification of uncertain regions and the visual presentation of these. Thus the 'zoom and filter' process can be understood as a collaborative effort between user and system. By adapting this concept, we end up with an iterative segmentation process which reflects the described interplay between user and system. Thus, the detailed steps of the workflow shown in Figure 8.1 are:

1. The system generates an intermediate random walker segmentation using seed points defined in the previous iterations.

2. The system analyzes the probability field in order to detect ambiguous regions.

3. To provide an overview, all detected regions are displayed in a table and are highlighted in the 3D multiplanar view as well as on the slice views.

4. The following steps are repeated until the user decides to re-run the random walker with modified input parameters (seeds):
   - The user focuses on a particular region by selecting it in the table widget: The multiplanar view and the slice views are adjusted to bring the selected region into focus.
   - For further inspection the user may decide to zoom onto a selected region and locally refine the segmentation.

8.3.1 Random Walker Parametrization

Since the first step of our workflow relies on a proper parametrization of the random walker algorithm, some background information is required to fully understand our design considerations. The random walker algorithm as formulated by Grady (2006) operates on a weighted graph, whose edge weights define the probability that the random walker chooses a certain neighbor node of its current location in the next step. For representing the structure of the volume to be segmented, we follow Grady's formulation of mapping each voxel to a node with edges representing its 6-neighborhood. To define the weight w_{ij} of the edge connecting the nodes i and j, a Gaussian function is used based on the difference between the intensities g_i and g_j of the corresponding voxels:

$$w_{ij} = \exp(-\beta(g_i - g_j)^2) \tag{8.1}$$

The only free parameter β can be interpreted as the permeability of boundaries: increasing β causes a larger decrease of edge weights within areas of high gradient magnitude than within more homogeneous regions. However, according to our experience, this parameter is rather insensitive: A value of around 4000 gave reliable results for data sets of different sizes and modalities. In our experiments, solely the segmentation of very thin structures, which is an application the random walker algorithm is less suited for, required an adjustment of β to values between 8000 and 16000. While the output of the random walker are probability values, the user also inputs probabilities represented by seed nodes. This is done by assigning probability values of 0.0 and 1.0 to background and foreground seeds. Based on this input, the random walker solution assigns to each unseeded node the probability with which a random walker starting from this node reaches one of the foreground seeds. Due to this reason, the probabilities are also commonly regarded as foreground membership scores.

Transfer Function Integration

Although achieving a highly accurate classification result through a transfer function is often a cumbersome process or might even be impossible, transfer functions have the significant advantage of immediate feedback over most segmentation approaches. Since modifications to a transfer function can usually be reflected in the volume rendering in real-time, they are well-suited for volume exploration. In our system, we use transfer functions for enabling the user to provide a rough pre-classification of the volume, which is then incorporated in the segmentation process. Exploiting such a pre-classification has the potential to improve the efficiency of the workflow, since the user may be able to quickly provide additional information about the region of interest, which might help the system to generate the desired result.

For incorporating an arbitrary transfer function in the segmentation process, we modify the edge weight w_{ij} by adding a term representing the distance between the optical properties assigned to the voxels corresponding to nodes i and j. In the following definition, $\vec{f}(i)$ denotes the color that is assigned to node i, abstracting from the actual domain of a specific transfer function:

$$w_{ij} = (1-\alpha)\,\exp(-\beta(g_i - g_j)^2) + \alpha\,\exp(-\beta(\vec{f}(i) - \vec{f}(j))^2) \tag{8.2}$$

We limit the weighting factor α to the range $[0, 0.5]$, in order to ensure that the obtained segmentation is always based on the original data and not solely derived from the user-provided classification.

8.3.2 Uncertainty Detection

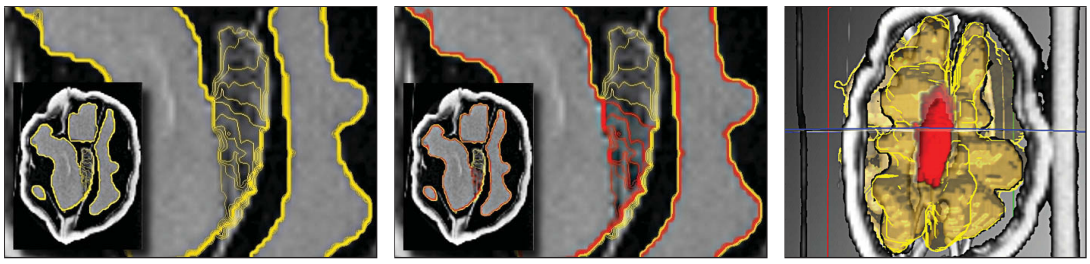
The ability to reliably detect uncertainty in the probabilistic segmentation result is crucial for our system. First of all, since even to obvious background or foreground parts no clear 0.0 or 1.0 probabilities are assigned, we perform a thresholding in order to suppress these obvious parts and thus extract the suspicious areas. The used thresholds can be set interactively; however, based on our experience with CT and MRI data, we have identified 0.2 to 0.8 as the probability range containing all uncertain regions. All probabilities lying outside this range can be classified as certain. Choosing a larger threshold range would be insufficient, because it would enclose certain regions, e.g., 0.1 is often assigned to obvious background parts that are not close to a seed.

The thus thresholded probability distribution usually still contains large parts of clearly classified boundaries. This is caused by the fact that due to the partial volume effect boundaries in scanned data sets do not appear as sharp edges but as rather smooth transitions. Therefore, instead of a hard drop-off we usually find a probability profile resembling the boundary intensity profile in the data set. Applying an erosion operation to the filtered probability field could in principle remove these boundaries, however at the risk of accidentally pruning smaller uncertain structures. Instead, we apply a gradient length threshold to the probability field, which we derive from the user-specified maximum expected width d_b of certain boundaries: assuming a linear probability profile across the boundary, the maximal tolerable gradient magnitude is defined as $(0.8 - 0.2)\,/d_b$. Such an edge filter works much more reliably on the probability field than in the data domain. This is due to the inherently smooth nature of the probability field, which is given, since each unseeded voxel represents the weighted average of its neighbors' probabilities (Grady, 2006). After this boundary suppression has been performed, we obtain the relevant areas of uncertainty. To be able to effectively guide the user through the distribution of uncertainty, we perform a connected component analysis yielding the coherent regions of uncertainty finally presented to the user.

To further support the user when processing the identified uncertain regions, we display them in the table widget sorted by their importance. To estimate this importance of an uncertain region R, we introduce a measure for *ambiguity* $amb(R)$, that takes into account both the size of the region and its probability distribution p:

$$amb(R) := \sum_{v \in R} (1 - 2 \cdot |p(v) - 0.5|) \tag{8.3}$$

The term $1 - 2 \cdot |p(v) - 0.5|$ specifies the uncertainty of a voxel v, since it is inversely



(a) (b) (c)

Figure 8.4: To convey the uncertainty of the segmentation results, we employ different uncertainty visualization techniques. In 2D, we propose the use of uncertainty isolines which depict the uncertainty inherently through the inter-line distance (a). These isolines can also be subject to a two-color mapping (b). In 3D, where an overview depiction is demanded, we use opacity modulation together with contour emphasized surfaces (c).

proportional to the distance of $p(v)$ to the most ambiguous probability value 0.5. Thus, the user can process the uncertain regions based on this order as displayed in the table widget.

8.3.3 Refining the Segmentation

In the usual case, ambiguity is caused by an insufficient number of seed points in the neighborhood. This is characterized by an uncertain area crossing a boundary, which in principle should provide sufficient contrast for a clear classification. This type of ambiguity can be efficiently solved by adding a few more seeds for both the fore- and background segment. Alternatively, when adding seed points one by another is not sufficient, the user may define all seed points on the edge directly by using a geometry widget. One way to perform this is to use a curve editor, which supports the user when fitting a curve to the boundary. Based on this curve multiple seed points are automatically set.

8.4 Uncertainty Visualization

To better guide the user's attention to the detected regions having a high uncertainty, we have tested different uncertainty visualization techniques. Thus, we are able to provide a meaningful emphasis of the detected uncertain regions, while also conveying the overall quality of the final result. Based on the multiple linked view setup of our system, we depict uncertainty in both the 2D as well as the 3D view.

While 2D slice views are well-suited for precise inspections, 3D visualizations have the advantage that they provide an expressive overview of structures. To achieve meaningful uncertainty visualizations, we take these qualities into account. Thus, the 2D uncertainty visualization should be able to depict all relevant details regarding the uncertainty, while the 3D view should provide a meaningful overview of the uncertain regions.

8.4.1 2D Techniques

To depict the uncertainty in the 2D slice views, we propose the usage of uncertainty isolines, as shown in Figure 8.4a. These isolines are directly derived by applying image processing operators to the probability distribution of the current slice. To reduce cluttering of the original data augmented by the isolines, we first mask regions with probability values outside the defined uncertainty range between 0.2 and 0.8, then apply a quantization operator, and finally perform an edge detection on the quantized probability field. Thus, all regions with uncertain probability values are covered by isolines, while isolines at well-classified boundaries collapse to a sharp contour, since these boundaries are characterized by a strong drop of probability values. When choosing the number of quantization steps, we have to make a trade-off between occlusion resulting from a too dense set of isolines, and loss of focus due to too few isolines. According to our experience, a number of 10 till 20 isolines seems to be appropriate.

Besides the actual uncertainty visualization strategy, the color mapping used to depict the uncertainty also has to be chosen carefully. In our initial implementation, we had integrated a two-color mapping based on two distinct hues (see Figure 8.4b). While lightness variations of the one hue were used for all probability values below 0.5, shades of the other hue were used for all probability values above 0.5. However, since changes in hue are perceived pre-attentively (Ware, 2000), this led to an inherent emphasis of one potential boundary, along the 0.5 uncertainty isoline. Since there is no evidence that this is the most probable boundary, we have omitted this two-color mapping as shown in Figure 8.4a.

8.4.2 3D Techniques

While already a substantial amount of research has been conducted to improve uncertainty visualization in 3D, the existing techniques are only applicable up to a certain extent in our case. Especially those techniques dedicated to visualizing surface uncertainty (Pang et al., 1996; Grigoryan and Rheingans, 2004) were not very well suited, since we wanted to avoid introducing the notion of a surface in uncertain

regions. Instead, in order to emphasize the volumetric nature of the uncertain region, uncertain volume visualization techniques are more adequate. We exploit a volumetric uncertainty visualization, which is inspired by the transparency approach described by MacEachren et al. (2005). Therefore, we apply an opacity transfer function to the uncertain regions, which sets the hue to constant and modulates the transparency based on the probability values. This leads to a blobby appearance, which fades out towards the areas of higher uncertainty. In Figure 8.4c we show the outcome of this technique in combination with a surface representation of the certain segment boundaries and a multiplanar view providing contextual information. As it can be seen, transparency is exploited when visualizing the segmentation boundary in order to deal with the occlusion problem. However, since transparency affects the shape perception of this boundary (Interrante et al., 1997), we have augmented the visualization by adding contour edges, which we have derived based on a Sobel filter applied to the corresponding depth values. The positive effects of such contour enhancement have also been described by other authors, when dealing with semi-transparent surfaces (Fischer et al., 2005). While the transparency modulation also influences shape perception, this is not an issue in our case, since the 3D view is only intended to provide an overview, while the 2D view is used to explore structures in more detail.

8.5 Integration into Voreen

The presented volume segmentation workflow has been realized in the Voreen framework by combining custom implementations of the random walker algorithm and the described segmentation analysis with standard components for the rendering and seed point definition.

8.5.1 Data-flow Network

Figure 8.5 shows a simplified data-flow network that creates the 3D view and one of the 2D slice views of the user interface presented in Figure 8.3. The sub-network on the right generates a rendering of the currently selected slice and overlays it with the user-defined foreground and background seeds. The result is then combined with uncertainty isolines of the current segmentation, which are generated by the `SegmentationOverlay2D` processor as described in Subsection 8.4.1, and finally passed to the `QuadView` processor that maps it to one of the quarters of the screen. In the complete network, the slice viewing pipeline is replicated for each of the major axes of the volume.

The `RandomWalker` processor computes the random walker solution using the foreground and background seed points it receives from the `ROI Foreground` and `ROI Background` processors, respectively. The probabilistic segmentation result is passed to the `RandomWalkerAnalyzer` for the uncertainty analysis described in Subsection 8.3.2. The resulting uncertainty volume along with the binary random walker segmentation is passed to the `MultiVolumeRaycaster` that generates the 3D uncertainty visualization introduced in Subsection 8.4.2.

The visualization components of the proposed system have been implemented using OpenGL and GLSL achieving interactive performance. The basic filtering operations as well as the connected component analysis performed on the probability field last less than 0.5s for volumes of size 256^3. Therefore, the main challenge for an interactive workflow was to realize an efficient implementation of the random walker method, which we discuss in the following subsection.

8.5.2 Random Walker Implementation

Grady (2006) exploited the equivalence between the random walker problem and the Dirichlet problem from potential theory to transform the computation of the random walker probabilities into the solution of a system of linear equations. Though this system is large, containing one equation per unseeded voxel, the corresponding matrix is also sparse and positive definite, allowing the application of efficient iterative solvers like conjugate gradients. However, interactive speeds on the CPU were only achieved for medium-sized 2D images, which matched our experiences with CPU-based conjugate gradient solvers. The OpenGL-based GPU implementation of the Random Walker presented in (Grady et al., 2005), on the contrary, enabled interactive results for 3D volumes of sizes up to 128^3.

Therefore, it was obvious that we had to exploit the processing power of the GPU for allowing an interactive workflow. We favored an OpenCL-based implementation of the conjugate gradient method over an GLSL-based approach due to the straightforward programming model, for instance not requiring the encoding of data into textures. Since the sparse matrix representing the random walker equation system can be reduced to three elements per row (Grady et al., 2005) and the conjugate gradient method requires three additional vectors for temporary storage, six values need to be stored in the GPU memory for each unseeded voxel. Therefore, our single-precision (4 byte floats) implementation requires 24 bytes per unseeded voxel to be stored. On our system equipped with an NVIDIA GeForce GTX 285 graphics board with 1 GB graphics memory we were able to process volumes with approximate maximal dimensions of 300^3, which corresponds to a memory consumption of about 633 MB.

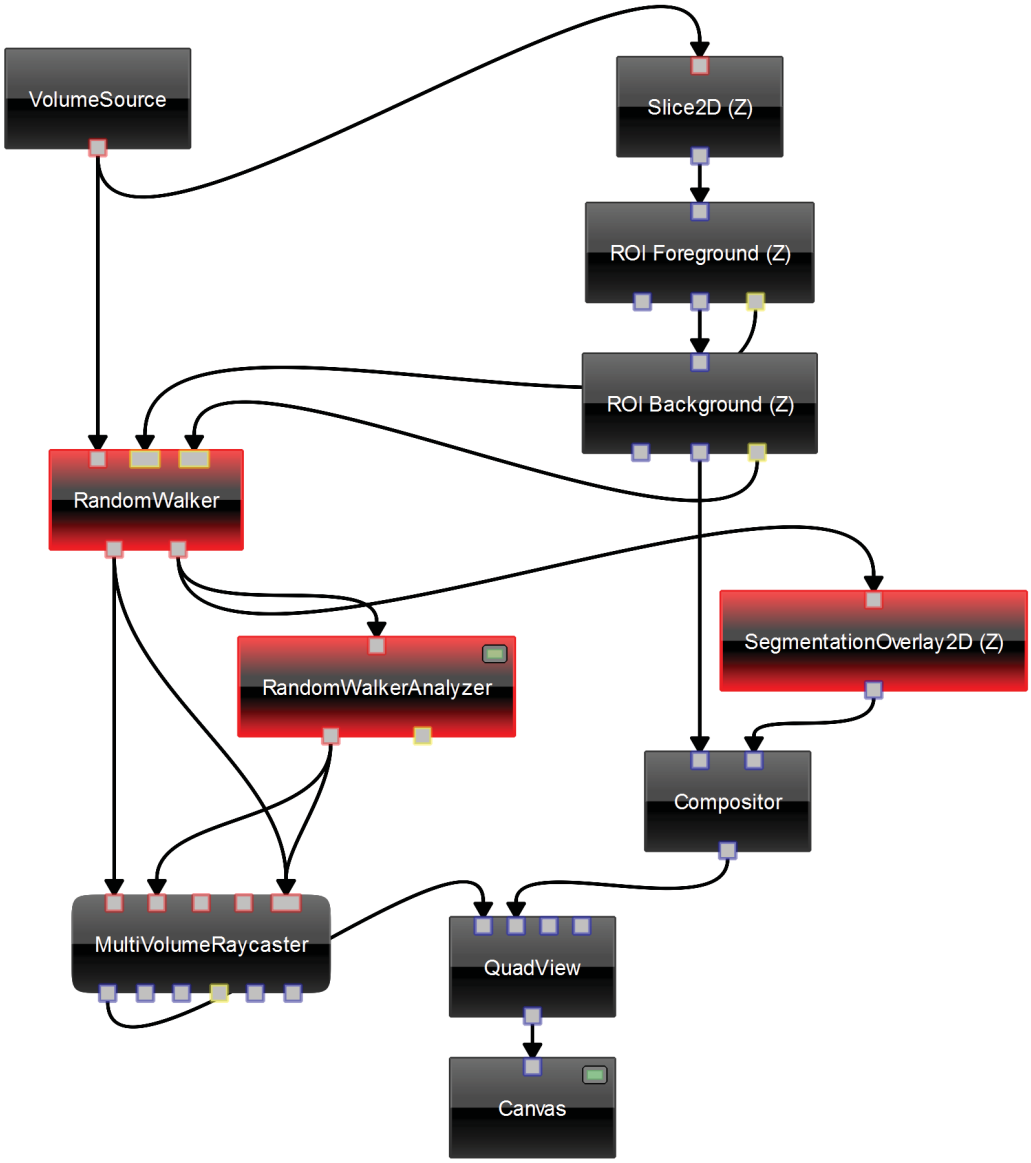


Figure 8.5: Simplified Voreen data-flow network implementing the proposed guided volume segmentation workflow. The sub-network on the right creates a 2D slice view, overlayed with uncertainty isolines, and allows the definition of seed points. The random walker results are additionally visualized in 3D by the MultiVolumeRaycaster.

Presumably, the remaining graphics memory is occupied by OpenGL resources, such as volume textures. By storing the matrix with half-precision, however, we were able to elevate this limit to approximately 350^3, with negligible impact on the runtime. When initializing our conjugate gradient solver with the result of the previous iteration of the workflow, one computation of the random walker solution for a 256^3 volume takes about 4-6 seconds. The first iteration of the system after the initial seed placement usually requires 20-30 seconds.

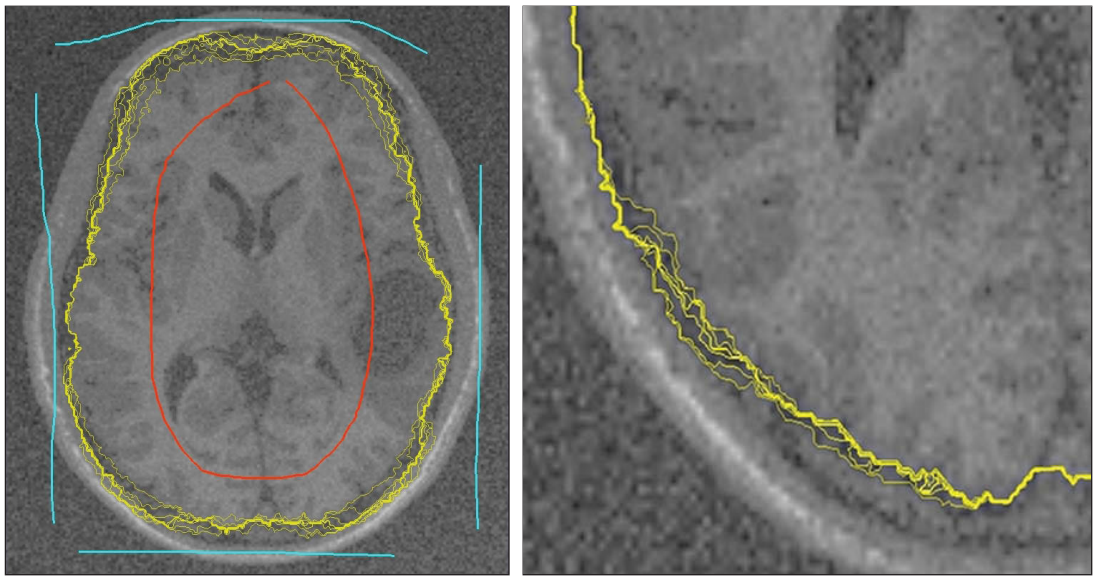
8.6 Results

To judge the benefits of the proposed system, we have performed an informal as well as a formal user study. Furthermore, we have quantitatively analyzed the obtained segmentation results, which we have compared to ground truth segmentations.

8.6.1 User Study

The informal user study was two-fold. In the first test, we have asked three users to segment a data set by using our system and a comparison system which is based on region growing. The second part of the study has been conducted as an informal interview with a medical expert.

Among the three users participating in the informal study where two male and one female, one of these persons had previous experience with volume segmentation techniques. Each of these users had to segment the seed of the walnut data set shown in Figure 8.3. This data set has been chosen, since besides several interior regions, which can be easily segmented, also several ambiguous regions occur, which suffer from a high degree of uncertainty. For some of these regions it is even difficult for a human observer to spot the most likely boundary. We have asked two of the users to first segment the walnut's interior by using our system and afterwards by using the region growing approach. The third user had to perform the segmentation tasks in the reverse order. For both approaches we gave a very short introduction, which took less than a minute. All users needed less than 5 iterations of the workflow when segmenting the interior with the proposed system, which took them between 3-5 minutes. After the tests have been completed, all users stated that our underlying workflow has been perceived as very helpful. Especially the user who had prior experience with volume segmentation appreciated the ease of use as well as the quality of the result. While the users needed roughly the same time to perform the task with the region-growing technique, they could not achieve comparable results.



(a) (b)

Figure 8.6: Typical workflow of the conducted user study. (a) shows the initial seed placement on an axial slice of the MRI scan along with a large uncertain layer enclosing the brain, while in (b) the system has zoomed onto a smaller uncertain region during a subsequent iteration.

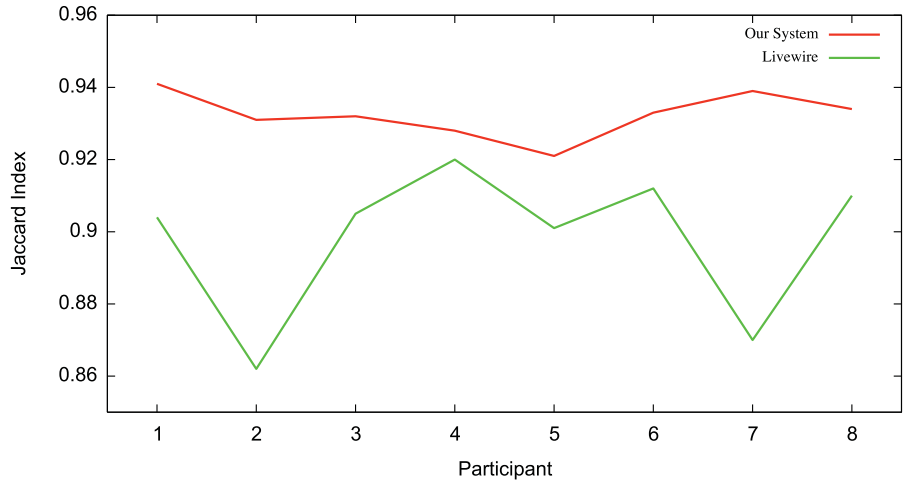


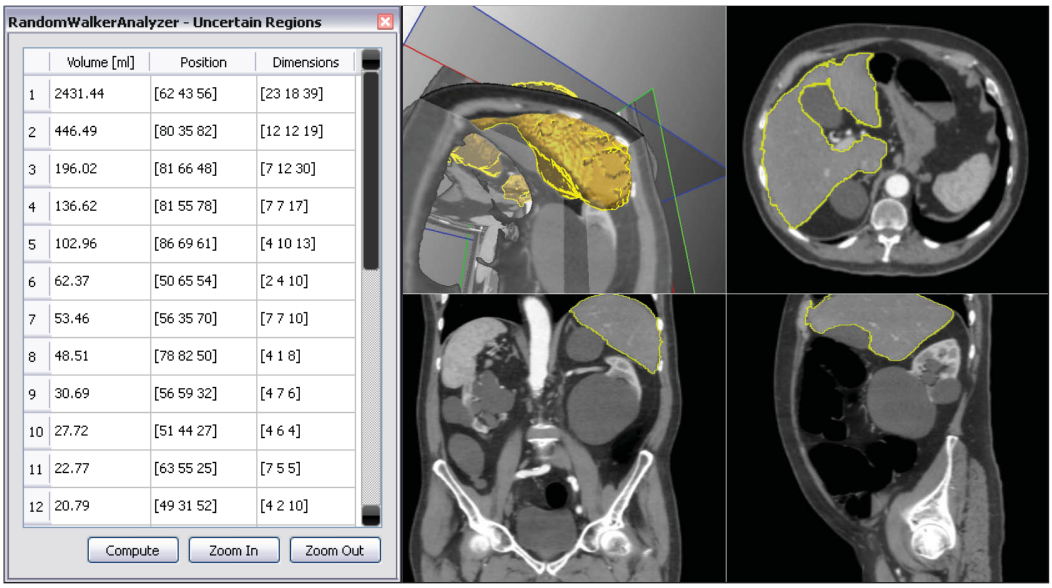
Figure 8.7: Quantitative evaluation of the user study showing Jaccard indices between the user-created segmentations and the ground truth.

Furthermore, it was not clear to the users when a sufficient quality of the result was achieved, and thus their confidence in the segmentation quality was lower.

The informal interview with the medical expert led to comparable results. Similar to the user, who had previous segmentation experience, the medical expert especially emphasized the importance of the workflow. Additionally, the conveying of the reliability of the segmentation was appreciated. Based on the interview and the shown demonstration, we could arouse interest, and it is planned that the system is used for medical research in the future.

Based on this initial feedback we have then conducted a formal user study involving quantitative analysis of the segmentation results achieved by the participants. As a practical application case we selected a data set from the *TumorSim* database (Prastawa et al., 2005), which provides simulated brain tumor MRI scans along with ground truth segmentations. Since these data sets are designed for a realistic validation of brain segmentation algorithms, they suffer from similar deficiencies as real brain MRI scans, especially high noise and bias (see Figure 8.6). The participants were asked to segment the complete brain including the tumor, first with our proposed system and afterwards with the livewire approach (Hastreiter and Ertl, 1998). Out of the available MRI modalities we have chosen the T1 data set (256x256x175), in which the brain exhibits the least contrast to the surrounding tissue and which therefore appears to be the most challenging segmentation task. After a short introduction to both systems the two female and six male participants were given at most 15 min for segmenting the brain with each of the systems. With the random walker approach only one user took advantage of the full time limit, while the remaining participants needed between 8 and 13 minutes. During the initial seed placement phase, most users marked the brain as foreground and the skull as background, but did not place any seeds in the gap between brain and skull. Therefore, after the first iteration, the system typically detected a large layer enclosing the brain as uncertain, as shown in Figure 8.6a. After additional background seeds had been placed in this gap, it usually decayed into several smaller uncertain regions, which were then worked through by the users during subsequent iterations. Within the livewire system, the user has to mark the contour of the region to be segmented slice-by-slice. It supports the user by offering contour snapping based on intensity differences and it also provides interpolation of contours across slices. Due to the limited amount of time, the participants were able to process at most 30 out of the approx. 150 slices actually containing brain tissue and had therefore to interpolate contours for the intermediate, unprocessed slices. Four participants took the full 15 minutes, the remaining ones needed more than 10 minutes.

When requested to rate the correctness of the segmentations generated with both



(a) **(b)**

Figure 8.8: The proposed system applied to abdominal CT data. The images show the representations generated when segmenting the liver: the table displaying the initial uncertain regions (a) and the final result as shown by the system (b).

approaches, most participants showed slightly higher confidence in the correctness of their random walker segmentation. To be able to judge the accuracy of the segmentation, we have further performed a quantitative analysis. Therefore, we have computed the Jaccard index (Shattuck et al., 2009) between the user-generated segmentations and the ground truth. The Jaccard index measures the similarity between two sample sets A and B and is defined as the size of their intersection divided by the size of their union:

$$J(A,B) := \frac{|A \cap B|}{|A \cup B|} \quad \in [0;1] \tag{8.4}$$

As shown in Figure 8.7, all participants achieved significantly more accurate results with the proposed system than with the livewire technique. Furthermore, all users were able to achieve a rather accurate segmentation with our approach, while higher variance occurred when comparing the livewire segmentations.

To evaluate the effectiveness of the different system components, we have asked the users to estimate the risk of overlooking critical segmentation errors. Therefore,

they had to rate this risk in a post-questionnaire, under four different assumptions: first, when using the proposed system; second, when only using the user guidance; third, when only using the uncertainty visualization; and fourth, when using no user guidance and no uncertainty visualization. On a scale from 0 to 5, where 5 meant to miss critical errors for sure, all users have rated the proposed system with a score smaller or equal to 2. In contrast, all participants have assessed the absence of one of the components with a score of 3 or higher, and four of them have rated the risk as 5 for a system lacking both components. Finally, the participants were asked to judge which system would allow them to obtain the most accurate segmentation with unlimited amount of time. While three participants rated both systems equally in that regard, the remaining users favored the proposed system over the livewire approach. Being asked for the reasons, most of them stated that though they consider it in principle possible to create a perfect segmentation with the livewire technique, the large amount of work required would inevitably cause a human to make mistakes. Two users were generally skeptical about their ability to precisely sketch complex contours and therefore favored the random walker system, which only required them to place seeds close to a boundary.

8.6.2 Result Comparison

We have applied our system to two application cases: Abdominal CT and brain MRI data. Figure 8.8 shows how the system has been used to segment a liver from an abdominal CT data set. As Figure 8.8a shows, initially a large number of uncertain regions has been detected. After reducing the number of regions during approximately 5 minutes in 5 iterations, the results shown in Figure 8.8b have been achieved.

The results in Figure 8.9 show the outcome when applying the system to the T2 modality of the TumorSim MRI data set. Due to the high contrast of the brain boundary in this modality, we were able to obtain a high-quality segmentation (Jaccard: 0.993) after only three iterations and 2 minutes. For a more challenging segmentation task, we applied our technique to the T1 modality (Case 2) of the brain MRI data from the 2010 Visualization Contest (see Figure 8.10), which also provides ground truth for the brain as well as the tumor. It took us 15 minutes to extract the brain (Jaccard: 0.953) and 5 minutes to segment the tumor (Jaccard: 0.937).

8.6.3 Limitations

A major drawback of the proposed system is the limited applicability of the random walker technique to thin structures. With the proposed technique, we were able to

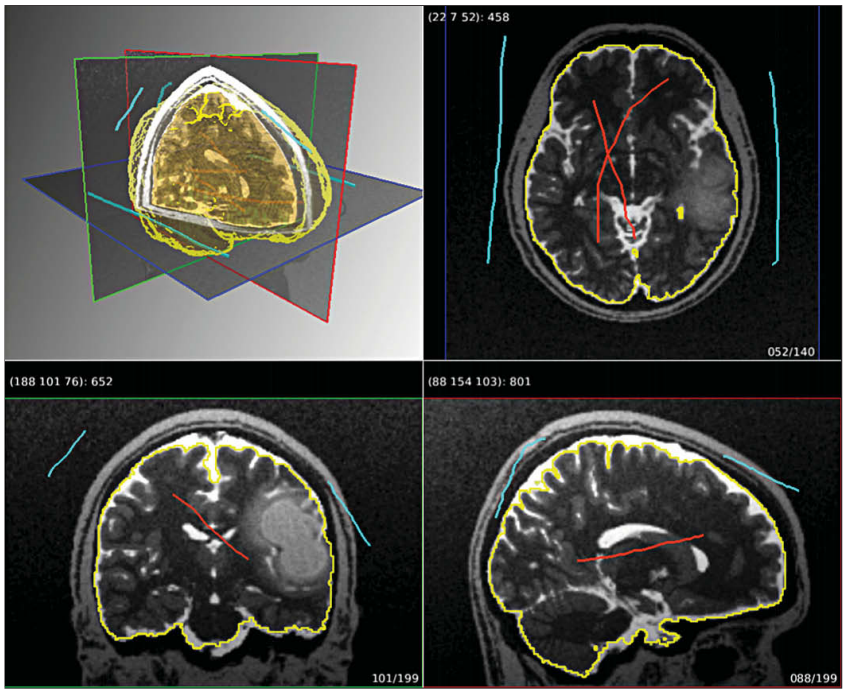
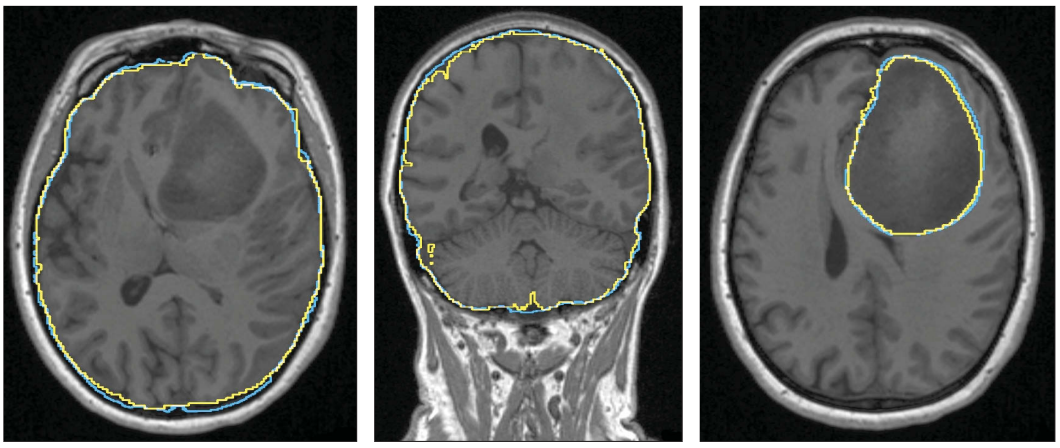


Figure 8.9: The results of using the proposed system on simulated brain MRI T2 data from the TumorSim database (Jaccard: 0.993).



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

Figure 8.10: Application of the proposed system to a T1 brain MRI scan from the 2010 Visualization Contest. (a) and (b) show axial and coronal slices, overlayed with the obtained brain segmentation (yellow) as well as the ground truth (blue). (c) displays the result of the tumor segmentation. Jaccard brain: 0.953, Jaccard tumor: 0.937.

easily segment longitudinal structures with a certain thickness, such as the spinal cord in the MRI data sets or the aorta in contrast-enhanced CT. However, segmenting the thin brain vessels in the MRI scans was only possible by an excessive placement of seeds. Nevertheless, according to our experience this is an inherent limitation of many seed-based segmentation approaches.

8.7 Summary

We have presented an interactive system based on the random walker technique which allows the user to generate reliable segmentations by incorporating uncertainty information. The system is based on a carefully designed iterative workflow, which exploits the strengths of a computer system and the human user by establishing a tightly coupled interaction between these two. Within the proposed workflow, uncertain regions of the segmentation are automatically detected and iteratively presented to the user for further refinement. In contrast to previous techniques, uncertainty visualization is interweaved within the whole workflow, and thus the user is always aware of the reliability of the segmentation achieved so far. To be able to communicate with the system, we support the user by allowing to draw rough strokes and to use other more precise interaction widgets. Based on the thus defined input, we directly parametrize the exploited random walker segmentation algorithm, which in comparison to other segmentation editing approaches has the advantage that we directly work on the original data, instead of modifying the resulting segmentation. Furthermore, we could show how to exploit the OpenCL programming API and thus the vast processing power of modern GPUs in order to realize a random walker implementation that is fast enough to support the proposed workflow. To evaluate the usefulness of the presented approaches, we have conducted user studies. The results of these tests indicate that the presented system allows fast and accurate segmentation by supporting user confidence regarding the results.

In the future we see several possibilities to conduct research for extending the presented system. A stronger adaptation of the random walker technique itself to certain application cases might prove valuable. For instance, it should be evaluated whether more advanced edge weight definitions could improve the applicability of the system to the segmentation of very thin structures. Furthermore, an in-depth analysis should be conducted in order to analyze which transfer function spaces can be optimally combined with the presented approach. In particular, more sophisticated classifiers, such as for instance shape-based (Sato et al., 2000) or occlusion-based classifiers (Correa and Ma, 2009), should be investigated.

Chapter 9

Conclusions

Although volume rendering is established as a powerful tool for the interactive analysis of volumetric data, the efficient and reliable detection of features of interest remains a challenging task. Usually, the user has the choice between an ad-hoc definition of a transfer function, which classifies materials based on their intensity value, and an elaborate pre-segmentation of the data set, which explicitly groups voxels into objects. While the identification of features through material classification is imprecise or even impossible in case of overlapping intensity ranges, volume classification plays a vital role in the interactive exploration of data sets, since changes to the transfer function can be immediately reflected in the resulting rendering. The precision of a volume segmentation, on the other hand, is specifically required for quantitative analysis, such as size measurements.

In this dissertation, we have proposed three novel volumetric feature detection techniques, which are located at different positions within the spectrum between volume classification and segmentation (compare Figure 9.1). The *LH classification* approach presented in Chapter 5 alleviates the often cumbersome process of transfer function specification, which is especially hampered by the unintuitive relation between the transfer function and the volume rendered image. By exploiting the LH transfer function space we were able to develop a semi-automatic classification system for the detection of boundaries in volume data, which completely shields the user from the transfer function domain and instead allows the extraction of features of interest by directly marking them in the volume rendered image. While we have demonstrated the robustness of the LH classification when applied to CT data, it might be worth investigating whether its applicability to imaging modalities with less pronounced boundaries, such as MRI and PET data sets, can be improved by exploiting more advanced derivative reconstruction schemes or an adaptation of the boundary model.

The *shape-based transfer function* approach presented in Chapter 6 blurs the border between classification and segmentation techniques. In contrast to previous

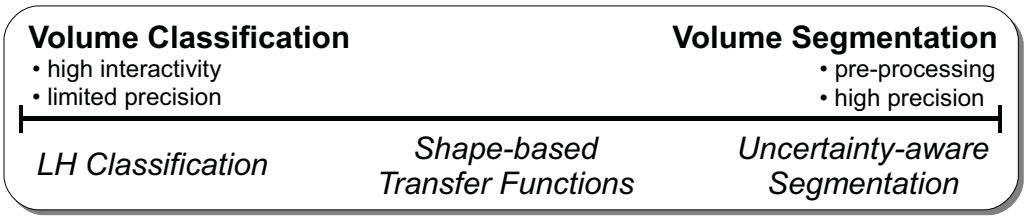


Figure 9.1: Location of the presented techniques in the feature detection spectrum.

approaches towards shape classification in volume data, which perform a voxel level classification involving the interpretation of complex histograms, the proposed technique decomposes the volume into a manageable set of shape-classified structures and offers an intuitive interface for the assignment of optical properties to these. A principal limitation of the approach is its dependency on a sufficiently precise pre-segmentation as basis for the volume decomposition. However, for volumetric scans with contrast agents, which is a typical use-case in the medical domain, obtaining such a rough pre-segmentation is usually possible with little effort by thresholding. A fruitful direction for future research might be the combination of our shape classifiers with existing size-based (Correa and Ma, 2008; Hadwiger et al., 2008) or texture-based (Caban and Rheingans, 2008) volume classification schemes, which could further improve the classification robustness.

The *uncertainty-aware volume segmentation* system exposed in Chapter 8 specifically focuses on the minimization of uncertainty in the resulting segmentation, which is achieved by a tight interplay between the user and the system. In an iterative process, the system continuously assesses uncertainty of a probabilistic segmentation obtained from the random walker algorithm. The user's attention is directed to regions with high ambiguity to support the correction of potential mis-segmentations. This approach not only reduces the risk of critical segmentation errors, but also improves the user's confidence in the correctness of the obtained result, since information about the segmentation's reliability is constantly conveyed to the user. The efficiency of the proposed workflow, which has been proven in user studies, has been achieved by incorporating classification information in the segmentation process as well as an OpenCL implementation of the random walker algorithm that significantly reduces computation times. A major limitation of our system arises from the limited applicability of the random walker technique to the segmentation of thin structures, such as blood vessels. Therefore, the development of more advanced edge weight definition schemes specifically tailored to such structures might provide a valuable extension to the presented system.

Bibliography

Bajaj, C. L., Pascucci, V., and Schikore, D. R. (1997). The contour spectrum. In *VIS '97: Proceedings of the 8th Conference on Visualization*, pages 167–173.

Bartz, D., Mayer, D., Fischer, J., Ley, S., Rio, A. d., Thust, S., Heussel, C. P., Kauczor, H.-U., and Strasser, W. (2003). Hybrid segmentation and exploration of the human lungs. In *VIS '03: Proceedings of IEEE Visualization 2003*, pages 177–184.

Bartz, D. and Meißner, M. (1999). Voxels versus polygons: A comparative approach for volume graphics. In *Proceedings of Volume Graphics 1999*, pages 33–48.

Ben-Zadok, N., Riklin-Raviv, T., and Kiryati, N. (2009). Interactive level set segmentation for image-guided therapy. In *ISBI '09: Proceedings of the Sixth IEEE International Symposium on Biomedical Imaging*, pages 1079–1082.

Binford, T. (1971). Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control*.

Blum, H. (1973). Biological shape and visual science. *Theoretical Biology*, 38:205–287.

Boykov, Y. and Kolmogorov, V. (2000). Interactive organ segmentation using graph cuts. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 276–286.

Boykov, Y. and Kolmogorov, V. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV '01: Proceedings of the Eighth IEEE International Conference on Computer Vision*, pages 105–112.

Boykov, Y. and Kolmogorov, V. (2003). Computing geodesics and minimal surfaces via graph cuts. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 26–33.

Caban, J. J. and Rheingans, P. (2008). Texture-based transfer functions for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1364–1371.

Chen, H.-L. J., Samavati, F. F., Sousa, M. C., and Mitchell, J. R. (2006). Sketch-based volumetric seeded region growing. In *Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 123–129.

Chen, Z., Qiu, T., and Ruan, S. (2008). A brain tissue segmentation approach integrating fuzzy information into level set method. In *ICAL '08: Proceedings of the IEEE International Conference on Automation and Logistics 2008*, pages 1216 – 1221.

Cornea, N. D., Silver, D., and Min, P. (2007). Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548.

Cornea, N. D., Silver, D., Yuan, X., and Balasubramanian, R. (2005). Computing hierarchical curve-skeletons of 3D objects. *The Visual Computer*, 21(11):945–955.

Correa, C. D. and Ma, K.-L. (2008). Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380–1387.

Correa, C. D. and Ma, K.-L. (2009). The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1465–1472.

Correa, C. D. and Silver, D. (2005). Dataset traversal with motion-controlled transfer functions. In *VIS '05: Proceedings of IEEE Visualization 2005*, pages 359–366.

Cremers, D., Fluck, O., Rousson, M., and Aharon, S. (2007). A probabilistic level set formulation for interactive organ segmentation. *Medical Imaging 2007: Image Processing*, 6512(1):120–129.

Cullip, T. J. and Neumann, U. (1994). Accelerating volume reconstruction with 3d texture hardware. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

Diepenbrock, S., Praßni, J.-S., Lindemann, F., Bothe, H.-W., and Ropinski, T. (2010). Pre-operative planning of brain tumor resections. In *IEEE Visualization Contest 2010*. Winning Entry.

Diepenbrock, S., Praßni, J.-S., Lindemann, F., Bothe, H.-W., and Ropinski, T. (2011). 2010 IEEE Visualization Contest winner: Interactive planning for brain tumor resections. *IEEE Computer Graphics and Applications*, 31(5):6–13.

Djurcilov, S., Kim, K., Lermusiaux, P. F. J., and Pang, A. (2001). Volume rendering data with uncertainty information. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 243–252.

Fischer, J., Bartz, D., and Straßer, W. (2005). Illustrative display of hidden iso-surface structures. In *VIS '05: Proceedings of IEEE Visualization 2005*, pages 663–670.

Grady, L. (2006). Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783.

Grady, L., Schiwietz, T., Aharon, S., and Westermann, R. (2005). Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation. In *MICCAI '05: Proceedings of the 8th international conference on Medical image computing and computer-assisted intervention*, pages 773–780.

Grigoryan, G. and Rheingans, P. (2004). Point-based probabilistic surfaces to show surface uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):564–573.

Gu, L. and Peters, T. (2004). Robust 3d organ segmentation using a fast hybrid algorithm. In *CARS '04: Proceedings of the 18th International Congress on Computer Assisted Radiology and Surgery*, volume 1268, pages 69 – 74.

Hadwiger, M., Laura, F., Rezk-Salama, C., Höllt, T., Geier, G., and Pabel, T. (2008). Interactive volume exploration for feature detection and quantification in industrial CT data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1507–1514.

Har-Peled, S. (2001). A practical approach for computing the diameter of a point set. In *SCG '01: Proceedings of the 17th annual symposium on Computational geometry*, pages 177–186.

Hastreiter, P. and Ertl, T. (1998). Fast and interactive 3d-segmentation of medical volume data. In *Proceedings of Computer Graphics International 1998*, pages 78–85.

He, T., Hong, L., Kaufman, A., and Pfister, H. (1996). Generation of transfer functions with stochastic search techniques. In *VIS '96: Proceedings of the 7th Conference on Visualization*, pages 227–235.

Hibbard, W. and Santek, D. (1989). Interactivity is the key. In *VVS '89: Proceedings of the 1989 Chapel Hill Workshop on Volume Visualization*, pages 39–43.

Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T. L. (2001). Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212.

Hladuvka, J., König, A., and Gröller, E. (2000). Curvature-based transfer functions for direct volume rendering. In Falcidieno, B., editor, *SCCG '00: Proceedings of the Spring Conference on Computer Graphics 2000*, volume 16, pages 58–65.

Huang, R. and Ma, K.-L. (2003). Rgvis: Region growing based techniques for volume visualization. In *Proceedings of the Pacific Conference on Computer Graphics and Applications 2003*, pages 355–363.

Interrante, V., Fuchs, H., and Pizer, S. M. (1997). Conveying the 3d shape of smoothly curving transparent surfaces via texture. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):98–117.

Jiawan, Z., Jizhou, S., Zhigang, S., and Zunce, W. (2003). Moment based transfer function design for volume rendering. In *ICCSA '03: Proceedings of the 2003 international conference on Computational Science and its applications 2003*, pages 266–274.

Johnson, C. (2004). Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17.

Kindlmann, G. and Durkin, J. W. (1998). Semi-automatic generation of transfer functions for direct volume rendering. In *VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 79–86.

Kindlmann, G. and Weinstein, D. (1999). Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *VIS '99: Proceedings of IEEE Visualization 1999*, pages 183–189.

Kindlmann, G., Whitaker, R., Tasdizen, T., and Möller, T. (2003). Curvature-based transfer functions for direct volume rendering: Methods and applications. In *VIS '03: Proceedings of IEEE Visualization 2003*, page 67.

Kniss, J., Kindlmann, G., and Hansen, C. (2002). Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285.

Kniss, J. M., Uitert, R. V., Stephens, A., Li, G.-S., Tasdizen, T., and Hansen, C. (2005). Statistically quantitative volume visualization. In *VIS '05: Proceedings of IEEE Visualization 2005*, pages 287 – 294.

Kohli, P. and Torr, P. H. (2008). Measuring uncertainty in graph cut solutions. *Computer Vision and Image Understanding*, 112(1):30 – 38.

König, A. and Gröller, E. (2001). Mastering transfer function specification by using volumepro technology. In *SCCG '01: Proceedings of the 17th Spring Conference on Computer Graphics*, pages 279–286.

Kontos, D., Wang, Q., Megalooikonomou, V., Maurer, A. H., Knight, L. C., Kantor, S., Simonian, H. P., and Parkman, H. P. (2006). A tool for handling uncertainty in segmenting regions of interest in medical images. *International Journal of Intelligent Systems Technologies and Applications*, 1(3/4):194–210.

Krüger, J. and Westermann, R. (2003). Acceleration techniques for GPU-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization*, pages 38–45.

Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169.

Lundström, C., Ljung, P., Persson, A., and Ynnerman, A. (2007). Uncertainty visualization in medical volume rendering using probabilistic animation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1648–1655.

MacEachren, A. M., Robinson, A., Gardner, S., Murray, R., Gahegan, M., and Hetzler, E. (2005). Visualizing geospatial information uncertainty: What we know and what we need to know. cartography and geographic. *Information Science*, 32:137–160.

Macrini, D., Siddiqi, K., and Dickinson, S. (2008). From skeletons to bone graphs: Medial abstraction for object recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 0:1–8.

Marks, J., Andalman, B., Beardsley, P. A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J., and Shieber, S. (1997). Design galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 389–400.

Marsden, J. E. and Tromba, A. J. (1996). *Vector Calculus*. W.H. Freeman and Company, New York.

Martin, K. and Hoffman, B. (2004). *Mastering CMake: A Cross-Platform Build System, second edition*. Kitware Inc.

Meyer-Spradow, J. (2009). *Interaktive Entwicklung Raycasting-basierter Visualisierungs-Techniken für medizinische Volumen-Daten mit Hilfe von Datenflussnetzwerken*. PhD thesis, Westfälische Wilhelms-Universität Münster.

Meyer-Spradow, J., Ropinski, T., Mensmann, J., and Hinrichs, K. (2009). Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29(6):6–13.

Mortensen, E. N. and Barrett, W. A. (1995). Intelligent scissors for image composition. In *SIGGRAPH '95: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 191–198.

Olabarriaga, S. and Smeulders, A. (2001). Interaction in the segmentation of medical images: A survey. *Medical Image Analysis*, 5(2):127 – 142.

Owada, S., Nielsen, F., and Igarashi, T. (2005). Volume catcher. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pages 111–116.

Pang, A. (2001). Visualizing uncertainty in geo-spatial data. In *Proceedings of the Workshop on the Intersections between Geospatial Information and Information Technology*, pages 1–14.

Pang, A. T., Wittenbrink, C. M., and Lodh, S. K. (1996). Approaches to uncertainty visualization. *The Visual Computer*, 13:370–390.

Patel, D., Haidacher, M., Balabanian, J.-P., and Gröller, M. E. (2009). Moment curves. In *PacificVis '09: Proceedings of the IEEE Pacific Visualization Symposium 2009*, pages 201–208.

Pfister, H., Lorensen, B., Bajaj, C., Kindlmann, G., Schroeder, W., Avila, L., Raghu, K., Machiraju, R., and Lee, J. (2001). The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–22.

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of ACM*, 18(6):311–317.

Pizer, S. M., Gerig, G., Joshi, S. C., and Aylward, S. R. (2003). Multiscale medial shape-based analysis of image objects. *Proceedings of the IEEE*, 91(10):1670–1679.

Praßni, J.-S., Ropinski, T., and Hinrichs, K. (2009). Efficient boundary detection and transfer function generation in direct volume rendering. In *VMV '09: Proceedings of Vision, Modeling, and Visualization 2009*, pages 285–294.

Praßni, J.-S., Ropinski, T., and Hinrichs, K. (2010a). Uncertainty-aware guided volume segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1358–1365.

Praßni, J.-S., Ropinski, T., Mensmann, J., and Hinrichs, K. (2010b). Shape-based transfer functions for volume visualization. In *PacificVis '10: Proceedings of the IEEE Pacific Visualization Symposium 2010*, pages 9–16.

Praßni, J.-S., Storm, K., Ropinski, T., and Tiemann, K. (2010c). Single shot quantification of gas-filled microbubbles with ultrasound. *Eurographics Workshop on Visual Computing for Biology and Medicine*. poster presentation.

Prastawa, M., Bullitt, E., and Gerig, G. (2005). Synthetic ground truth for validation of brain tumor MRI segmentation. In *MICCAI '05: Proceedings of the 8th international conference on Medical image computing and computer-assisted intervention*, pages 26–33.

Reniers, D., Jalba, A., and Telea, A. (2008). Robust classification and analysis of anatomical surfaces using 3D skeletons. In *VCBM '08: Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine 2008*.

Rezk-Salama, C., Keller, M., and Kohlmann, P. (2006). High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):1021–1028.

Rhodes, P. J., Laramee, R. S., Bergeron, R. D., and Sparr, T. M. (2003). Uncertainty visualization methods in isosurface volume rendering. In *Proceedings of Eurographics 2003*, pages 83–88.

Ropinski, T. and Praßni, J.-S. (2010). DIY Vis Applications. *Tutorial at the IEEE Visualization Conference 2010*.

Ropinski, T., Praßni, J.-S., Steinicke, F., and Hinrichs, K. H. (2008). Stroke-based transfer function design. In *Proceedings of the IEEE/EG International Symposium on Volume and Point-Based Graphics 2008*, pages 41–48.

Röttger, S., Guthe, S., Weiskopf, D., Ertl, T., and Straßer, W. (2003). Smart hardware-accelerated volume rendering. In *VISSYM '03: Proceedings of the Symposium on Data Visualisation 2003*, pages 231–238.

Saad, A., Möller, T., and Hamarneh, G. (2010). Probexplorer: Uncertainty-guided exploration and editing of probabilistic medical image segmentation. *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization*, 29(3).

Sabella, P. (1988). A rendering algorithm for visualizing 3d scalar fields. *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 22(4):51–58.

Sato, Y., Westin, C.-F., Bhalerao, A., Nakajima, S., Shiraga, N., Tamura, S., and Kikinis, R. (2000). Tissue classification based on 3d local intensity structures for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180.

Segars, W. P. (2001). *Development and Application of the New Dynamic NURBS-based Cardiac-Torso (NCAT) Phantom*. PhD thesis, University of North Carolina at Chapel Hill.

Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials*. Cambridge University Press, 2 edition.

Shattuck, D. W., Prasad, G., Mirza, M., Narr, K. L., and Toga, A. W. (2009). Online resource for validation of brain segmentation methods. *NeuroImage*, 45(2):431 – 439.

Sherbondy, A., Houston, M., and Napel, S. (2003). Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *VIS '03: Proceedings of IEEE Visualization 2003*, pages 171–176.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages 1996*, pages 336–343.

Takahashi, S., Takeshima, Y., and Fujishiro, I. (2004). Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49.

Tzeng, F.-Y., Lum, E. B., and Ma, K.-L. (2005). An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284.

Šereda, P., Bartroli, A. V., Serlie, I. W. O., and Gerritsen, F. A. (2006a). Visualization of boundaries in volumetric data sets using LH histograms. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):208–218.

Šereda, P., Gerritsen, F. A., and Vilanova, A. (2006b). Mirrored LH histograms for the visualization of material boundaries. In Kobbelt, L. and Kuhlen, T., editors, *VMV '06: Proceedings of Vision, Modeling, and Visualization 2006*, pages 237–244.

Ware, C. (2000). *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc.

Wu, Y. and Qu, H. (2007). Interactive transfer function design based on editing direct volume rendered images. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1027–1040.

Zhu, S. C. and Yuille, A. (1996). Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:884–900.

Acronyms

API	application programming interface
CDF	cumulative distribution function
CPU	central processing unit
CT	computed tomography
DTI	diffusion tensor imaging
DVR	direct volume rendering
EEP	entry-exit points (ray parameters)
FPS	frames per second
GLSL	OpenGL shading language
GPU	graphics processing unit
GUI	graphical user interface
MIP	maximum intensity projection
MRI	magnetic resonance imaging
OpenCL	open computing language
OpenGL	open graphics library
PET	positron emission tomography
PSF	point spread function
ROI	region of interest
TF	transfer function