UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI
MINISTERUL MUNCII, FAMILIEI ŞI PROTECŢIEI SOCIALE
AMPOSDRU

Fondul Social European
POSDRU 2007-2013

Instrumente Structurale
2007-2013

OIPOSDRU

UNIVERSITATEA TEHNICĂ "GHEORGHE ASACHI" DIN IAŞI

UNIVERSITATEA TEHNICĂ
"GHEORGHE ASACHI" DIN IAŞI

Şcoala Doctorală a Facultăţii de
AUTOMATICĂ ŞI CALCULATOARE

VISUALIZATION AND GRAPHICAL PROCESSING OF VOLUME DATA

VIZUALIZAREA ŞI PRELUCRAREA GRAFICĂ A DATELOR VOLUMETRICE

PhD Thesis

(TEZĂ DE DOCTORAT)

Supervisors (Conducători de doctorat):

Prof. dr. **Vasile Manta**

Prof. dr. **Eduard Gröller**

PhD Student (Doctorand):

Eng. **Marius Gavrilescu**

Iaşi, 2011

Teza de doctorat a fost realizată cu sprijinul financiar al proiectului „Burse Doctorale - O Investiţie în Inteligenţă (BRAIN)".

Proiectul „Burse Doctorale - O Investiţie în Inteligenţă (BRAIN)", POSDRU/6/1.5/S/9, ID 6681, este un proiect strategic care are ca obiectiv general „Îmbunătăţirea formării viitorilor cercetători în cadrul ciclului 3 al învăţământului superior - studiile universitare de doctorat - cu impact asupra creşterii atractivităţii şi motivaţiei pentru cariera în cercetare".

Proiect finanţat în perioada 2008 - 2011.

Finanţare proiect: 14.424.856,15 RON

Beneficiar: Universitatea Tehnică "Gheorghe Asachi" din Iaşi

Partener: Universitatea "Vasile Alecsandri" din Bacău

Director proiect: Prof.univ.dr.ing. Carmen TEODOSIU

Responsabil proiect partener: Prof.univ.dr.ing. Gabriel LAZĂR

Acknowledgements

This project was carried out in the framework of a Joint Doctoral Agreement between the "Gheorghe Asachi" Technical University of Iasi, Romania, and the Vienna University of Technology, Austria.

First of all, I would like to extend my sincerest gratitude to Professor Eduard Gröller, head of the VisGroup from the Institute of Computer Graphics and Algorithms, for his unconditional support and help with research throughout the years. None of my accomplishments and experience gained within the field of volume visualization would have been possible without his guidance. I'd also like to thank the members of the VisGroup for creating a positive working atmosphere and for providing help with my work whenever I needed it.

I am also grateful to Professor Vasile Manta for his support, patience and continued assistance with my work, as well as to the teaching staff from the Faculty of Automatic Control and Computer Engineering.

I would like to thank the organizers of the "BRAIN-An Investment in Intelligence" Doctoral Scholarship Program for their efforts, and for providing me and my colleagues with a lucrative opportunity to carry out our research.

Last but not least, I would like to thank my family and friends for their moral support and care.

Contents

1. Introduction

The computer-aided visual representation of information obtained from non-visual data has become increasingly important over recent decades. As this data increases in size and complexity, the use of automated means of increasing sophistication and power is indispensable for the proper extraction, analysis and representation of information. To this extent, the field of visualization involves the generation of images which convey relevant information about data and phenomena. The development of this field is a response to the quasi-exponential progress of data acquisition technologies. The various methods and algorithms developed within visualization allow improved perception and a more efficient representation of models, structures and complex physical phenomena. A field which has gained significant momentum particularly in the past decade is volume visualization. It encompasses methods for the extraction and display of meaningful information from volume data sets using interactive graphics processing and imaging means. Volume visualization has wide applicability in the representation of medical images (such as Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) scans), industrial CT, ultrasound and other such kinds of data. Nowadays, the field has reached a significant degree of maturity, due in no small part to the progress in related hardware. Modern Graphics Processing Units (GPUs) provide the computational power required to handle the large amount of data and the large number of computations usually required by volume rendering applications. Features such as a large amount of video memory, programmable shader units and a high clock frequency means that state-of-the-art volume visualization frameworks are capable of generating high quality renderings and complex interface elements at interactive frame rates.

This thesis presents techniques which aim to illustrate the experience gained by the author within the field of volume visualization, as well as to bring forth contributions to this field. The work developed for the purpose of the thesis addresses multiple issues pertaining to volume visualization, such as the rendering, representation, classification and graphical in-depth analysis of the information contained in volume data sets. The covered topics include volume rendering algorithms, classification by means of transfer functions, the accentuation of features in volume data, the visualization of information from medical data, the assessment of volume rendering parameters and the analysis of parameter behavior for various purposes and types of data.

There are multiple general motivations which drove this work forward. Firstly, it was important to accumulate knowledge and know-how within the field of volume visualization. For this purpose, we consulted an extensive quantity of scientific material from various topics within volume graphics, such as papers, PhD theses, Master theses and books. We have also had extensive collaboration with the Institute of Computer Graphics and Algorithms from the Vienna University of Technology, whose visualization research group have shared their substantial expertise and feedback. Based on the experience gained from these various sources, we developed our own prototyping environment, *UniVolume*, in which we implemented most of the algorithms and techniques described in the thesis. Unless otherwise noted, all illustrations in the thesis are produced using UniVolume, based on our own implementations. We provide detailed descriptions of the fundamentals of volume rendering and visualization, as well as various techniques encountered throughout the relevant literature, which helped push forward the research and documentation effort required for the development of this work. Based on this research and the experience accumulated thereof, we have developed our own contributions, which draw upon the current state of the art in the field. These are each presented in their respective sections of the thesis.

The work was carried out within the *BRAIN - An Investment in Intelligence* Doctoral Scholarship Program, under joint PhD supervision between the Faculty of Automatic Control and Computer Engineering from the "Gheorghe Asachi" Technical University of Iasi, Romania and the Institute of Computer Graphics and Algorithms from the Vienna University of Technology, Austria.

1.1. Objectives

Given the broad and encompassing title of the thesis, we have attempted to cover as much from the field as our experience and available time have allowed. Our aims with this project can be summarized as follows:

- achieve an understanding of volume graphics, including modern volume classification and rendering techniques;

- develop a volume rendering framework for prototyping and the implementation of various volume visualization algorithms;

- implement and improve volume classification methods based on various voxel spaces and find ways to combine transfer functions which operate in these spaces;

- develop ways to highlight the features of structures within volumes (such as contours, ridges, surfaces, etc.);

- provide methods for the visualization of cardiac MRI and for the analysis of cardiac parameters;

- develop methods to analyze the behavior of volume rendering parameters across their respective domains;

- develop and implement augmented interface elements which provide feedback on the behavior of volume rendering parameters.

1.2. The UniVolume prototyping environment

UniVolume is a volume visualization framework currently under development, which aims to offer implementations of state-of-the-art volume visualization techniques, as well as efficient prototyping tools for volume classification, rendering and analysis. The vast majority of the techniques described in this thesis are implemented in UniVolume. Nearly all images in the Figures are also rendered using the tools developed and incorporated into UniVolume.

The framework started as a learning tool, and initially only implemented basic volume rendering algorithms. Since then, it has expanded considerably, and now includes features such as enhanced volume rendering, various visualization packages, multiple/composite classification methods, optimizations, and various other implementations of related algorithms, most of which are described in this text. While still in an early stage of development, the goal is to develop and include a full featured set of volume visualization, analysis and manipulation tools, to serve in the research and development of tools for volume graphics.

1.3. Thesis overview

The thesis is organized into four main parts, each with a dedicated purpose and handling a different topic. In each section we provide brief background information concerning the topic under scrutiny. We then go into in-depth explanations of the methods and algorithms involved. The evaluation of each method proposed is carried out through explanations and images illustrating the results of our work. Aside from this introductory section, the components of this work are as follows:

Chapter 2 provides a frame for the thesis as a whole and explains concepts which are necessary for a better comprehension of other chapters, as well as for integrating them into the context of volume visualization. Various aspects of volume visualization are explored, such as the mathematical theory behind volume rendering, optical models, and the propagation of light through heterogeneous media. A brief overview of the graphics pipeline in GPUs and details regarding the functionality of GPU programming for volume graphics are then presented. We then address the issue of volume rendering and explore mainstream rendering algorithms.

Chapter 3 addresses two significant aspects of volume visualization: the classification of volume data, a stage of the volume rendering pipeline where information of interest is extracted from volume data sets, while other irrelevant data is discarded. The focus is on the use of transfer functions for volume classification. In this context, we explore the use of transfer function domains, defined by voxel properties such as the scalar value, gradient magnitude, the local curvature, spatial distance and visibility. We also provide our own versions or optimizations of these methods. We show how transfer functions which operate in these domains can be combined to achieve flexible approaches for classification.

The second aspect addressed in this chapter is the enhancement of volume features. Thus, we explore the use of outlines and post-render filtering methods to better highlight structures within volume objects. We present our results through explanations and before-and-after image sets.

Chapter 4 deals with the visualization of cardiac MRI data sets. The focus is on the left-ventricular chamber of the heart. Starting from segmented contours, a 3D geometric representation of the myocardium is used to display various cardiac parameters in order to help provide better insight on heart functionality. We develop methods for the separate and concurrent representation of parameters, for multi-dimensional dynamic cardiac data sets. We show how these methods can be used for exploratory, educational and informative purposes within the frame of cardiologic diagnosis and treatment.

Chapter 5 deals with the issue of volume rendering parameters in more generic terms. We develop enhanced user interfaces and methods for improved parameter adjustment, and provide tools for the assessment of variations induced by parameter changes. The methods are based on the image-space evaluation of parameter effects, and allow the development of tools such as graph sliders and local stability maps for transfer functions. We show how the information derived from parameter behavior can be used for assigning values more efficiently, as well as for providing relevant feedback on the characteristics of the parameters across their domains.

A final chapter provides concluding remarks and a brief discussion on the results achieved. It is shortly followed by references and other relevant information.

2. Volume Visualization and Rendering

The discipline of visualization deals with the representation of meaningful information from various types of non-graphical data. Volume visualization in particular consists of methods for the extraction, classification and display of information from a class of data referred to as *volume data sets*, or simply *data sets*. These are collections of scalar values which define a measurable physical property of a region of space (such as density). By employing a variety of sampling, filtering, classification and rendering methods, a visual representation of the objects and media within this region can be generated based on their associated scalar values.

The images most commonly encountered in 3D computer graphics research and art are generated from objects and scenes modeled using polygonal arrays. Objects in conventional computer graphics consist mostly of their outlining surfaces, which are approximated through rectangular or triangular meshes (Mortenson, 1999). This is consistent with the majority of real-world objects, which are opaque to visible light. At the same time, this manner of representing objects is highly scalable, since the level of detail visible in a 3D scene can be adjusted by means of the number of triangles used for the approximation of the features displayed. However, the use of strictly outlined surfaces has the disadvantage that it cannot properly account for "fuzzy" regions of space, where objects either do not have a clearly-defined surface or such a surface cannot be clearly delineated. This is the case for data resulting from 3D scanning devices. Volume data sets correspond to a region of space, and do not contain explicitly-defined structures or surfaces. This has led to the development of an extensive number of dedicated algorithms and tools for the classification and representation of such data (Meissner, 2000).

This chapter presents theoretical aspects related to volume graphics, as well as rendering methods used in state-of-the-art volume visualization. It covers details such as the mathematical and physical models of volume rendering, GPU programming, and various volume rendering methodologies.

2.1. Volume data sets

Volume data, in its most basic form, is described by a function *V(x, y, z)* which will henceforth be referred to as the *volume function*, and which is described by Equation (2.1). This function associates one or more scalar values with every position in a region of space, therefore defining a scalar field. Data sets most commonly have one such single value per position, though multiple such values can be assigned, as is the case with multimodal data (Burns et al., 2007; Rieder et al., 2008).

$$\begin{aligned} &V : R^3 \rightarrow R^n \\ &V(x, y, z) = (scalar_1, scalar_2, \ldots, scalar_n) \end{aligned} \tag{2.1}$$

In practice, the data sets used in volume graphics are discretized, meaning that scalar values are available for a finite number of positions within the volume. These positions are referred to as *voxels*, short for *volume pixels*. Volume objects are made up of voxels in a manner similar to how 2D images are made up of pixels. Unlike pixels however, they can be arranged in multiple configurations. These are listed from the least, to the most generic (Jonsson, 2005):

- Regular: the voxels are arranged in a rectangular (or cubic) grid, voxels are the same size and are evenly spaced;

- Rectilinear: A more general case: the voxels are arranged in non-uniformly-sized rectangular grids;

- Structured / curvilinear: Non-linear grids in which the voxels are located irregularly; the grid may even wrap around structures;

- Unstructured: The most general case, where the voxels are located at arbitrary positions inside the volume (Leven et al., 2002; Callahan et al., 2006;).

Throughout the thesis, we will focus on datasets arranged in *regular grids*, with each voxel having a single associated scalar value. In a subsequent part of the thesis, we will show how other voxel properties can be derived from the scalar value and the spatial position of the voxel. However, for the purposes of this section, we consider a volume data set to be a multitude of scalar values arranged in a 3D matrix. If this matrix were super-imposed on the region of space where the data set originated, the elements of the matrix would characterize a measurable property of the various media inside the region. For example, in the case of CT medical data, higher scalar values would correspond to higher density material, such as bone, while lower values would denote softer tissue, such as skin or blood vessels.

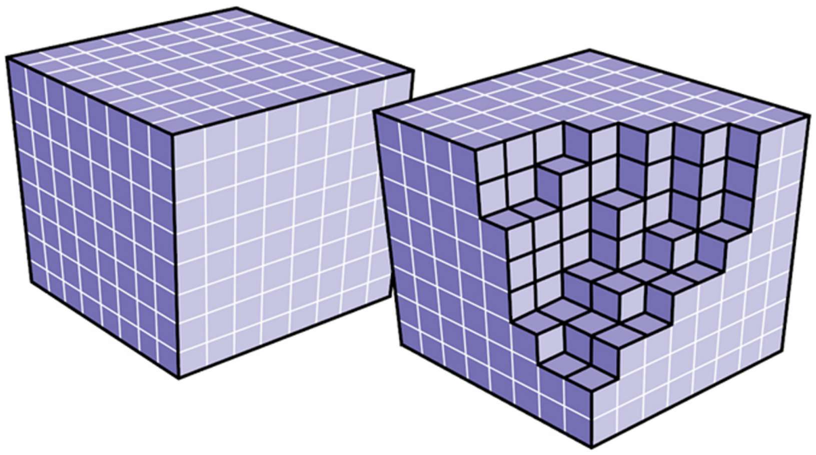


Figure 2.1. Representation of a volume dataset arranged in a regular grid (Hadwiger et al., 2006)

Figure 2.1 shows an intuitive depiction of a volume data set, with the voxels arranged in a regular grid. The scalar value is considered to be in the center of each of the smaller cubes, while the media inside each cube is homogeneous and completely characterized by its corresponding scalar value. The resolution of the data set (i.e. the number of cubes on each axis), influences the level of detail of the information that can be obtained from the data.

2.1.1. Data acquisition

There are multiple methods for obtaining volume data, each with its own applications, advantages and shortfalls. Most volume data sets result from a method of acquisition using a scanning device with a particular functionality. As a result, most data sets contain information from real-world objects or phenomena, such as the physiology of a medical patient, the structure of an industrial/mechanical part, or the shape, properties and features of environmental phenomena such as atmospheric disturbances or geologic underground layers.

One of the most commonly-encountered scanning methods is computed tomography (CT), with variations such as positron emission tomography (PET) or single photon emission computed tomography (SPECT). In CT, x-ray images are obtained from different directions around the scanned object. These images are then combined, while a reconstruction method is used to generate slices and, eventually, 3D data. X-ray scanning methods are efficient at detecting boundaries between different materials (such as bone and muscle tissue, for instance) but are not well suited for detecting boundaries within similar tissue (Bushong, 2000; Hsieh, 2003; Blankenberg, 2004; Seeram, 2009).

Magnetic Resonance Imaging (MRI) relies on a powerful magnetic field which causes the magnetic moments of protons from molecules inside the human body to align with the direction of the field. When the field is removed, the protons release energy in the form of photons, which are in turn detected by the scanner. Various properties of this phenomenon, such as the spin density or relaxation time, can be used to separate types of tissues (Rajan, 1997; WestBrook and Roth, 2005; Brown, 2010). Depending on the values of the scanning parameters, MRI imaging better suits the separation of tissues with similar density, such as brain tissue and cerebrospinal fluid. Figure 2.2. shows slices from CT and MRI scans of the same object. CT is efficient for accurately tracing the cranium and isolating higher density material (Figure 2.2(a)), while the MRI slice shows better contrast between similar soft tissues (Figure 2.2(b)).

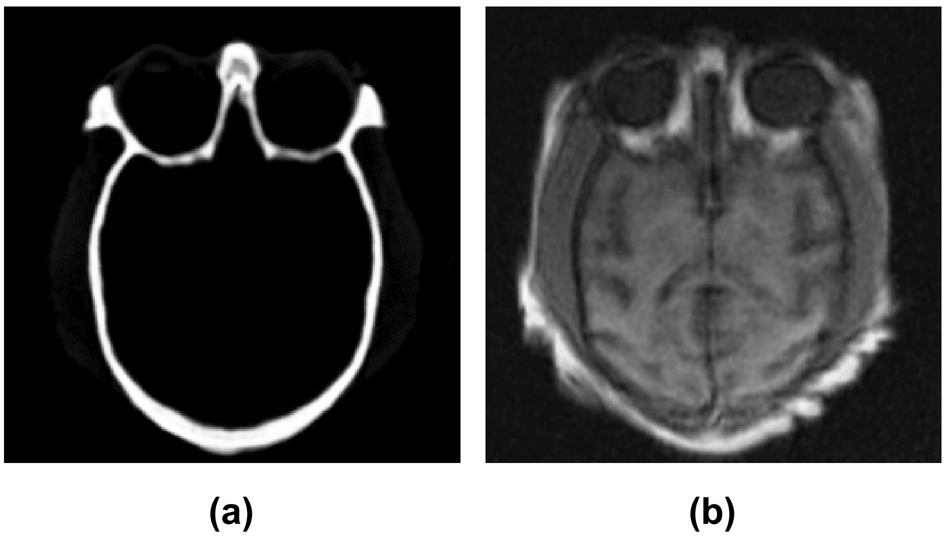


Figure 2.2. Slices from a multimodal dataset: (a) CT; (b) MRI. Bone tissue is more evident in the CT scan, while MRI data is better suited to classify soft tissue of similar density.

One of the more important aspects of an MRI or CT scan is the degree to which the substructures of the scanned object are capable of being separated from one another. This is especially problematic for areas where materials of similar consistency and density are found close together, often intermixed. One case which illustrates this issue is when attempting to identify vasculature among brain tissue. Cerebral blood vessel classification is notoriously difficult and often done through manual segmentation. Improving the quality and the degree of tissue separation in a volume data set can be carried out in a number of ways. One of these involves the use of contrast agents, which are substances used to increase the contrast between structures or fluids. Examples of such substances are iodine and barium, which improve the contrast of images obtained through x-ray-based methods (Antoch et al., 2003), and gadolinium, used to a similar effect for MRI scans (Caravan et al., 1999).

Medical sonography uses ultrasound to detect various types of tissues. A piezoelectric transducer emits high-frequency sound pulses which propagate into the human body and are partially reflected by the boundaries between different tissues. Sonography is particularly useful for separating the layers between solids and fluids. In medical imaging, this method is often preferred to CT or MRI, since the absence of radiation or powerful magnetic fields means that it poses considerably less risk than the alternatives (Szabo, 2004; Bluth, 2008).

Seismic imaging deals with the detection of subsurface features, such as geological structures or oil reserves. The data is gathered by emitting elastic waves into the underground, which are then reflected at boundaries between different materials. The reflections are picked up by seismometers, which register different amplitudes based on the physical properties of the materials encountered. Various techniques have been developed for the representation of such data, many of which focus on the identification and visualization of seismic features such as horizons and fault lines (Gibson et al., Patel et al., 2010; Gavrilescu and Manta, 2011).

Aside from the diverse range of acquisition systems, volume data sets may also be synthesized. In this case, scalar values are generated procedurally, usually according to some mathematical function or model. The most basic such data sets contain simple geometric primitives such as spheres, cones or cubes. These are primarily used for testing purposes in volume rendering applications when real-world data is not available, or when the use of simple data better illustrates a certain technique. A more complex type of synthetic volumes is used to simulate gaseous atmospheric phenomena, or, more generally, 3D objects with an undefined boundary. This is useful for depicting clouds or pockets of gas in simulations or electronic entertainment applications. Several state-of-the-art videogames feature volumetric clouds, fog and lighting. While these effects can be mimicked by polygonal geometry, the volume-rendered version is usually more realistic-looking and easier to manipulate within the context of the host application (Schpok et al., 2003).

In this thesis, we primarily focus on medical CT and MRI data sets, since volume visualization has vast applicability in medical imaging. The 3D visualization of a data set with an appropriately chosen distribution of color and transparency provides a more comprehensive means of interpreting the data, particularly for educational purposes, for medical staff with limited training in radiology, or for informing patients with no medical background.

2.1.2. Reconstruction and Filtering

Computerized acquisition methods produce discrete data sets comprised of a finite number of voxels, each with its own scalar values. When the volume data set is generated, samples are taken from the originally scanned object, or from the original simulation. This process is analogous to sampling a continuous signal and generating a discrete version, which is suitable for digital processing. However, in order to generate an image from this data, at some point a continuous signal should be re-generated from it. This process is referred to as *reconstruction*, and it involves applying an interpolation or approximation function to the discrete signal in order to obtain the data in-between the sampled points.

An ideal reconstruction of the original data from scanned samples usually comes with pitfalls, which are mostly a result of mismatching between the sampling frequency and the frequencies present in the original signal. Ideally, the sampling frequency should be twice as high as the maximum frequency in the continuous signal. However, in practice this is almost never the case. Nearly all volume data sets, especially medical ones, contain multiple structures with sharp boundaries in-between them. These sudden transitions result in very high (near-infinite) frequencies of the signal corresponding to the original volume, which would require an unrealistically high sampling frequency to properly capture. The frequency with which the volume is sampled during scanning translates to the resolution (i.e. the number of voxels) of the resulting data set. The higher the sampling frequency, the smaller and more numerous the voxels in the data set. Similarly to how a large number of pixels produces a more detailed 2D image, a high number of voxels results in more detailed structures within the volume. However, a sharp boundary such as the surface of the skin would theoretically require infinitely small voxels to properly trace. Reconstructing such data from a limited number of voxels results in aliasing artifacts, which cannot be completely eliminated, but can be masked to a lesser or greater extent (Haykin and van Veen, 1998).

Reducing sampling and aliasing effects depends largely on the type of filtering and anti-aliasing used for reconstruction. This is carried out using a properly-chosen reconstruction filter, i.e. a function which generates data values for positions that do not correspond to the original sampled points. Ideally, the sinc filter (Equation 2.2) is suited for a very accurate reconstruction of the original signal, but it is difficult to implement, due to its infinite spatial extent. In practice, frequently used filtering methods are the box and tent filters (Equations 2.3, 2.4) (Marschner and Lobb, 1994).

$$sinc(x) = \frac{\sin(\pi x)}{\pi x} \tag{2.2}$$

$$box(x) = \begin{cases} 1 & if\,|x| < T,\ T \in R^{+} \\ 0 & otherwise \end{cases} \tag{2.3}$$

$$tent(x) = \begin{cases} 1 - \frac{|x|}{T} & if\ |x| < T,\ T \in R^{+} \\ 0 & otherwise \end{cases} \tag{2.4}$$

The box and tent functions are low pass filters, meaning that all frequencies above a certain limit are eliminated, which usually results in a smoothing of the data, but a degradation of detail due to the loss of potentially important frequencies. Graphs for the functions associated with these filters are shown in Figure 2.3. The tent filter is considered a good compromise between quality and performance. It is implemented by doing a trilinear interpolation of the sampled points. The main advantage of such a filter is that it is supported by most GPU hardware, is thereby very efficient and need not be explicitly implemented in a visualization application. For greater image quality, higher order filters may be used, such as the windowed sinc or b-spline filters. These need explicit implementation and are therefore more customizable and scalable, but also require greater processing power. If the resolution of the dataset is high enough, only a very minor difference can be perceived in the rendered image, when a linear and higher order filter are applied. Whether a more computationally expensive filtering method is beneficial depends on the nature of the data set and the quality requirements of the resulting images (Carlborn, 1993).

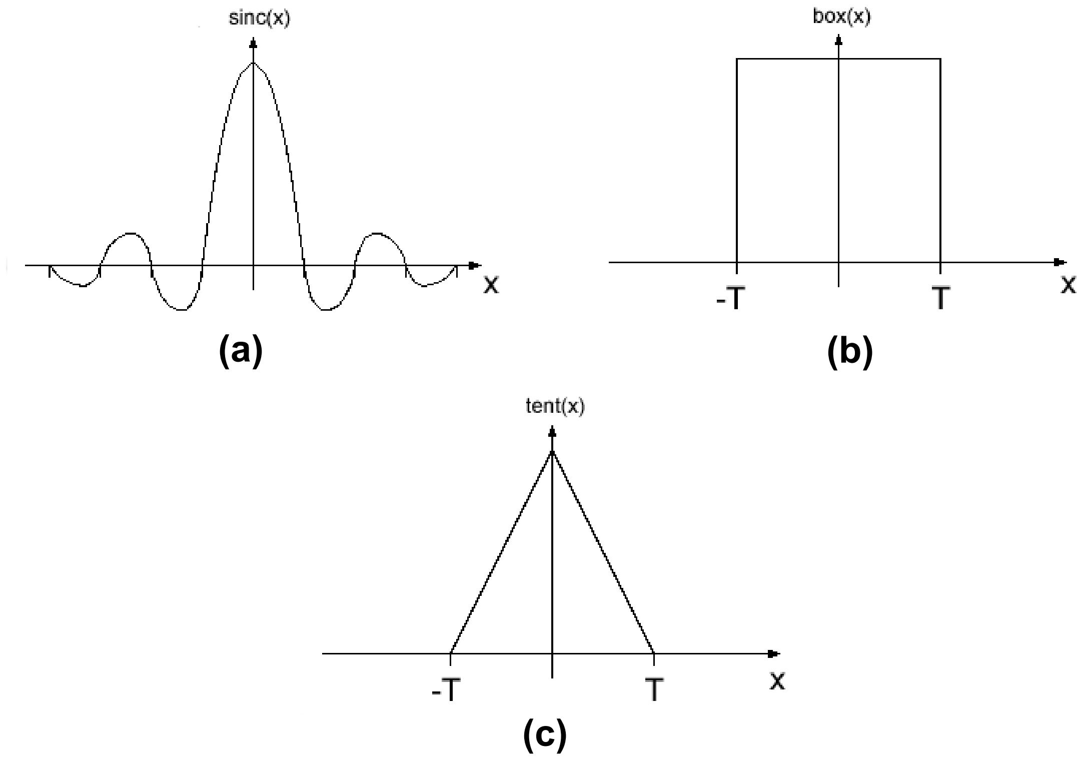


Figure 2.3. Various filters for signal reconstruction: (a) sinc filter; (b) box filter; (c) tent filter

The reduction of aliasing artifacts is intuitively referred to as *anti-aliasing*. A common anti-aliasing approach in volume graphics is to apply a filtering pass on the data set in a pre-processing stage, so as to remove higher frequency components. However, this approach has the drawback of blurring the data and thus eliminating potentially important fine details from the volume. Moreover, in the case of medical images, it is usually preferable that the originally-scanned data is not altered prior to applying any visualization methods.

2. 2. Theoretical Model of Volume Rendering

The purpose of volume rendering is to produce a 2D image from data which does not necessarily have any optical properties of its own. This is primarily achieved by traversing the data set, sampling, classification and then projecting the data onto a 2D space. The basic idea is to achieve a model which would characterize the interaction of light with various materials with different properties. Volume rendering aims to simulate the passage of light through the volume, while accounting for interactions between light rays and the various media inside the data set. The problems that volume rendering has to solve are mainly focused on the nature of the interaction between light and the material through which it propagates. Light may be absorbed, scattered, or emitted by the various materials inside a volume.

2.2.1. Physical Model of Light Propagation

Volume visualization deals with the extraction of visually-relevant information from a 3D scalar field, as described by Equation 2.1. The volume function V maps scalar values to positions in 3D space. As mentioned, these positions are typically arranged in a regular grid, and it is necessary to define V in every point inside the bounding box which encompasses the grid. This is achieved through various types of interpolation, the most common of which is trilinear interpolation (as explained in Section 2.1.2). One of the main tasks of volume rendering is to assign optical properties to the points described by V, based on their non-optical attributes. Each point provides its own contribution in terms of transparency and color when a light ray passes through it. Another important task of volume rendering is to account for the contributions of all points traversed by light, and to accumulate their contributions in order to form an image in a 2D display.

2.2.1.1. Equation of Light Propagation

The theoretical basis of volume rendering consists of geometric optical models, where it is assumed that light rays propagate in straight lines, unless certain interactions with the media take place. Specifically, we consider the following light-material interactions (Hadwiger et al., 2006):

- Emission: The points from within the propagation medium which are traversed by light rays emit their own light. For example, a hot gas may emit light through the conversion of heat energy into radiant energy.

- Absorption: The points from the media of propagation absorb incoming light rays partially or completely, and convert light energy into other forms of energy, such as heat.

- Scattering: The direction of light rays may be altered at certain points, depending on the properties of the local media. This causes the light to scatter (or spread) in multiple directions. This behavior means that light rays altered by certain points may contribute to the light energy emitted or absorbed by other points.

Absorption, emission and scattering alter the radiant energy along light rays. Light energy is described by its radiance *I*, defined in Equation 2.5. The propagation of light through any kind of medium other than complete void alters the value of *I* along the direction of propagation. The combined effects of absorption, emission and scattering result in the light propagation equation, shown in 2.6.

$$I = \frac{dQ}{dA\, d\Omega\, dt} \tag{2.5}$$

$$\omega \cdot \nabla_x I(x,\omega) = -\chi I(x,\omega) + \eta \tag{2.6}$$

Equation 2.5 defines the radiance as the radiant energy Q which is incident on the unit surface dA, in solid angle **Ω**, in time unit *dt*. In equation 2.6, $\omega \cdot \nabla_x I(x,\omega)$ is the dot product between the direction of light ω and the gradient of radiance I at position x along the light ray. If we assume that the light ray is parameterized with parameter s, then $\nabla_x I(x,\omega)$ can be rewritten as *dI/ds*. χ is the total absorption coefficient and it expresses the level of attenuation of light during its propagation through the media. Similarly, η is the coefficient of total emission and it symbolizes the direct contribution of the media to the increase of radiance (Hege et al, 1993).

The total absorption coefficient is itself composed of two terms: the real absorption coefficient k results for example from the conversion of light energy into other forms of energy such as heat; the scattering coefficient σ, as the name suggests, signifies the loss of light energy as a result of scattering. Therefore, χ can be rewritten as shown in Equation 2.7.

$$\chi = k + \sigma \tag{2.7}$$

Similarly, the total emission coefficient also encompasses two components: the first, q, is caused by various physical phenomena such as thermal molecular agitation, while a second one, j, quantifies the contribution of scattering (Equation 2.8).

$$\eta = q + j \tag{2.8}$$

The components *k*, σ and *q* are optical properties which are specified explicitly, either based on a physical model, or by means of a transfer function. *j* can be directly computed considering the contributions of light rays from all possible directions, as depicted in Equation 2.9 (Hadwiger et al., 2006).

$$j(x,\omega)=\frac{1}{4\pi}\int_{sphere}\sigma(x,\omega')p(x,\omega',\omega)I(x,\omega')d\Omega' \qquad (2.9)$$

The contributions of incidental light $I(x,\omega')$ are accumulated by integrating over all directions ω'. These contributions are weighed by the scattering coefficient σ and a phase function *p*, which signifies the probability that a light ray is redirected through scattering, from the initial direction ω' in a new direction ω. Therefore, the phase function represents the dependency of scattering on the angle of incidence. Different propagation media may have different phase functions, which lead to distortions in images obtained through volume rendering, in a matter similar to how a variation of the reflective properties of a surface alters the images of objects reflected by that surface.

By combining emission, absorption and scattering, assuming the phase function is normalized and considering Equations 2.6 - 2.9, the equation of light propagation results in Equation 2.10.

$$\omega\cdot\nabla_x I(x,\omega)=-(k(x,\omega)+\sigma(x,\omega))I(x,\omega)+q(x,\omega)+\int_{sphere}\sigma(x,\omega')p(x,\omega',\omega)I(x,\omega')d\Omega' \qquad (2.10)$$

Equation 2.10, in its current form, can only be applied to grayscale images. In order to process images which also contain color, the wavelength of the light rays must also be taken into account. This is achievable considering that radiance is dependent on the wavelength, as in $I_\lambda = dI / d\lambda$. However, visible light covers a limited range from the spectrum, between approximately 400 - 800 nm, corresponding to blue and red, respectively. Considering that light scattering is elastic (i.e. the wavelength does not change along a light ray), Equation 2.10 can be solved for individual wavelength values. In practice, only a very limited number of such wavelengths are considered, namely red, green and blue (Hadwiger et al., 2006).

2.2.1.2. Optical Models

Because of the difficulties in computing the equation of light propagation, simplified optical models are used instead. Usually, the simplification is carried out by eliminating one or more of the components from Equation 2.10. This results in models which are much more easier to work with, especially when limited computational resources are involved. The following optical models are therefore considered (Max, 1995):

- Absorption model: the propagation medium is initially completely dark, and may absorb incoming light.

- Emission model: the medium is completely transparent and may emit its own light.

- Absorption-emission model: this is the most frequently-used model in volume rendering. The medium may absorb external light or it might emit its own. In volume graphics, this corresponds to the situation where each voxel has its own color and opacity, i.e. it may contribute its own color component to the resulting image and, at the same time, it may partially or completely occlude other voxels behind it.

- Scattering and shading model: this model takes into account the contributions of scattered light from an external source. Shading is carried out considering the attenuation of incidental light.

- Multiple scattering model: the most complete model of this list and also the most difficult and computationally intensive to implement. All previously-described light behavior, including emission, absorption and scattering are taken into account.

The emission-absorption model is the most frequently-used in volume graphics. It provides a suitable compromise between generality, quality of the resulting images, and the computational power required. The model is described by Equation 2.11, which as mentioned, is obtained by simplifying Equation 2.10.

$$\omega \cdot \nabla_x I(x,\omega) = -k(x,\omega)I(x,\omega) + q(x,\omega) \tag{2.11}$$

Equation 2.11 is the *volume rendering equation*. For a single light ray, it can be written as in 2.12, where *s* is a parameter which denotes positions along the ray.

$$\frac{dI}{ds} = -k(s)I(s) + q(s) \tag{2.12}$$

2.2.2. The Volume Rendering Integral

This integral draws upon the results from the previous subsection and constitutes the formula for computing the radiance along a light ray, from starting position $s = s_0$, until the end position $s = s_E$. The integral is shown in Equation 2.13.

$$I(s_E) = I_0 e^{-\int_{s_0}^{s_E} k(t)dt} + \int_{s_0}^{s_E} q(s) e^{-\int_{s}^{s_E} k(t)dt} ds \tag{2.13}$$

In Equation 2.13, I_0 is the starting radiance of a light ray as it enters a region of space (or, for our purposes, the volume), in position $s = s0$. $I(s_E)$ is the radiance of the light ray which exists the volume and reaches the viewing position of the observer. The first component of Equation 2.13 represents the light which travels from the exterior, and which is attenuated as it traverses the volume. The second component expresses the accumulated emissive and attenuative contributions of the points along the ray, from the initial to the exit positions .

In order to simplify Equation (2.13), let $T(s_a, s_b)$ be the transparency between any two positions s_a and s_b. T can be expressed as in Equation 2.14. In turn, this leads to a simpler expression for the volume rendering integral, shown in Equation 2.15.

$$T(s_a, s_b) = e^{-\int_{s_a}^{s_b} k(t)dt} \tag{2.14}$$

$$I(s_E) = I_0 T(s_0, s_E) + \int_{s_0}^{s_E} q(s) T(s, s_E) ds \tag{2.15}$$

Each point along the ray contributes its own color and transparency, as the ray passes through. In the exit position s_E the color and intensity of light as it leaves the volume and reaches the observer results from the accumulated contributions of points from each position along the direction of propagation. The expression in Equation 2.15 is a frequently-used mathematical model for volume rendering. In later sections, we will present numerical methods for solving this equation and explain how it is used within the context of volume visualization (Hadwiger et al., 2006).

2.3. GPU Architecture and Processing

Nearly all modern volume visualization applications, including UniVolume, make extensive use of the capability of recent-generation GPUs. Volume rendering and all associated operations require a lot of computational power. For instance, in modern volume rendering approaches, the Equation 2.15 has to be numerically solved for every pixel of the output image. This involves a lot of calculations, not factoring in other volume visualization enhancements, such as shading, illumination, or other more complex processing. It is therefore difficult to achieve real-time volume visualization using the CPU and conventional system memory. Conversely, GPU hardware is dedicated to fast, multi-threaded operations, and makes it possible to implement complex visualization algorithms at manageable frame rates.

One of the most important innovations in computer hardware consists in the introduction of fully programmable elements within the graphics pipeline. Programmers can write code which influences how the GPU handles vertices, fragments and geometry. Though this programming model differs from conventional CPU programming, it is nevertheless a welcome advantage for computer graphics in general.

2.3.1. The Hardware Graphics Pipeline

In graphics processing, objects are decomposed into polygons, a process known as *tesselation*. The GPU has the capability to efficiently produce raster images from tesselated objects. The process through which the GPU generates images composed of pixels from polygons, via rasterizing, is referred to as *display traversal.* All GPUs implement display traversal by means of a graphics pipeline which consists of several processing stages. This pipeline is shown in Figure 2.4.

The vertex processor performs transformations for each vertex in 3D space. Vertex coordinates are transformed by a model matrix, converted to the viewer coordinate system through a viewing matrix, and to screen space by means of a projection matrix. In the primitive generation phase, geometric shapes such as lines and triangles are formed by grouping vertices together. In the fragment processing stage, a rasterization step first decomposes geometric objects into fragments. Fragments correspond to on-screen pixels, but each object has its own set. A pixel in screen space is assembled from one or multiple fragments, depending on whether or not objects overlap.

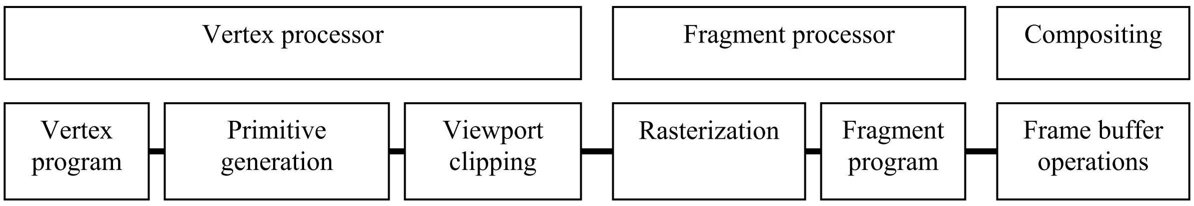


Figure 2.4. Stages of the GPU hardware pipeline; the data passes through from left to right

Other calculations also take place in the fragment processing step, such as perspective correction, texture fetches, and per-fragment filtering. Finally, the compositing step takes place right before transferring fragments to the frame buffer, and consists in a series of tests where it is decided which fragments are to be discarded (for example, if they are completely occluded). It is then decided whether the fragments shall be combined with the values already present in the frame buffer (Nguyen, 2008).

2.3.2. The Fragment Processor

Many of the essential operations in volume rendering are carried out in the fragment processor. Here, the developer can write shader programs which perform custom computations for the color and opacity of each fragment. Common languages for writing such programs are GLSL (Rost et al., 2009), CG (McBrewster, 2011), developed by nVidia, and HLSL, developed by Microsoft for Direct3D (St-Laurent, 2005).

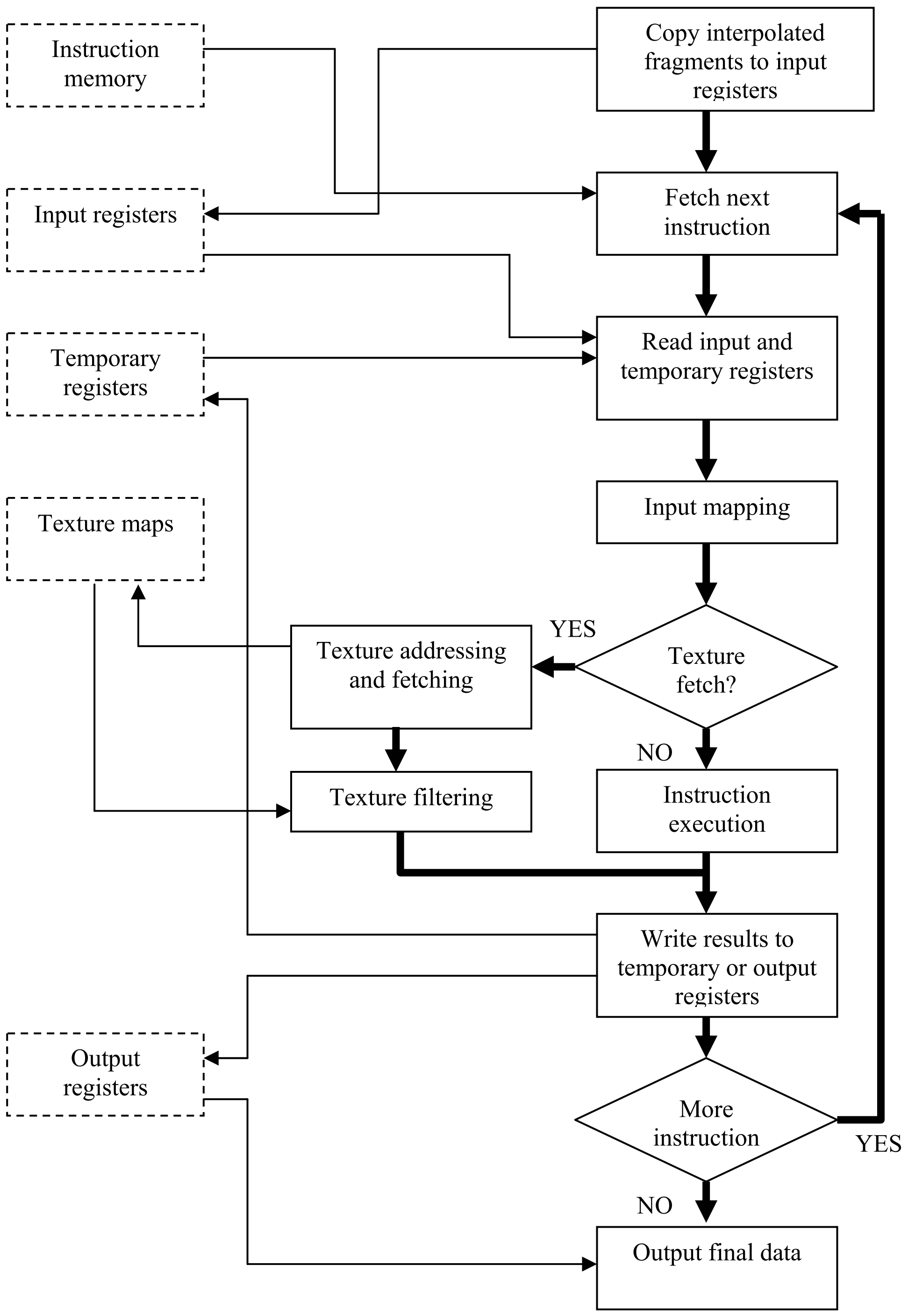


Figure 2.5. Diagram illustrating the operation of the fragment processor (adapted from Hadwiger et al. (2006))

A shader program is executed once for each fragment, in a separate hardware thread. The fragment processor receives attributes or texture coordinates from the rasterization unit or from data written by the developer to input registers, then it executes the currently-bound fragment program and writes fragment information (color and opacity) to output registers. A diagram of the operations performed within the fragment processor is shown in Figure 2.5.

Following fragment processing, a series of tests are performed for each fragment, such as the alpha, stencil and depth tests. The fragments which correspond to each pixel are then combined through alpha blending, i.e. an opacity-weighed accumulation of color. The fragment data can then be written to a frame buffer, a region of memory where a complete set of fragments (forming a rendered image) is stored. The frame buffer can be the standard one used for sending the image to a display device, or custom-defined via a frame buffer object (FBO). FBOs may have various regions of memory attached to them, the most common of which are textures. Thus, a rendered image can be retained in a texture of the same size as the display viewport. This allows for great flexibility when customizing the output result, since post-processing effects and various other algorithms can be applied to the image within the texture.

2.4. Volume Rendering Methods

Volume rendering encompasses a series of techniques used to generate a 2D projection from three- or multi-dimensional volume data sets. For the purposes of this thesis, we consider the data sets to be regular grids, with the voxels arranged in a uniformly-spaced 3D array. Volume rendering implements methods for the numerical computation of the integral in Equation 2.15. The basic idea is that, once the initial discrete volume has been reconstructed via some form of interpolation/filtering, the rendering method traverses the volume and samples it at certain locations, which may or may not be evenly-spaced. The resulting sampled points are then used to model the propagation of light through the volume.

Color and opacity are assigned to each sampled point by a classification algorithm, then the colors and opacities of sampled points are combined to form the colors of on-screen pixels. The colors seen in the resulting images are obtained from the accumulated contributions of sampled locations from within the volume. Therefore, volume rendering algorithms attempt to approximate light behavior as described by Equation 2.15 as closely as possible. Naturally, since the processing is done by a digital device which can only work with finite values, the volume rendering integral cannot be perfectly evaluated. However, through a combination of appropriately-chosen sampling rates, artifact reduction and elimination algorithms, and image enhancement techniques, the rendering process can produce smooth, high quality images at interactive frame rates. As mentioned in Section 2.3, the computational capabilities of GPU hardware are helpful in achieving this goal.

Several stages are involved in volume rendering, which together form the volume rendering pipeline. A generalized version of this pipeline is shown in Figure 2.6. Essentially, once the dataset is in memory, a combination of reconstruction, sampling and classification yields a number of sampled points from within the volume, each of which has an associated color and opacity (most commonly in the RGBA color space). The RGBA values for individual points are combined together in a compositing step, which produces the colors of pixels in image space. Compositing is usually carried out as a function of opacity, i.e., sampled points with low opacity contribute their own color less than points with a high opacity. This is meant to mimic real-world vision, where the more transparent an object is, the more visible is the background or other objects behind it.

In the classification stage, color and opacity are assigned to points inside the volume based on various properties. Classification is a wide topic and is extensively discussed in the third part of the thesis. The ordering of the pipeline in Figure 2.6 places classification after reconstruction. However in certain situations, these positions might be altered, and the classification stage would take place before reconstruction. In this case, voxels would be assigned color and opacity directly, and any reconstruction would take place based on color, rather than scalar values. This is referred to as *pre-classification*. Image generation refers to the fact that, once colors have been generated for each pixel, the resulting image may be sent to the video frame buffer for displaying on-screen, or it may be saved to an intermediate location, such as an FBO attachment (usually a texture). In the latter case, further image processing algorithms may be incorporated in the optional post-processing stage, in order to further improve or enhance the resulting image.

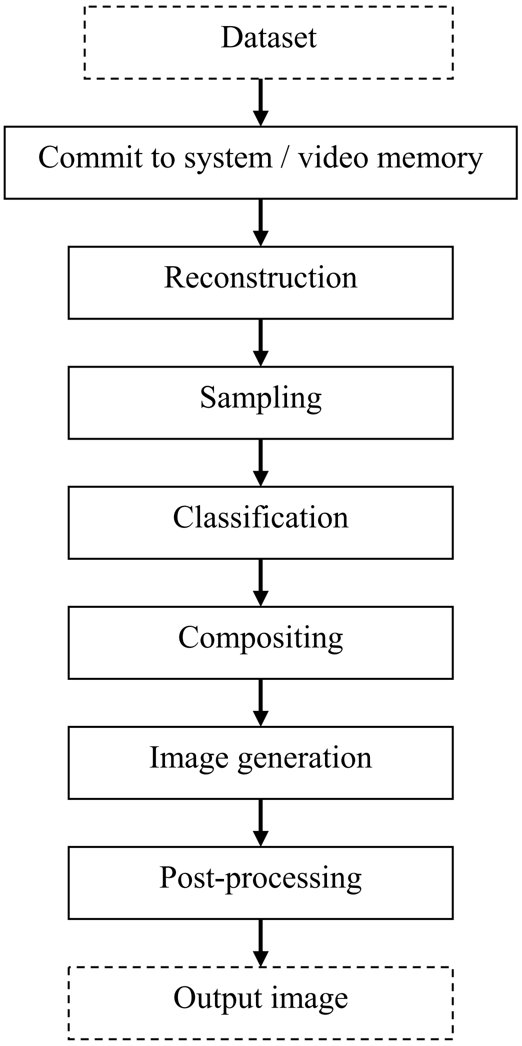


Figure 2.6. Generalized volume rendering pipeline.

There are two main spaces in which volume rendering can operate: object space and image space. This results in two main categories of volume rendering techniques: object-ordered and image-ordered. The main difference is that the first volume rendering approach takes place in the coordinate system of the volume, while the second one is carried out on a per-pixel basis, starting from the 2D viewport.

2.4.1. Object-Ordered Volume Rendering

Current GPU hardware only supports vertex-based primitives as input, and it cannot process voxels directly. For example, it cannot directly work on a solid sphere or cube. Therefore, the solution adopted by volume rendering techniques is the decomposition of the volume into polygonal primitives, which the GPU can operate on.

The main idea behind object-ordered volume rendering is that a discrete 3D dataset can be split into a stack of 2D slices. Thus, a 3D volume can be represented by rendering a sufficiently-large number of semi-transparent slices extracted from it. The slices are drawn on 2D primitives, while the GPU renders the polygons which form these primitives. We refer to these primitives as *proxy geometry*, because they aid in rendering the volume by describing its domain, or parts of its domain. Proxy geometry does not, in itself, contain information on the shapes and particularities of the volume. The volume data is stored in 2D or 3D textures, depending on the implementation (Hadwiger, 2004).

The method involves slicing the volume with planes and assigning vertex positions and texture coordinates to proxy polygons within these planes. The most basic implementation of this method involves storing the data on a set of 2D textured polygons. Classification, illumination and shading are carried out inside these polygons. Then, they are combined together, to form a final, rendered image. In this case, the proxy geometry is a stack of slices which are aligned along the axes of the object coordinate system. Three sets of such slices are necessary, one for each axis, to allow for the interactive rotation of the volume. Figure 2.7 shows two images obtained through axis-aligned slicing. In Figure 2.7(a), a small number of textures is used, therefore the individual slices are clearly visible. Figure 2.7(b) shows a rendering using the full number of slices; the slicing effect is no longer visible in the image (Figure 2.7(b)).

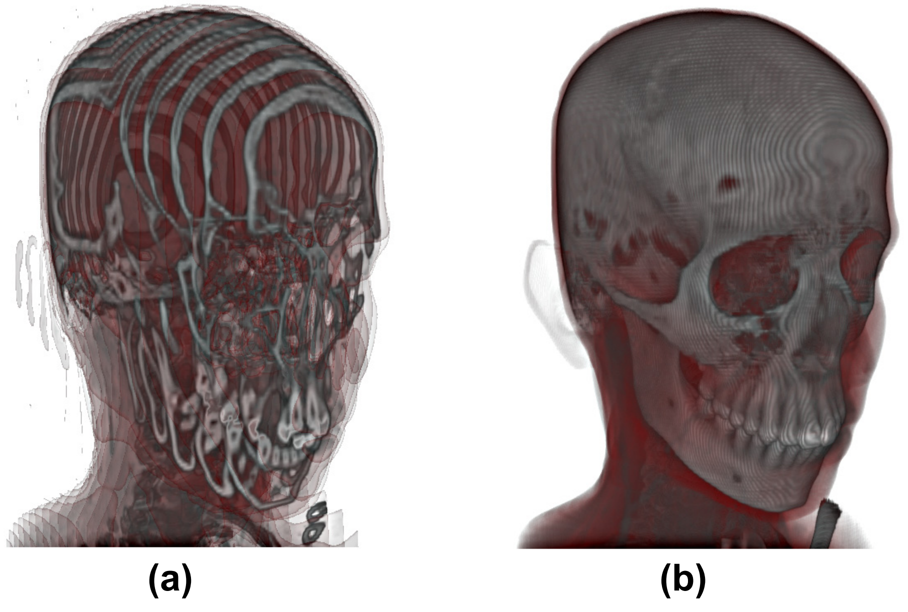


Figure 2.7. Object-ordered slice-based rendering. (a) a small number of textured polygons shows individual textures; (b) the same data set, rendered with the full number of slices.

The main advantage of this approach is the compatibility with less capable GPUs, which do not support 3D textures. Indeed, virtually any GPU can render 2D textured polygons and perform bilinear interpolation, which is required to combine the slices. The method suffers however, when it comes to image quality. Aliasing effects may occur at the edges of the polygons due to the low sampling rate, which is usually fixed and determined by the resolution of the volume along the axis of alignment. Since three sets of polygons are required to ensure proper rotation, the method is also rather memory inefficient.

An improvement upon this approach can be obtained by switching from object-aligned slices to view aligned ones. The main difference where GPU processing is concerned, is that this time the volume is stored in a 3D texture, a feature which is supported by later generation GPUs. Using a 3D texture avoids the need to split the volume into individual, static slices. In this case, multiple planes are used to slice through the volume. These planes may have an arbitrary orientation, independent from the directions of the axes of the object coordinate system. The obvious choice for the positioning of the slicing planes is to orient them parallel to the viewport, thus, the resulting slices would be view aligned, as opposed to object aligned.

Figure 2.8 shows two images rendered using this approach. In Figure 2.8(a), a lower sampling rate is used in order to reveal the slicing planes, while in Figure 2.8(b) a significantly increased sampling rate means the image has a more continuous look. The method produces higher-quality images, while the now-flexible sampling rate means fewer aliasing artifacts. The algorithm for determining the intersections between slicing planes and the bounding box of the volume is, however, more complex (Rezk-Salama, 2005). Since it has to be applied multiple times per second, it constitutes a bottleneck in terms of performance.

Generally speaking, object-ordered rendering methods are hindered by the fact that they depend on the size of the data set, and are strongly influenced by its complexity. They also generate a large number of fragments, many of which are not used in the final image, thus resulting in an unnecessarily high number of computations. However, they are suited to certain volume rendering-related enhancements, such as the implementation of certain global illumination algorithms (Lindemann, 2011).

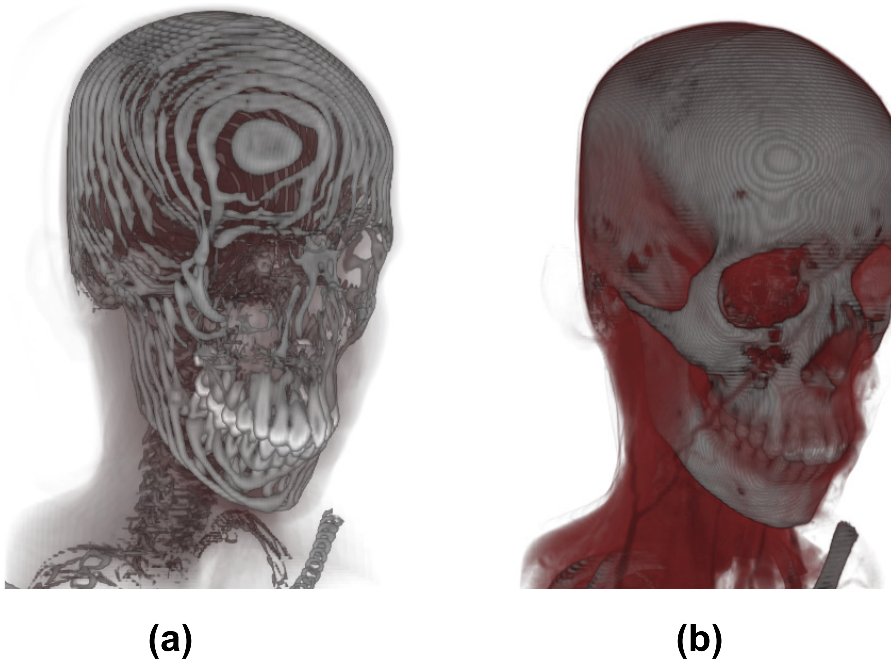


Figure 2.8. A volume data set rendered using view-aligned slices: (a) a low sampling rate reveals individual slices; (b) a higher sampling rate produces a more accurate image.

2.4.2. Image-Ordered Volume Rendering

This class of volume rendering methods operates on a per-pixel level, while computations are carried out in image space. Such methods do not depend on the size and complexity of the volume data set, but rather on the resolution of the output image. Most of the methods used by state-of-the-art volume rendering applications are variations of the *ray casting* algorithm. The idea is to project rays from every pixel into the volume, while sampling, classification, shading and any other processing take place along these rays. The color for each pixel results from the accumulated contributions of points along its corresponding ray. Ray casting allows for a direct computation of the volume rendering integral, since it directly mimics the behavior of light rays passing through the volume (Ray et al., 1999; Krüger and Westermann, 2003; Hadwiger et al., 2005). The principle of ray casting is illustrated in Figure 2.9.

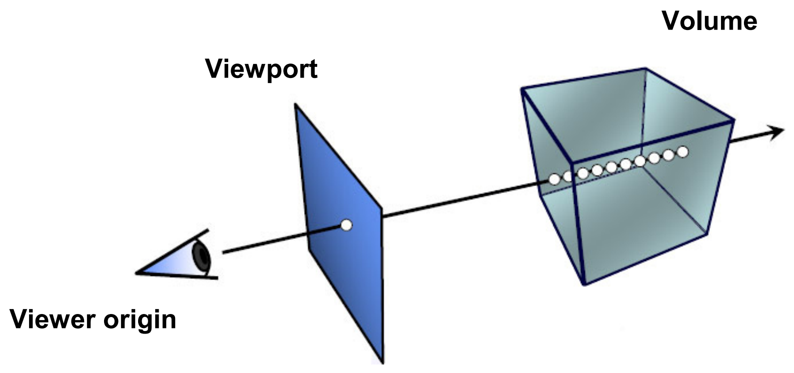


Figure 2.9. Conceptual depiction of the ray casting algorithm. A ray is sent from the viewport into the volume, and samples are taken along the ray at discrete positions.

Image-ordered rendering techniques need not map data to textured proxy geometry, and instead rely on directly computing color from points in the volume. The basic structure of ray casting is as follows:

- for every pixel, a ray is sent into the volume (Figure 2.10(a)). Implementation-wise, this means that the volume is linearly traversed, between start and and exit positions located either inside of it or on the faces of its associated bounding box.

- sample values are taken from the volume, at certain positions along each ray (Figure 2.10(b)). Most commonly, the sampling rate is constant, which means that the sampled points belonging to a ray are equidistant. Sampled points on different rays are not usually aligned with each other, particularly if perspective projection is used.

- a classification method is used to assign optical properties to each sampled point (Figure 2.10(c)). This is commonly done by means of a transfer function, which maps RGBA quadruplets to scalar values from the dataset or to other voxel properties, such as the spatial position or visibility. Illumination and other kinds of shading may also be applied in this step.

- finally, the colors and opacities of sampled points along each ray are combined together, resulting in the color of the corresponding pixel (Figure 2.10(d)).

In the final step of ray casting, the accumulation of optical properties which results in the color of the corresponding pixel is done through color and opacity compositing. A common compositing method is shown in Equation 2.16.

$$\begin{aligned} C_{dst} &= C_{dst} + (1 - a_{dst})\, C_{src}\, a_{src} \\ a_{dst} &= a_{dst} + (1 - a_{dst}) a_{src} \end{aligned} \tag{2.16}$$

C_{dst} accumulates color values for a pixel. C_{src} is the color value of a sampled point along the ray. a_{dst} and a_{rsc} are the accumulated opacity and the opacity of the currently-sampled point, respectively.

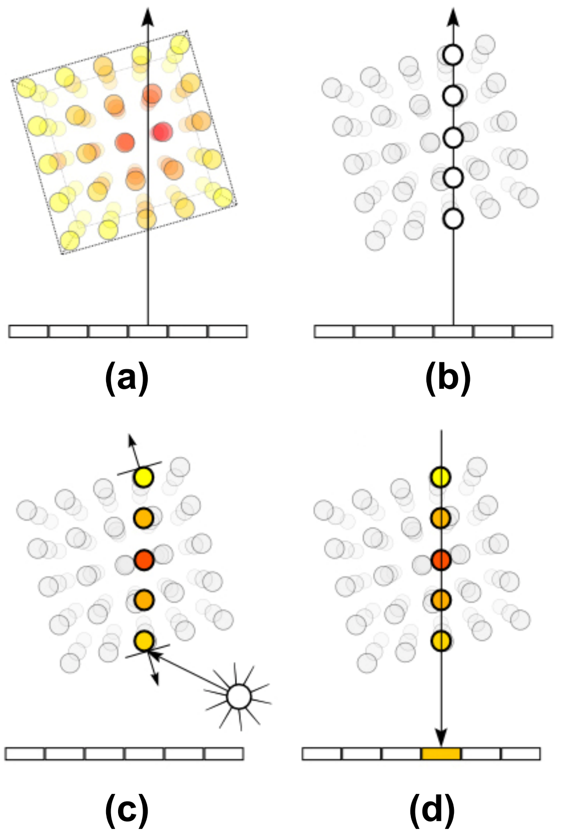


Figure 2.10. Steps of the ray casting algorithm: (a) casting a ray through the dataset; (b) sampling; (c) classification and shading; (d) compositing

Figure 2.11 presents an example of an image obtained via ray casting. The image shows two main isosurfaces (i.e. sets of voxels with the same scalar value). An external one which corresponds to lower-density material has been rendered transparently, and it reveals another isosurface underneath, which corresponds to higher-density bone tissue, which is opaque. The flexibility of direct volume rendering approaches allows for high quality images to be obtained. Image enhancing effects such as illumination are also much easier to implement within a ray casting loop and add a dose of realism to the resulting image (Ropinski et al., 2008).

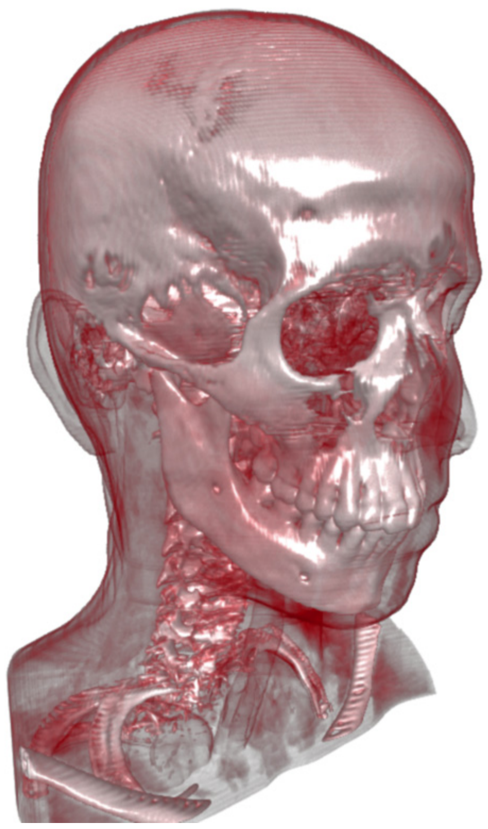


Figure 2.11. Image of a CT dataset obtained using ray casting-based rendering

Conclusions

In this first part of the thesis we have presented concepts and techniques pertaining to the fundamentals of state-of-the-art volume visualization. When rendering a volume data set, the idea is to simulate the propagation of light through the heterogeneous media within the data set. The influence of these media on incidental light rays is mathematically modeled by means of the volume rendering integral. This formula essentially states that when light passes through a point in the environment, the optical properties of that point may change the radiant energy and/or direction of the ray. In volume graphics, this concept translates to the fact that when hypothetical light rays pass through sampled voxels within a volume, the color of the pixels in image space is influenced by the color and opacity of the voxels, in the same way as semi-transparent and opaque objects affect the color we perceive in real life. Multiple methods have been developed for the simulation of this phenomenon in volume data. These methods have progressed alongside GPU technology: while object-ordered rendering methods are mostly suited to older-generation GPUs, current state-of-the-art renderers use ray casting-based approaches. This is not universally valid of course, since there is always the case that certain methods are suited to certain tasks.

In UniVolume, we primarily rely on ray-casting methods for our renderers. Aside from the elements mentioned so far in this chapter, several optimizations such as early ray termination or empty space skipping contribute to increased performance, while various methods for the reduction of sampling and filtering artifacts contribute to the quality of the rendered images. However, elaborating upon all optimizations and methods used within the rendering pipeline would extend far beyond the purposes of this thesis. In subsequent chapters, we describe more in-depth elements concerning the representation of information from volumes, and present our own methods and related implementations.

3. Volume Classification and Feature Enhancement

Section 2.4 contained a few images depicting rendered images from medical volume data, specifically Figures 2.7, 2.8 and 2.11. In these images, a semi-transparent layer of soft tissue (skin, blood) surrounds an opaque interior, mostly depicting opaquely-rendered bone tissue. This is, however, only a small portion of the entire data set. Indeed, the information of interest from a data set might be as low as 10-15% of the total available data. Usually, the vast majority of the data is unimportant, and the user/developer is interested in only a small percentage. Figure 3.1 shows a rendering of the same data set as in Figure 2.11. This time, however, a bounding box showing the extent of the entire data set is drawn around the volume. It can be seen how a large portion of the box is empty, with the volume occupying the central area only. Specifically, 23.12 % of the voxels in this image are semi-transparent or opaque, while the rest are fully-transparent. The information of interest has been separated from the rest of the data based on opacity. Furthermore, it can be seen that even among this information there is also a degree of separation, by means of opacity and color. Softer tissue is red and less opaque, while harder tissue is white and more opaque. Therefore, the structures within the volume have been classified according to their density. Anatomical features such as the skin, blood vessels and skull have been visually separated from other materials inside the scanning device.

The above is a simple example of volume classification. The classification stage is an important component of the volume rendering pipeline shown in Figure 2.7. It has to be explicitly implemented, since, as mentioned, volume data sets do not intrinsically contain any information regarding the features and structure of the elements within. A data set only contains an array of scalar values and it is up to the developer to establish criteria for the suitable separation of meaningful data from the meaningless rest. What specifically makes data meaningful is determined on a case-by-case basis. For medical data, the information of interest might constitute bone, blood vessels, tumors, or other anatomical structures.

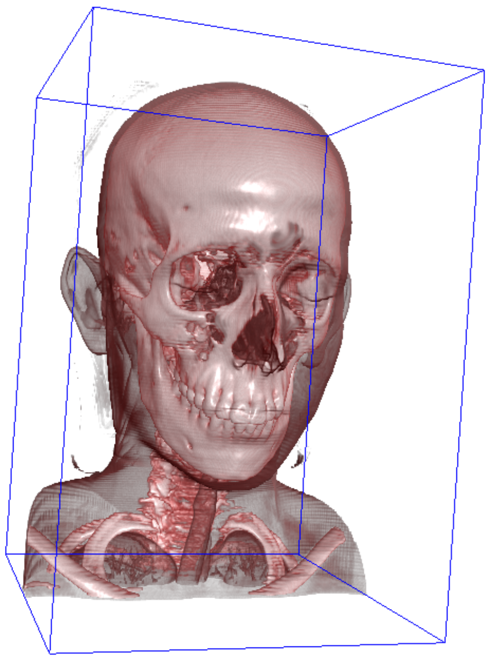


Figure 3.1. Rendering of a volume data set and its associated bounding box. The information visualized from within the dataset represents a small percentage of the total data.

In the case of seismic data, one might need to identify oil deposits, or geological features such as horizons and fault lines (Jeong, 2006). Due to the wide variety of existing data sets, each with its own distribution of scalar values, the topic of volume classification is very broad. Since volume datasets do not intrinsically contain information on the shape of the originally-scanned objects, classification methods have to be implemented based on explicitly-established criteria.

Classification essentially involves the separation of shapes, structures and area of interest from within a data set. While an extensive number of approaches exists in this direction, classification methods fall within two main categories:

- **segmentation**: various specific algorithms are applied, usually in a pre-processing stage, to identify the voxels which belong to surfaces, edges or meaningful structures. Rendering of the data is usually applied separately from segmentation, thus the actual visualization stage is decoupled from the classification stage. The segmentation process results in one or several segmentation masks. In the simplest case, the mask is an array of the same shape and size as the data set, where each element is a boolean value (i.e. either 1 or 0). Each value in the segmentation mask corresponds to one voxel from the data set. The values in the mask specify whether the associated voxels belong to the segmented structure or to the remaining unsegmented data. In more complex cases, where more than one structure has been segmented, the mask contains object IDs for each voxel. These specify which of the segmented structures the voxel belongs to (Sherbondy et al., 2003; Olowoyeye et al., 2009; Prassni et al., 2010).

- **transfer functions**: this approach encompasses a class of methods which use various kinds of mappings to assign color and opacity to voxels based on various criteria. Most commonly, a transfer function maps RGBA quadruplets based on the scalar value of voxels within the dataset. This is a powerful classification criterion which allows, for instance, the separation of tissues in a CT medical dataset based on their relative densities. Transfer functions are specified using various user interfaces, though automated specification methods have also been developed (Kindlmann, 1998; Zhou, 2009).

The first part of this chapter focuses on classification by means of transfer functions. Since it is an important part of volume visualization, we have implemented multiple classification methods in UniVolume, based on various types of transfer functions. These functions primarily operate in domains defined by voxel properties such as the scalar value, the gradient magnitude, and others. We present methods for computing these properties and then show how they can be used for classification purposes. Each property allows for the separation and highlighting of increasingly complex features of the volume, from a classification point of view. We then present a practical method through which various 1D transfer functions can be combined, forming multidimensional transfer functions with an unrestricted number of dimensions. We show how these functions allow for flexible classification, while their specification is carried out in a manner which draws upon tried-and-tested methods. Lastly, the classification approaches are applied to segmented MRI data, in a manner which enables focus on features of interest while preserving context.

The second part of this chapter presents various methods for the highlighting of features in rendered volume data. The features primarily refer to contours, shapes and non-uniform regions within volumes. Depending on the rendering method used, these features may not be readily visible, therefore requiring some manner of emphasis. Thus, we propose methods based on the highlighting of contours and edges, and we show how these methods can improve the visibility of various details in rendered images.

The topics of classification and feature enhancement have been dealt with rather extensively, but not exhaustively, in the related literature. Thus, we bring contributions in this area through the implementation of improvements to the discussed algorithms, and the development of alternative, often more efficient ways to achieve the desired results. Each method is described in its respective section and results are conveyed through rendered images (sometimes in the *before* - *after* style) and appropriate explanations.

3.1. Transfer Function-Based Classification

As items for the classification of volume data, transfer functions are part of the volume rendering pipeline. Along with illumination and shading, they are among the most significant factors which control the colors in the resulting rendered image. As mentioned, volume data sets do not intrinsically contain any information on the optical properties of their constituents. Still, the simulation of light propagation through the volume does require that these properties are known. In order to address this issue, transfer functions assign opacity and color to voxels, based on one or more of their non-optical properties. In the general case, a transfer function is defined as shown in Equation 3.1. T_f is a multidimensional transfer function which maps an RGBA quadruplet to each voxel within a data set based on voxel properties v_1, ..., v_n. Such properties might be, for example, the scalar value, gradient magnitude, or visibility.

$$\begin{aligned} &T_f : R^n \rightarrow R^4 \\ &T_f(v_1, v_2, \dots, v_n) = RGBA \end{aligned} \tag{3.1}$$

The most basic transfer functions map color and opacity based solely on the scalar value. These are referred to as *data-based transfer functions* (Kindlmann, 2002). This type of transfer function is the most frequently used and it requires the least amount of computation, since it operates on voxel samples directly. The basic principle is that meaningful information should be made visible, while other less useful data is rendered semi- or fully-transparently.

3.1.1. Pre- and Post-Classification

Earlier, in section 2.4, we mentioned that the classification stage may be placed in different positions within the volume rendering pipeline. Specifically, the issue is where classification is placed in relation to the reconstruction phase. If classification takes place before reconstruction, color and opacity values are assigned to the voxels in the original discrete data set. Reconstruction then takes place in screen space, based on the optical properties. For positions other than exact grid locations, color and transparency are computed by interpolating from neighbors in the closest grid points. This is referred to as *pre-classification* (Engel et al. 2001, Hadwiger et al., 2006).

The alternative is to perform classification after the reconstruction phase, i.e., *post-classification*. First, voxel data for each position in the volume is computed by interpolating from grid points. This is followed by the classification phase, where color and opacity are assigned to samples taken from the now-continuous volume. Optical properties are obtained from a look-up table where the corresponding interpolated value is the search index. The two methods are presented in Figure 3.2. Usually, pre-classification produces lower quality images than the alternative, due to sampling artifacts which arise from the fact that the volume is classified at the original resolution of the data set. Post-classification allows the use of an arbitrary sampling rate, which makes it possible to better handle transfer function-induced high frequencies.

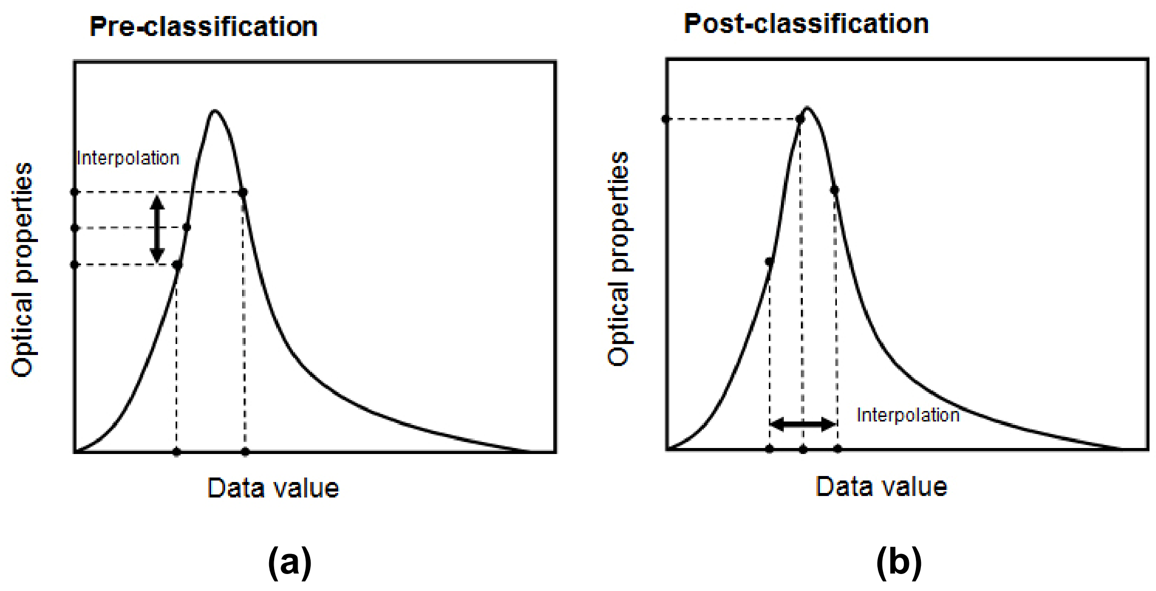


Figure 3.2. Optical property mapping for (a) pre-classification and (b) post-classification.

In the case of pre-classification, optical properties are first assigned to data values, then interpolated for an arbitrary point. In post-classification, data values are first interpolated, then optical properties are assigned to the interpolated point (adapted from Hadwiger et al., 2006).

3.1.2. Transfer Function Specification

The specification of transfer functions is a task made complex by the diversity of data available in a volume. Medical data sets, in particular, contain a lot of fine details, such as thin surfaces, blood vessels and small structures. It is also difficult to assess which specific structures are of interest and which are less useful, since different users want to look at different details in a rendered image, depending on their specialization, interests and needs. For this reason, in UniVolume we opt for manual transfer function specification. We have implemented tools and interface elements which guide the user in deciding on the level of detail and the amount and type of information which should be displayed on-screen, as a result of classification. The most basic such interface element is presented in Figure 3.3, and it is used to adjust a 1D data-based transfer function. The Figure shows a control *widget* and the associated resulting rendering of a volume data set (the term *widget* stands for *window gadget* and is a generic term for user interface elements). The horizontal axis contains the scalar value which in this case is equivalent to the density of the material in the volume. The vertical axis contains opacity values, from completely transparent, to fully opaque.

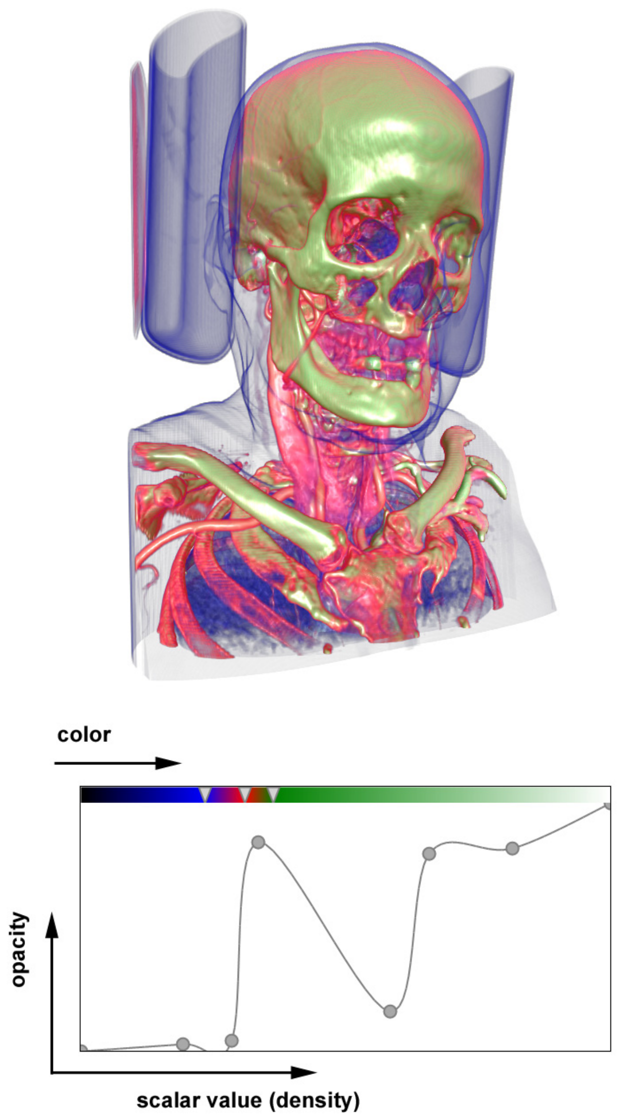


Figure 3.3. Transfer function specification interface and resulting rendering. The circular and triangular control points allow the adjustment of the distribution of opacity and color among voxels, respectively.

The circular control points in the middle area of the widget are used to specify the opacity component of the transfer function. The component is then obtained by applying some type of interpolation in-between these control points. In this particular case, we use a Catmull-Rom spline (Catmull and Rom, 1974), which provides a smooth transition among transfer function values. The spline is constructed from positions generated as shown in Equation 3.2. Considering four consecutive control points, $P_1,...,P_4$, a point $Q(t)$ on the spline is computed in a position parameterized by t, in-between P_2 and P_3.

$$Q(t) = 0.5\left[1, t, t^2, t^3\right]\begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \tag{3.2}$$

The second component of the transfer function, which is used to specify color, is adjusted at the top of the widget. The actual color is achieved by linear interpolation in-between these control points. Both components are adjusted independently via their respective sub-regions of the widget.

In UniVolume, we use this type of widget extensively, because it is practical and similar to other interface elements encountered in larger volume rendering applications (Bruckner and Gröller, 2005; Meyer-Spradow et al., 2009). The shape of the function depicted in Figure 3.3 determines the distribution of opacity throughout the volume. Lower density material, located toward the left side of the widget, is given low opacity. This is visible in the rendered image by means of the transparent outline showing the skin, the support braces from the CT scanner, and other low-density structures. Medium density material, mostly consisting of blood vessels and muscle tissue, is rendered opaquely and given a red coloration, so as to stand out from the image. Higher density tissue such as bone, located toward the right of the widget, is given high opacity values, which translates to the fully-opaque skull in the rendered image. By allowing users to freely manipulate such interface elements, a great degree of control is provided for the classification of volume data. The rendering shown in Figure 3.3 exhibits a pattern often encountered in volume rendering: as the focus changes from the outside toward the inside of the volume, the density of encountered materials generally increases, and the distribution of opacity ranges from fully-transparent to fully-opaque from external features to internal ones. It can be seen that, while the rendered image shows multiple different structures, it is not overloaded with information and the various illustrated anatomical features are easy to visually separate.

While a 1D data-based transfer function provides a significant amount of flexibility for classification purposes, it comes with its own set of limitations. Since it operates solely on scalar value differences, any two structures which share scalar values cannot be properly separated from each other. In other words, tissues with similar densities could not be differentiated using this approach. In the worst case scenario, one such tissue is in front of the other, therefore, when opacity is raised for the corresponding scalar values, the tissue futher away is occluded. There exist, however, other criteria for classification which allow to overcome the aforementioned limitations. In the following sections, we explore the use of transfer functions based around various voxel properties and show how these constitute efficient classification techniques.

3.1.3. Gradient-Based Classification

Gradient-based transfer functions operate on a property derived from the scalar values. Considering the volume function V from Equation 2.1, the gradient vector (*gradient*, for short) is the first derivative of this function. Since V is a three-dimensional function, its derivative is a vector with three components, containing the partial derivatives of V in the three directions x, y, z (Equation 3.3).

$$grad(V) = \nabla V(x,y,z) = \begin{pmatrix} \dfrac{\partial V(x,y,z)}{\partial x} \\ \dfrac{\partial V(x,y,z)}{\partial y} \\ \dfrac{\partial V(x,y,z)}{\partial z} \end{pmatrix} \tag{3.3}$$

A useful property of the gradient is that its direction indicates a change in scalar values, therefore pointing out a potentially important surface in the volume. The magnitude of the gradient signifies the degree of scalar value variation, thereby indicating the difference between the media located on either side of the surface pointed out by its direction.

For the purpose of this thesis, we estimate the gradient based on the *central difference* approach. The computation may be done in a pre-processing step as shown in Equation 3.4. The gradient is computed in position *(i, j, k)*, where *i*, *j*, and *k* are indices which iterate through the discrete volume V_d.

$$\nabla V(i,j,k) = 0.5 \begin{pmatrix} V_d[i+1,j,k] - V_d[i-1,j,k] \\ V_d[i,j+1,k] - V_d[i,j-1,k] \\ V_d[i,j,k+1] - V_d[i,j,k-1] \end{pmatrix} \tag{3.4}$$

In order to increase precision, all 27 values from the 3x3x3 cell around the target position may be used, i.e. all $V_d[i+i_x,j+j_y,k+k_z]$, where i_x, j_y, k_z = {-1, 0, 1}. This results in a smoother look for rendered images, especially when shading and illumination are involved. The downside of pre-computed gradients is the additional memory required to store the computed values. Pre-calculating gradients for an entire data set requires storing a three floating-point-component volume of the same resolution as the original data set. Depending on the hardware used, the storage requirements for such a volume may well exceed the available video memory. The advantage of using pre-computed gradients is that there is essentially no impact on performance during rendering, since obtaining the gradient can be done through a single texture look-up.

Unlike in the pre-computed case, dynamically-computed gradients are calculated on the fly, in the course of the rendering loop. In this case, the computation is carried out on the continuous, reconstructed volume. It is no longer restricted to discrete, fixed positions, and thus the neighboring voxels can be chosen by sampling at arbitrary distances around the target position. Similar to pre-computation, six neighbors can be used, two per axis, as shown in Equation 3.5. *x*, *y*, *z* are the coordinates for the target position while *d* is an arbitrary distance at which neighbor sampling takes place.

$$\nabla V(x,y,z) = \frac{1}{2d}\begin{pmatrix} V(x+d,y,z) - V(x-d,y,z) \\ V(x,y+d,z) - V(x,y-d,z) \\ V(x,y,z+d) - V(x,y,z-d) \end{pmatrix} \tag{3.5}$$

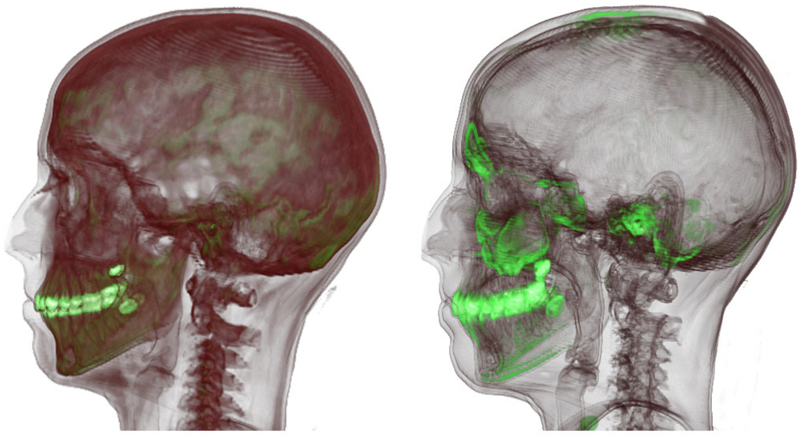


Figure 3.4. Classification using 1D transfer functions, based on (a) scalar values and (b) gradient magnitudes. Certain features within the volume are visible in (b), which could only be isolated using gradient-based means.

The dynamic estimation of gradients takes up virtually no extra memory, thereby being suitable for data sets of all sizes. Multiple neighbors can be used by sampling in multiple positions around the target. The downside lies in the extra computational effort required to compute the gradient on-the-fly. This may have a significant negative impact on frame rates, particularly for a large number of samples.

In our application, gradient-based classification is used to identify features in volume data which could not normally be isolated using scalar value-based means alone. Figure 3.4 shows two side-by side renderings. The image in Figure 3.4(a) is generated using scalar value-based classification. This allows the separation of anatomical components such as the skin, skull and teeth, but makes it difficult to properly classify certain structures inside and in the vicinity of the skull cavity, which are similar in density. Modifying the opacity and color of one such structure has the same effect on similar, neighboring structures, thus they cannot be properly separated. A different criterion is needed for the classification of these structures. Figure 3.4(b) shows a rendering of the same data set, using a 1D transfer function which operates on gradient magnitudes instead of scalar values (Equation 3.6). The image shows how using the gradient as a classification criterion makes it easier to emphasize the outlines of bordering surfaces, as opposed to showing solid structures.

$$\begin{aligned} &T_f : R \to R^4, \\ &T_f\left(\|\nabla V\|\right) = RGBA \end{aligned} \tag{3.6}$$

Gradient-based classification also makes it possible to identify previously-inaccessible anatomical features. While such structures may have overlapping densities, their surface properties are different, which allows their separation. Figure 3.4(b) shows certain features such as the sinus cavity and encephalon, which are highlighted through color. These share densities with other neighboring structures, however, the different nature of their bordering surfaces make them distinct.

3.1.4. Classification Based on Local Curvature

Curvature-based transfer functions have seen limited use in classification, with the most notable solutions being developed by Hladuvka et al., (2000) and Kindlmann et al., (2003). They rely on a rather complex convolution-based method which involves the computationally-taxing evaluation of the second derivative for the volume function. This is then used to assign color so as to bring out finer detail in surfaces and contours. Bruckner et al., (2007) suggest a solution based on evaluating the angles between gradient vectors along rays in a ray casting loop. The downside is that the evaluation can only be done in the direction of the ray.

Based on the result obtained by Bruckner et al. (2007), we propose a more flexible solution which provides a good approximation of local curvature. For a certain target position in the volume, the local curvature of the corresponding voxel is evaluated by assessing the differences among the orientations of neighboring gradients. The concept is shown in Figure 3.5. The greater the angle between neighboring gradients, the greater the local curvature in-between the neighbors.

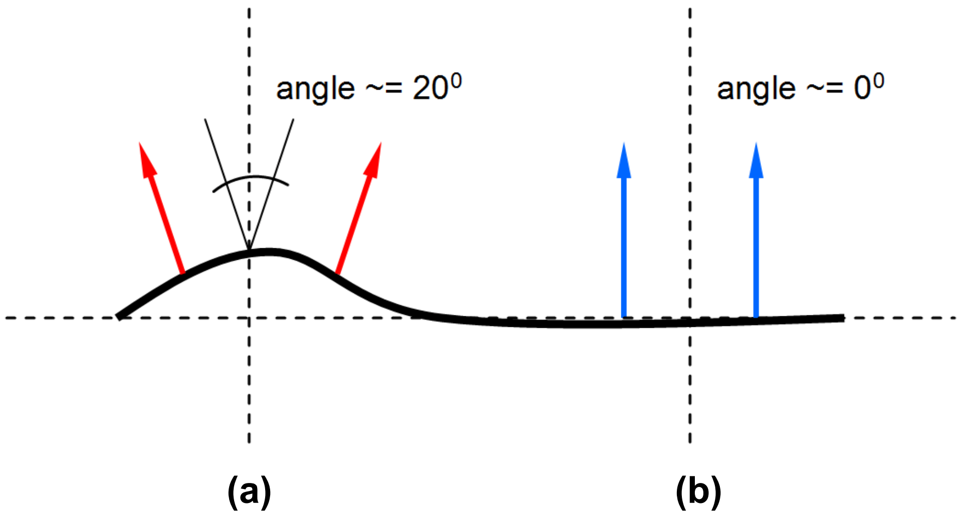


Figure 3.5. The evaluation of curvature based on gradient angles. In position (a), an angle between neighboring gradients indicates a degree of curvature in the intersecting isosurface, while in position (b) parallel gradients correspond to a flat surface.

We evaluate the curvature by sampling six neighbors around the target position, two for each axis, on either side. We then evaluate the angle between each pair of gradients by normalizing them and computing their dot product. The result is a three-component vector, whose magnitude is used to indicate the local curvature (Equation 3.7). While the method is not perfectly accurate due to the fact that the evaluation is not carried out along any particular isosurface, due to the limited resolution of the data set this approximation has proven sufficient for classification purposes.

$$k(x,y,z) = \left\| \begin{pmatrix} \nabla V(x+d,y,z)_n \cdot \nabla V(x-d,y,z)_n \\ \nabla V(x,y+d,z)_n \cdot \nabla V(x,y-d,z)_n \\ \nabla V(x,y,z+d)_n \cdot \nabla V(x,y,z-d)_n \end{pmatrix} \right\| \qquad (3.7)$$

In Equation 3.6, the curvature *k* at position *(x, y, z)* is the magnitude of the vector composed of dot products from normalized gradients, chosen at arbitrary distance *d*. The method may also be implemented in a pre-processing step, where *k* would be calculated from pre-computed gradients. This avoids an impact on performance, since the values for *k* are accessible via a single texture fetch.

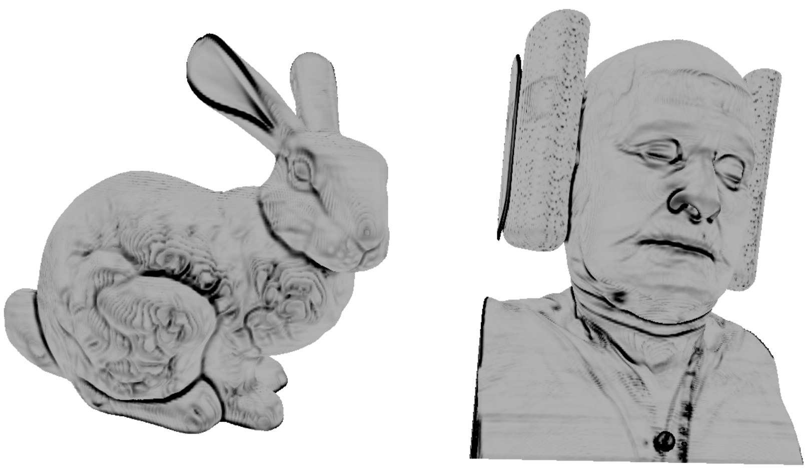


Figure 3.6. Two volume renderings, where local curvature is shown through grey values. Darker areas belong to surfaces with a higher degree of curvature.

Figure 3.6. shows two rendered volumes, where local curvature has been highlighted. Darker grey values indicate a greater degree of curvature. Areas which exhibit a high degree of curvature usually produce meaningful contours in the resulting images. Identifying such contours often helps to isolate certain structures with otherwise similar densities and surface properties.

3.1.5. Distance-based Classification

So far, the voxel properties used for classification were computed based on the scalar value alone. Distance transfer functions, however, take into account the spatial positions of voxels as an additional criterion which can be incorporated into the classification process. In our applications, we use distance transfer functions based on focal regions. These regions may vary from a simple point inside the volume, or a plane of arbitrary orientation, to more complex geometry, or even isosurfaces. The focal-region based classification method implemented in UniVolume is similar to techniques developed by Zhou et al. (2004) and Tappenbeck et al. (2007). Such transfer functions operate on the principle that voxels are discriminated and separated based on their distance to the focal region.

Distance transfer functions are limited in practical use on their own, because little advantage is gained from assigning color and opacity based solely on the distance from a point or plane, for instance. This is why we use these functions alongside other transfer functions, mostly data-based ones. A distance transfer function therefore modulates other functions, thereby introducing distance criteria into the classification process.

The method involves first pre-computing and storing the distances in a *distance volume*. This volume is retained in video memory, alongside the regular data set. The distance transfer function then operates in the scalar field defined by the distance volume, i.e. it assigns optical properties based on the values from the distance volume. Figure 3.7 illustrates the use of a distance transfer function. The example uses a focal point for distance computation, thus the resulting distance volume contains distances from every voxel to the focal point.

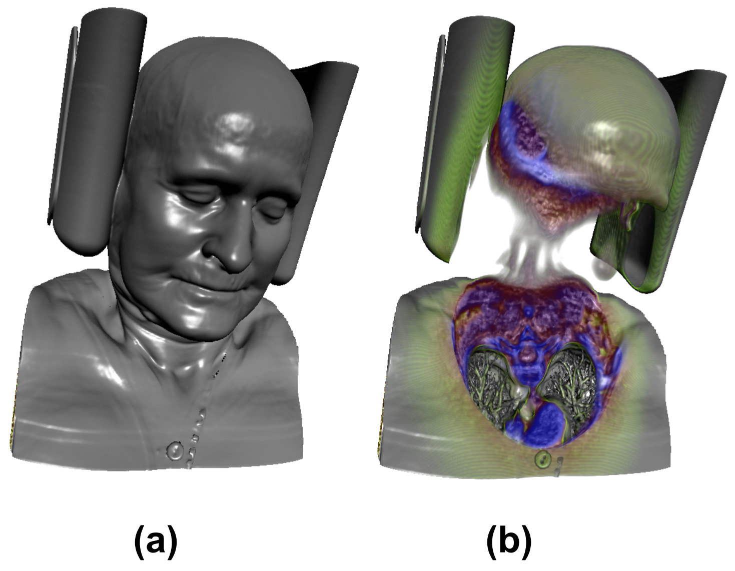


Figure 3.7. An image rendered from a volume dataset using (a) a regular scalar-based transfer function, and (b) a scalar-based transfer function which assigns opacity and color and which is modulated by a distance transfer function which operates on a focal point.

The scalar-based transfer function assigns color and opacity, while the distance transfer function modulates opacity based on the positions of the voxels in relation to the focal point. In this case, the position of the focal point is in the center of the volume. By using the distance transfer function a portion of the volume around the focal point is removed, thus revealing details such as cross-sections through various bones and organs.

An alternative to a focal point is the use of a focal plane. The distance volume now contains distances from every voxel to the plane in question, and thus voxels can be separated based on their positions from the focal plane. An example of the use of such a function is shown in Figure 3.8.

The images in Figure 3.8 are obtained, as previously, using a distance transfer function which modulates the opacity of a scalar-based transfer function. Groups of voxels located at certain distances from the focal plane are rendered transparently, thereby creating the "slicing" effect seen in the images. This technique can be used to successfully emulate clipping geometry. Setting up an array of clipping planes can be tedious, while a distance transfer function allows a quick assignment of clipping regions, using an interface element similar to the one in Figure 3.3. Unlike the rather similar method developed by Zhou et al.

(2004), the use of smoothly-interpolated transfer functions means that certain areas of the image can transition smoothly from opacity to transparency. The effect can be noticed in the images from Figure 3.8., where sharp cross-sections on the left side of the images can be generated alongside smoother transitions located more centrally, all from the same distance transfer function and a single control interface. Voxel distance provides a robust criterion for volume classification, while distance-based transfer functions are capable of achieving what their scalar and gradient-based counterparts cannot.

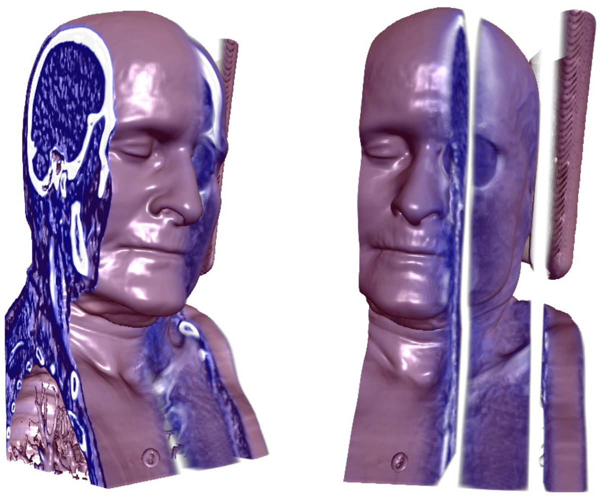


Figure 3.8. Two rendered images of the same volume, seen from different angles. A distance transfer function which operates on a focal plane is used to emulate multiple clipping planes.

3.1.6. Visibility-based Classification

The design of useful transfer functions is mainly held back by two factors: the increase in complexity as new dimensions and components are added, and the subjective factor involved in volume classification. In many cases, a developer has little knowledge about what users want to see in a rendered image, and which information from the data set they consider useful. The most important problem related to the subject of visibility-based classification is that, in order for it to work, visibility has to be quantitatively assessed. Once a measurable evaluation of visibility is available, a transfer function would then be used to map optical properties based on quantitative visibility.

The most significant results on the topic of visibility-based classification come from Correa and Ma, (2009) and Correa and Ma, (2011). Their approach of assessing visibility is to measure the occlusion level of objects inside the volume data. This is done by computing ambient occlusion, a property which takes into account contributions from neighboring voxels to the visibility of the target sample. This is a rather expensive method from a computational point of view, since it takes into account multiple neighbors, including those located behind the target structure.

We propose a simpler method for the assessment of visibility, which assumes that a ray casting-based method is used for rendering. As a ray enters the volume, the opacity of the sampled voxels intersected by it accumulates as shown in Equation 2.16. Let A be the opacity accumulated from the volume entry point to the position immediately in front of a sampled voxel v. A therefore represents the total accumulated opacity of the voxels in-between v and the viewer. Once A is available, the visibility of voxel v is $Vis_v = a_v(1 - A)$, where a_v is the opacity of v. The idea is that the visibility of a voxel is influenced by its own opacity and by the accumulated opacity of the voxels which occlude it.

The visibility values of the voxels constitute a voxel domain in which a transfer function may well operate. In our application, we use such a function to assign color based on the visibility of sampled voxels. We are specifically interested in highlighting less visible data, which may be easily overlooked when examining a rendered image. The result of such an endeavor is shown in Figure 3.9. The image from Figure 3.9(a) is generated with conventional scalar-based classification. A visibility transfer function is used for the image in Figure 3.9(b). The color mapping of this function is shown below the image, and it is set up so that low-visibility areas are highlighted. These areas include the transparent outer surface and various partially-occluded or low opacity structures, such as blood vessels, vertebrae and soft tissue inside the skull. These elements do not necessarily have common densities or surface properties, and are classified primarily based on their own transparency and level of occlusion. Visibility constitutes an important criterion for the separation and highlighting of important features which may be partially occluded due to erroneous or imprecise classification.

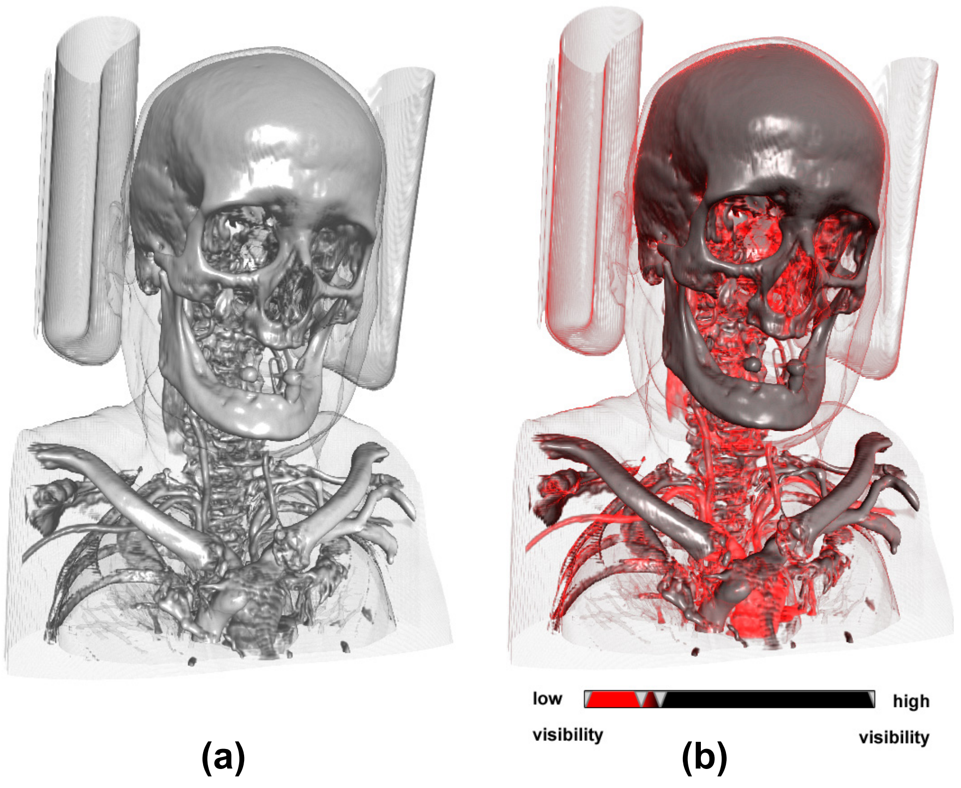


Figure 3.9. Images rendered with (a) a regular scalar-based transfer function and (b) a transfer function which maps color based on visibility.

3.1.7. Transfer Function Combinations

In the previous sections, we have explored the use of various local voxel properties for the classification of volume data. So far, however, these properties have been used separately, and no particular attention was paid to potential advantages gained from using multiple classification criteria concurrently. When two or more classification criteria are used together, in the same step, the result are multidimensional transfer functions, which map color and opacity based on multiple voxel properties. Multidimensional transfer functions constitute a powerful tool for the classification of volumes, because more criteria result in increased control over transparency and color. Such a transfer function might, for example, take into account both scalar value and gradient magnitude when assigning optical properties, thereby accounting for both the densities and surface properties of the materials in the volume.

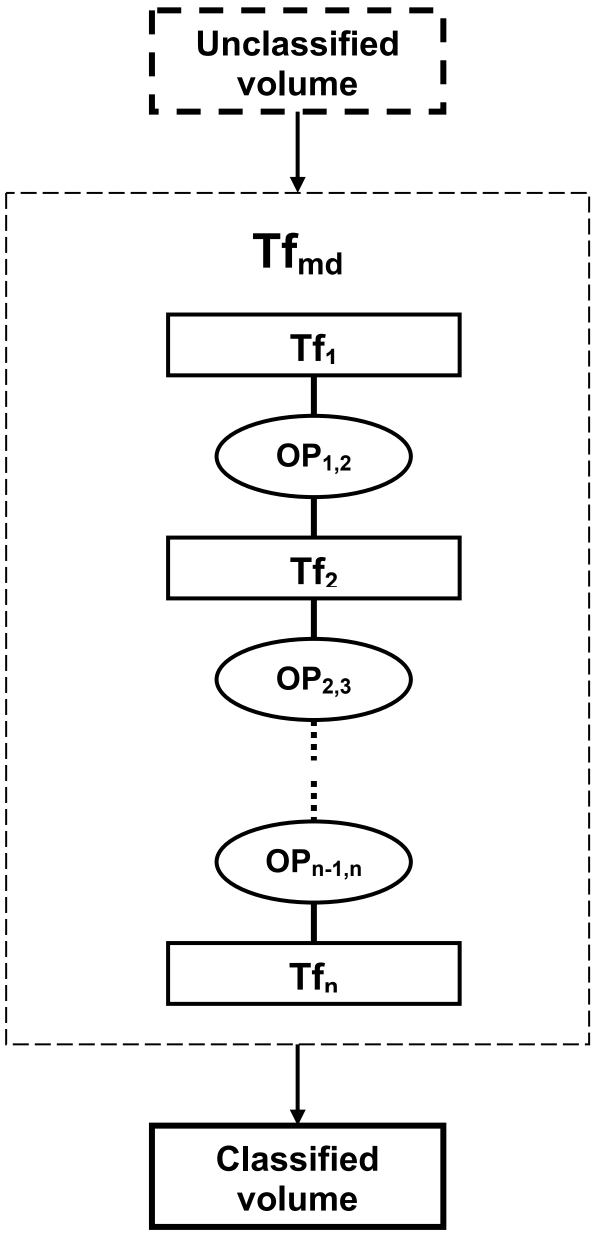


Figure 3.10. Concept of a multidimensional transfer function made up of a combination of 1D transfer functions, which are joined by various operators

While multiple methods exist for the specification of such functions (Kniss et al., 2002; Pinto and Freitas, 2007), in our application we opted for a simple yet effective specification technique. We consider the multidimensional transfer function to be made up of multiple individual 1D components, which are joined together by various relations. Once reconstructed through filtering and interpolation, the unclassified volume is processed through a series of 1D transfer functions, each constituting one dimension of the overall, multidimensional transfer function.

While the individual components could very well operate independently, they are connected to one-another by means of inner-relations. Such relations may range from simple, linear operators, to more complex ones which take into account the distribution of opacity as specified by the individual components. The concept is shown in Figure 3.10. Tf_{md} is a multidimensional transfer function made up of n components Tf_i, i = 1, ..., n. These are joined together by operators $OP_{i,\,j}$, i =1, ..., n, j =2, ..., n. Each component Tf_i, i =1,...,n constitutes one dimension of Tf_{md}. The components are each 1D transfer functions, and they operate on a different voxel property, such as the scalar value, gradient, or visibility.

We next provide several examples of the use of component-based multidimensional classification. One of the simplest ways to construct a multidimensional transfer function from components is to assign different sub-regions from the scalar value domain among two or several transfer function components. One way to divide the domain is to split it in two sub-regions using a threshold value. Two 1D components of the transfer function would then operate on each of the sub-regions. We refer to this classification method as a *thresholded transfer function*, and present the concept in Equation 3.8.

$$Tf_{th} = \begin{cases} Tf_g & scalar < th \\ Tf_d & scalar \geq th \end{cases} \tag{3.8}$$

The thresholded transfer function Tf_{th} is constituted from two components. The first, Tf_g, operates on the gradient magnitude, while the second, Tf_d operates on scalar values. A threshold *th* is set in the scalar value domain, thereby dividing it into two sub-regions. The first component, Tf_g, is applied in the sub-region containing the smallest scalar values, while the second one, Tf_d, is applied in the region with larger scalar values. Figure 3.11 shows two side-by side images, rendered from the same CT dataset. Two isosurfaces are shown in each image, one representing bone tissue, while the other softer one depicts skin tissue. The image in Figure 3.11(a) was rendered using a 1D scalar-based transfer function, while the image in Figure 3.11(b) was generated using a threshold transfer function, as described above. The threshold is set in-between the scalar values of the two isosurfaces. While using a single, 1D transfer function allows the separation of the two tissue types based on their different densities, the outer-most isosurface can only be highlighted to a certain extent before it starts to occlude the inner isosurface (Figure 3.11(a)). However, when a gradient based transfer function is used for rendering the outer isosurface, it becomes possible to better highlight this isosurface, while still allowing the inner isosurface to be completely visible. This is due to the fact that gradient-based methods are usually better suited for representing surfaces. The outer isosurface is more clearly visible, while the classification of the inner isosurface is not altered.

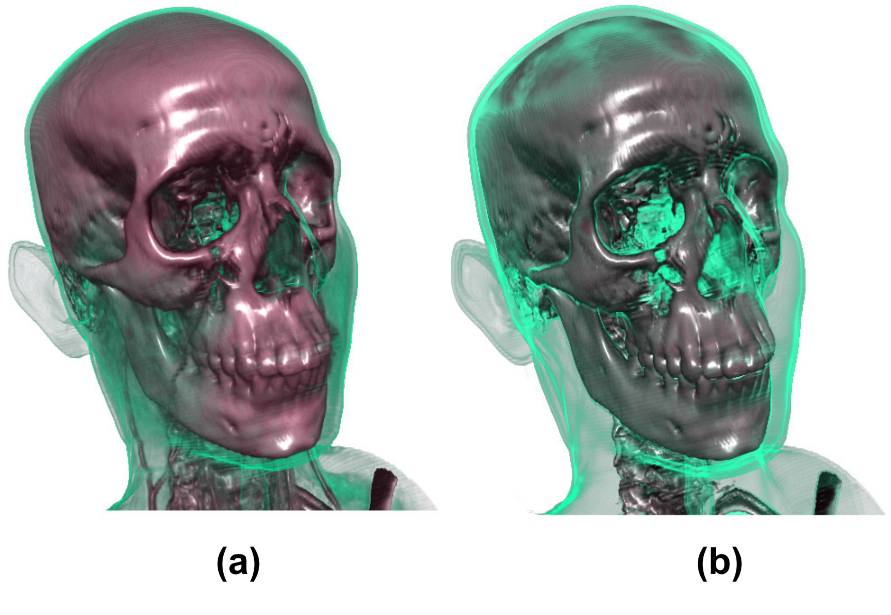


Figure 3.11. Renderings of a CT data set using (a) a 1D scalar transfer function and (b) a thresholded transfer function (Gavrilescu et al., 2011).

A more general type of multidimensional transfer function incorporates two or more voxel properties into the classification process. Equation 3.9 presents the generic definition for a function which incorporates three such properties, namely the scalar value, gradient magnitude and local curvature.

$$\begin{aligned} &Tf_3 : R^3 \rightarrow R^4, \\ &Tf_3(V, \|\nabla V\|, k) = RGBA \end{aligned} \qquad (3.9)$$

As a first example, we only use two of these three components to classify a CT data set (Equation 3.10).

$$Tf_2 = w_d \; Tf_d + w_g \; Tf_g \qquad (3.10)$$

The two components of the transfer function, Tf_d and Tf_g, operate on scalar values and gradient magnitudes, respectively, while w_d and w_g are weights used to regulate the influence of each component on the output.

In Figure 3.12, the idea is to remove the skin layer, thus completely exposing the brain tissue underneath. By using a two-component transfer function, skin and brain tissues can be classified separately, while the opacities for gradient values corresponding to the surface of the skin can be lowered. This way, as these gradient values are lowered, the skin starts to "peel" away, revealing the brain underneath.

In order to achieve a similar result for MRI data, we have found a two-component transfer function to be insufficient for the proper classification of brain tissue. This is due to the more problematic nature of MR images, where densities may overlap and where the noise content is significantly higher. For this reason, it was necessary to introduce a third component into the classification process, namely local curvature (Equation 3.11).

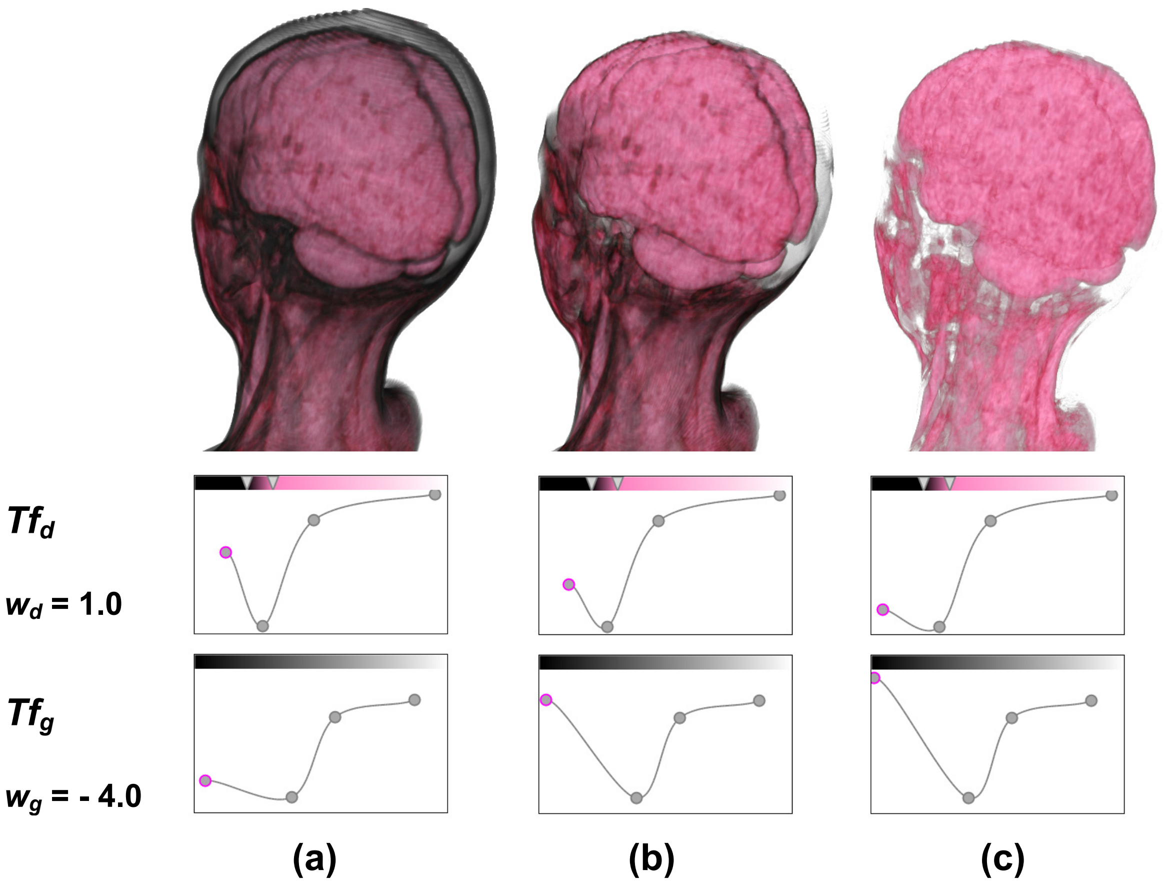


Figure 3.12. Images of a CT data set rendered with a two component transfer function. The images below the rendered outputs indicate the shapes of the transfer function components Tf_d and Tf_g which operate on scalar values (top row) and gradient magnitudes (bottom row); (a) the skin which is fully visible is (b) partially removed and (c) completely removed (Gavrilescu et al., 2011).

$$Tf_3 = w_k\ k\left(w_d\ T_{fd} + w_g\ T_{fg}\right) \tag{3.11}$$

Results are shown in Figure 3.13. In the image from Figure 3.13(a), a 2D transfer function has proven insufficient for proper removal of the skin. Once curvature is involved, the skin layer is removed due to differences between the shape of the skin layer and the brain tissue. The latter exhibits greater local curvature values.

The methods presented thus far come with advantages and drawbacks. The main disadvantage of such an approach is that it requires the tweaking of several parameters. This can, however, be compensated by an intuitive user-interface. One advantage of this method is that it uses known, tried-and-tested interfaces for the specification of the 1D components (as shown in Figure 3.3). Another advantage arises from the practicality of the implementation. Each component can be separately uploaded to video memory as a 1D texture. This means, among other things, that a theoretically unlimited number of dimensions can be added to the transfer functions, without significant changes to the code or specification procedure.

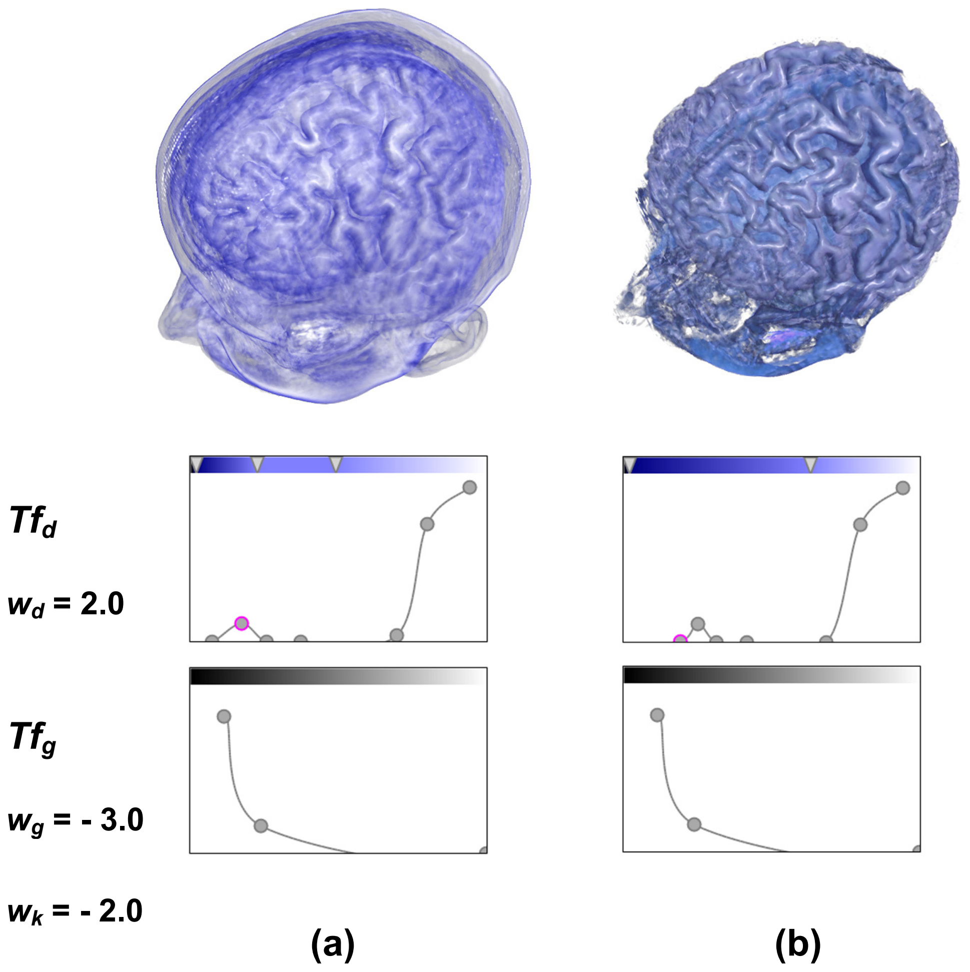


Figure 3.13. Classification of brain tissue from MRI data. (a) a regular transfer function is insufficient to remove the skin layer. (b) with curvature modulation, the skin layer is removed almost completely (Gavrilescu et al., 2011).

3.1.8. Applications for Segmented MRI Data

The high noise content of MRI data often makes it more difficult to classify than, for example, CT images. Often, segmentation is used to highlight certain features in the data, which could not be correctly identified otherwise. Figure 3.14 shows an example of a rendered MRI data set, where a tumor has been manually segmented.

The tumor has been manually segmented and therefore it is easy to identify and highlight using a separate transfer function. However, when producing images from data with a lot of underlying structures, it is important to draw focus to the information of interest, while also clearly showing context information. Figure 3.14(b) clearly highlights the tumor, but it is difficult to show other, unsegmented structures alongside it, due to the noisy nature of the data set used. Increasing opacity for certain density regions brings forth multiple occluding structures, producing undesired results. It is, however, possible to achieve results of reasonable quality using multidimensional transfer functions.

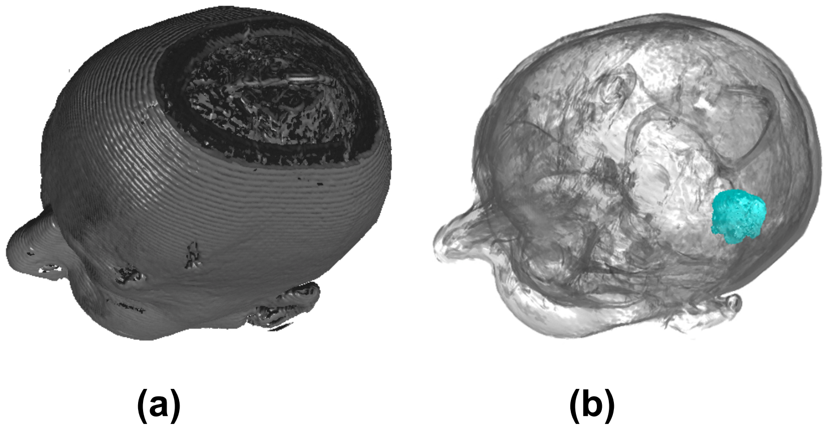


Figure 3.14. Rendered images of an MRI dataset showing (a) an opaque isosurface and (b) a semi-transparent rendering where a segmented tumor has been highlighted (Herghelegiu et al., 2011).

A result is shown in Figure 3.15, where a 3-component transfer function which operates on scalar value, gradient magnitude and curvature is used to highlight context information alongside the segmented tumor. However, it is not yet possible to properly highlight the outermost isosurface (i.e. the surface of the skin) without occluding important information (such as the tumor).

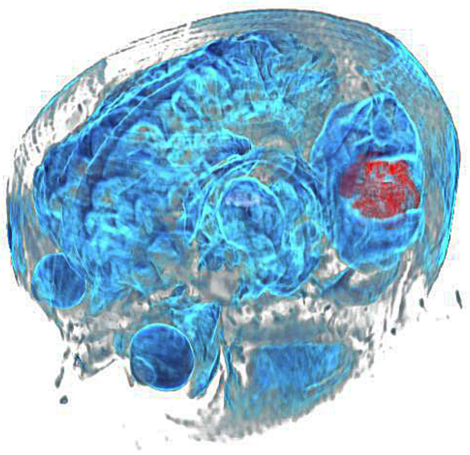


Figure 3.15. An image rendered using a multidimensional transfer function, where a segmented tumor is highlighted among context information (Herghelegiu et al., 2011).

The problem gets more complicated when multiple segmented data is available. In order to overcome this, we use a lens tool to highlight the segmented information, while still preserving details from the unsegmented data. Figure 3.16 shows the result of such an endeavor.

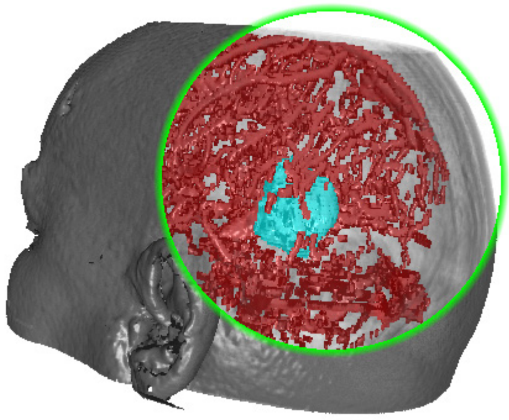


Figure 3.16. A lens which shows segmented information inside an opaquely-rendered isosurface (Herghelegiu et al., 2011).

The data set now contains a lot of segmented information, namely the previously-mentioned tumor and cerebral vasculature. Here, we use a lens tool to show segmented structures from the dataset. Outside the lens, the outermost isosurface is rendered opaquely and provides clearly-outlined context information. Inside, the opacity of the isosurface is scaled down and the segmented structures are revealed. These can be highlighted using separate transfer functions. The lens can be manipulated in multiple ways: it can be moved using the mouse or a touch-screen interface, or it can be snapped to a region or structure such as the tumor. Thus, the information of interest can be brought into focus using the lens, while context information provided by the rest of the dataset is preserved. This method does not suffer from the noise and artifacts of previously-mentioned approaches.

3.2. Enhancement of Features in Volume Data

While classification by means of transfer functions is flexible enough, that many structures and features of the volume data are identifiable through color and opacity, the visualization of such data can be improved upon by means of techniques which bring out the finer aspects in rendered images. These serve to bring focus on harder-to-see details, or otherwise improve perception. In this section, we address two issues related to feature enhancement: on the one hand, we show how volume outlines may be emphasized through specialized techniques. On the other hand, we develop a technique which accentuates the edges of volume features in rendered images, based on post-rendering filters. These techniques work independently from classification and do not generate new information. Instead, they are solely used to highlight already existing and potentially useful information which may otherwise not be visible enough after classification.

3.2.1. Display of Outlines

Outlines are contours which surround and highlight features or structures in volumes, thereby drawing focus upon them and making them more visible. These are frequently used in illustration, where they help trace the shape of objects and enhance the visualization of data.

The main criterion for identifying the outlines of a structure is the direction in which its surface is facing. Specifically, the outline is traced by regions of the surface which face away from the viewer . This can easily be evaluated through the dot product between the view and gradient vectors. The outline is drawn (for example by darkening the surface) if the viewing vector and gradient are almost orthogonal, i.e. their dot product is close to zero (Neumann et al., 2000). Kindlmann et al. (2003) suggest a method of adjusting the thickness of outlines based on local curvature. If the dot product between viewing vector *v* and gradient *g* is smaller then *mDot* as defined in Equation 3.12, then the corresponding sample is considered to be on the outline. *k* is the local curvature, which can be computed as shown in Equation 3.6, and *T* is the user-assignable thickness of the outline. *k* is clamped to *1/T*, in order to ensure that smaller features are not overlooked.

$$mDot = \sqrt{Tk(2 - Tk)} \tag{3.12}$$

Aside from thickness control, we add the capability of controlling outline smoothness, by interpolating between the original color of the surface and an outline color which is established in advance (most commonly, it is black).

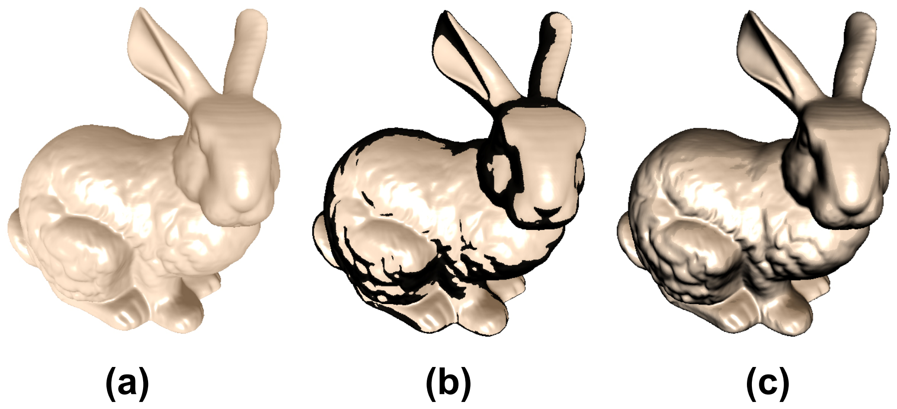


Figure 3.17. Rendering of a volume with: (a) no outlines; (b) outlines with sharp thickness; (c) outlines with smooth thickness.

The outlines in Figure 3.17(b) define a contour around the volume, though due to the sharpness of the contour certain details are harder to distinguish. In the image from Figure

3.17(c), a smooth outline highlights the features of the rendered isosurface while preserving the finer surface details.

3.2.2. Detail Enhancement in Post-processing

In the course of the rendering pipeline (Figure 2.7), once all the fragments have been computed and blended, the resulting image is rendered to a texture attached to a frame buffer object. This provides access to all pixels in the image, before it is sent to the video framebuffer for displaying on screen. In the post-processing stage, various image-processing techniques may be applied to further improve or enhance the resulting image.

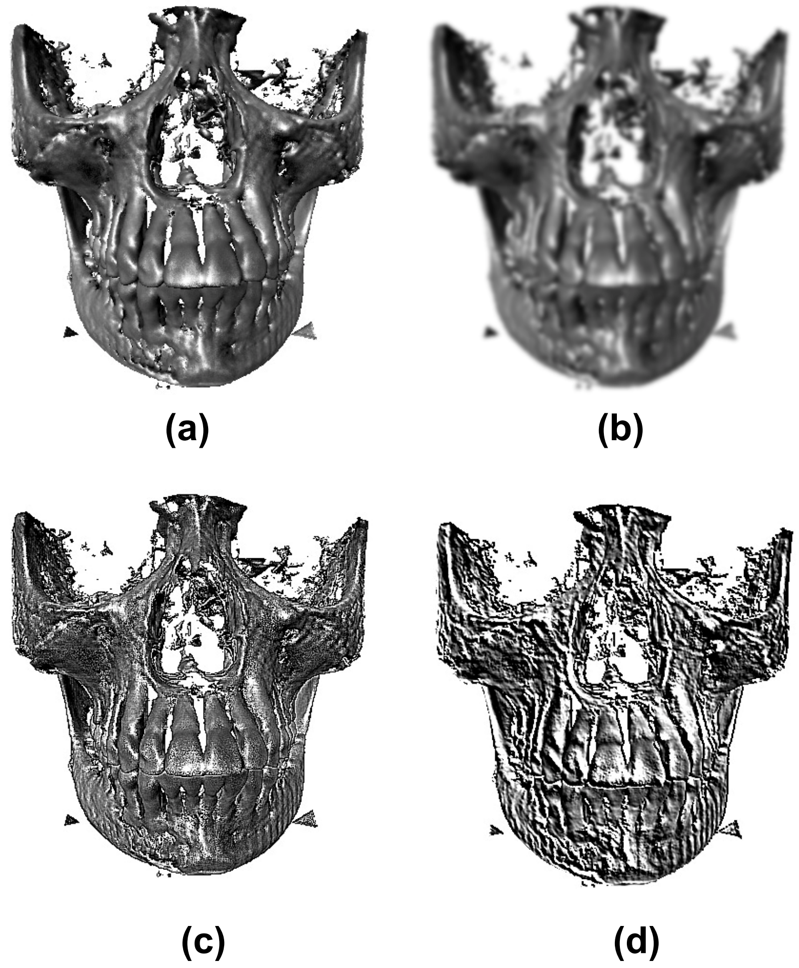


Figure 3.18. Use of various post-processing real-time filters: (a) unprocessed image; (b) blurring filter; (c) sharpen filter; (d) emboss filter (Gavrilescu et al., 2009).

The gallery in Figure 3.18 illustrates a few uses of common filters. The low-pass and high-pass filters in images (b) and (c) create blurring and sharpening effects, respectively, while an embossing filter is responsible for the "relief" look of image (d). These filters are applied to the texture which contains the rendered image during the post-processing stage. The computational capabilities of modern GPUs allow real-time frame rates for filtering kernels as large as 7x7. This allows to tweak the aspect of resulting images in real-time without interfering with the rendering process.

A problem which is sometimes encountered in rendered images is that, due to the interpolation and blending used during the rendering process, the resulting images may have a "washed-out" look, and certain finer details might not be readily visible. In order to circumvent this problem, we use a post-processing filter which involves two steps:

- the first step involves convolving the rendered image with a high-pass filtering kernel, defined as follows (Petrou and Petrou, 2010):

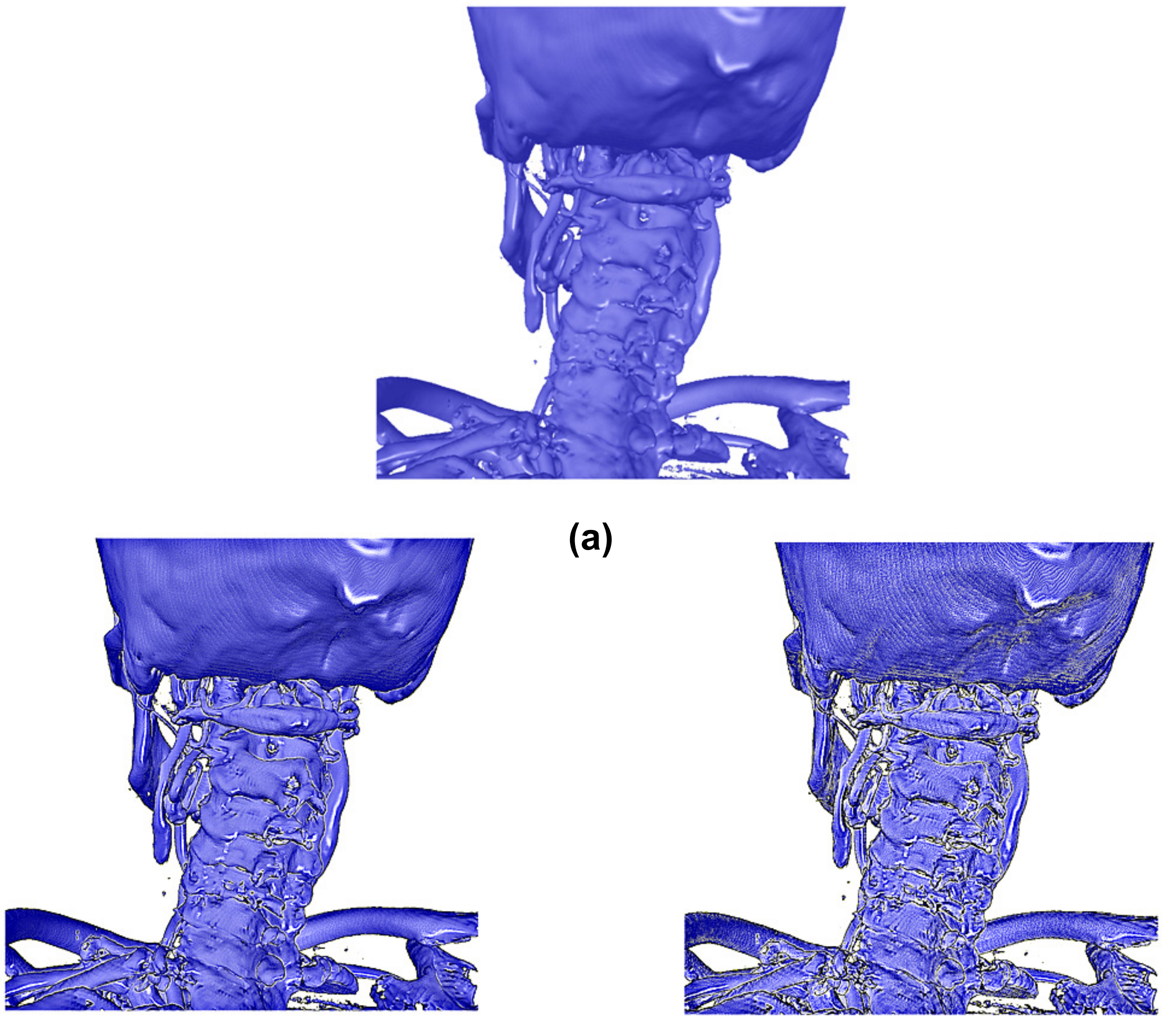
$$\begin{pmatrix} -m/8 & -m/8 & -m/8 \\ -m/8 & m & -m/8 \\ -m/8 & -m/8 & -m/8 \end{pmatrix}$$

where *m* is a user-adjustable parameter.

- the second step involves using screen blending to combine the original color c_0 with the filtered color c_1:

$$finalColor = 1-(1-c_0)\,(1-c_1)$$

The combination of high-pass filtering and screen blending is referred to as the *hard light* blend mode, similar versions of which are used in high-performance image processing software to achieve comparable results for 2D images. A result is illustrated in Figure 3.19. In image (b), the filtering method provides a sharper look with more visible details, such as the contours of the vertebrae. In image (c), higher values of parameter *k* were used, for an exaggerated effect. While this further enhances edges and sharpens the image, it also introduces noise. Therefore, the filtering methods need to be used in moderation, depending on the level of detail and clarity of the original image.



(a)

(b) (c)

Figure 3.19. Illustration of post-processing image enhancement. (a) unprocessed image; (b) subtle filtering provides a clearer image with slight edge enhancement and a sharper look (c) exaggerated filtering further enhances edges, but introduces noise

Conclusions

In this section we have explored, developed and illustrated various techniques for the classification and enhancement of volume data. One of the most effective and frequently used classification methods in volume graphics involves the use of transfer functions. These operate in various voxel spaces, resulting in numerous classification possibilities. Depending on the classification criteria used, tissues may be separated based on density, surface properties or the distance from a target geometry. An efficient way of computing local curvature allows features like depressions and ridges in surfaces to be better highlighted. The visibility of voxels is assessed by accounting for the level of occlusion of voxels located on the same ray. It is then used to highlight features which would normally be less visible. Transfer function specification is done by means of intuitive interface elements which allow users to dynamically specify color and opacity components, while the use of a cubic spline for the shape of the transfer functions allows for a smooth transition between classified regions. The possibilities of classification are extended by means of multidimensional transfer functions. We specify such functions by individually controlling their dimensions via individual 1D components. Thresholded transfer functions allow the separate highlighting of soft density surfaces, while more generic types of functions allow improved control over the opacity and color of the structures within the volume. Feature-enhancement techniques work alongside classification to further improve the quality and detail of the resulting images. The depiction of outlines shows contours which trace the shape of surfaces from volumes. Image processing techniques are used in the post-processing stage of the rendering pipeline to further enhance the image once rendering-related computations are complete. An image processing method based on high-pass filtering is used to make rendered images sharper and, clearer looking, while enhancing the edges of fine structures. The classification and enhancement of volumes constitute powerful means of generating high-quality expressive images where meaningful information is visually conveyed in an intuitive manner.

4. Visualization of Multidimensional Cardiac MRI Data

The continuous advances in cardiovascular medical research of recent decades mean that there is a growing demand for more advanced and improved tools for imaging and the interpretation of images, to aid in the assessment of heart-related issues. MRI images allow medical staff to better examine anatomical features and to handle medical conditions which may not be as easily-detectable using other imaging techniques, such as CT or ultrasound (Taylor et al., 2005). The segmentation of cardiac data also plays an important role in proper diagnosis. However, the traditional interpretation of MRI, which is carried out by examining individual grayscale slices, can be a tedious and time consuming task (Hashemi et al., 2004). Multidimensional MRI data may comprise several phases each with multiple slices, thereby requiring increasingly sophisticated, automated and intuitive means of representing the data. Furthermore, the information in individual slices is largely inaccessible to patients who are not trained in the interpretation of MR images, thus requiring either extensive explanations, or tools which convey the information in a more intuitive, comprehensive manner. The development of such tools has been made possible by the progress in GPU technology, which, although adapted for fast 3D rendering, can also be used for other types of processing (Gavrilescu and Manta, 2008).

In this section, we propose methods for the visually-intuitive representation of cardiac MRI. The data under analysis are 4D segmented MRI data sets of the left ventricle (LV), which is the most solicited chamber of the heart and therefore the most prone to cardiac conditions. Our method involves the computation and display of parameters derived from segmentation contours which outline the epicardium and endocardium. Four parameters are first computed and then displayed on 3D surfaces constructed from segmentation contours, which closely approximate the myocardial wall. These parameters are: the thickness of the myocardial wall, the thickening of the wall, the speed of motion and the moment of maximum thickness. Once computed, the parameter values are represented on the surfaces using color coding, allowing them to be interpreted across multiple slices and phases. We also develop a method for displaying information from 5D (five-dimensional) data, which, aside from the three Euclidean dimensions and the phase of motion, also encompasses several stress levels.

The method makes it easier to visually inspect the data and see anomalies in various areas of the myocardium, without necessarily requiring training in radiology. It addresses both medical staff, who may benefit from viewing data from multiple slices in the same image, and patients, who, due to the intuitive character of the method, are able to better understand details regarding their pathology.

4.1. Medical Overview

The heart consists of two pumps, each supplying blood to the lungs or the entire body. Each pump contains an atrium and a ventricle. The ventricle situated in the left pump (the LV) supplies the whole body with oxygenated blood. It therefore has the thickest and strongest wall (Bary, 2006). It is therefore the most important chamber of the heart and the one which is under the greatest stress when in comes to maintaining blood circulation. Even though the myocardium is adaptable to significant changes, such as variations of the inflowing blood volume (Iaizzo, 2005), it may become affected by various factors, both internal and external, which may cause it, or areas of it, to deviate from normal operation. Such problems may easily evolve into more serious medical conditions, such as ischemia, coronary artery disease, or myocardial infarction. Potential symptoms of these problems can be visually assessed from multi-dimensional cardiac data sets, which include several motion phases and stress levels.

Modern cardiac MRI scanners acquire several images, using various protocols and imaging directions. Cardiac MRI works on the same principles as other medical MRI techniques. In the case of 4D data sets, the cine protocol is used to scan several phases within a heart cycle. A set of orthogonal images are taken per phase, which are used to assess a series of physiological parameters that influence ventricular functionality. The quantification of these parameters requires contours which delineate the surfaces that border the left-ventricular wall, namely, the endocardium and epicardium. The contours are fully segmented using various semi-automated techniques (Hautvast, 2004; Hautvast et al., 2006; Feng et al., 2009) . These contours are closed and fully encompass the LV wall (Figure 4.1). A pair of such contours is available for each slice in the data set (Figure 4.1).

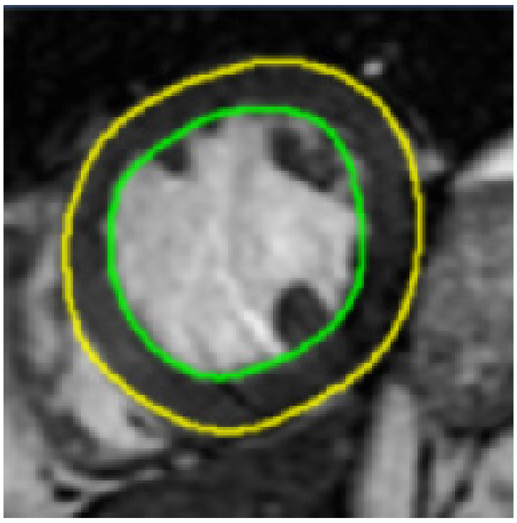


Figure 4.1. Contours segmented on a slice from an MR data set; the outermost contour corresponds to the epicardium, while the innermost one delineates the endocardium (Barry, 2007).

4.2. Cardiac Parameters

The data used for processing consists of several MRI data sets, each comprising several slices. Each data set corresponds to a phase of the cardiac cycle. Therefore, the complete data set is four-dimensional, as it also includes the temporal dimension. The slices are pre-segmented and the epicardium and endocardium are outlined by an epi-contour and an endo-contour, respectively (Figure 4.1.). First, several parameters are computed, based on the two contours. These parameters characterize the status and functionality of the LV wall, and any abnormal values may indicate a potential heart problem.

4.2.1. Wall Thickness

Wall thickness is calculated between pairs of points on each of the contours, in every phase of motion. The thickness values result from the smallest distance between the epi- and endo-contours. In order to compute this parameter, the centerline in-between the contours is first needed. The simplest solution for determining the centerline is to find the middle between the points from the segmentation mask with the same indexes. The centerline would then result from interpolating between the middle points. This approach is computationally effective, but has the drawback of not producing satisfactory results if the contours exhibit concavities. Therefore, we use a computationally more demanding algorithm, but one that produces better results. The steps required for centerline computations are as follows:

- first, the epi- and endo-contours are uniformly sampled, resulting in a set of evenly-spaced points;
- for every point on the epi-contour, the shortest distance to the endo-contour is computed;
- a contour C1 is obtained by generating a cubic spline from the means of the above distances;
- for every point on the endo-contour, the shortest distance to the epi-contour is computed;
- similarly to C1, another contour, C2, is generated by interpolating between the means of the above distances.
- repeat the above with C1 and C2 as starting contours.

With each iteration, contours C1 and C2 end up increasingly closer to one-another. The algorithm converges to the point where C1 and C2 coincide with the desired centerline. In our implementation, 3 iterations were sufficient for C1 and C2 to be essentially identical, and at the middle locations in-between the epi- and endo-contours. Thus, either one of them could be considered the centerline.

Once this centerline is obtained, the wall thickness is computed as the distance between the intersections of the normal of the centerline and the epi- and endo-contours (Figure 4.2).

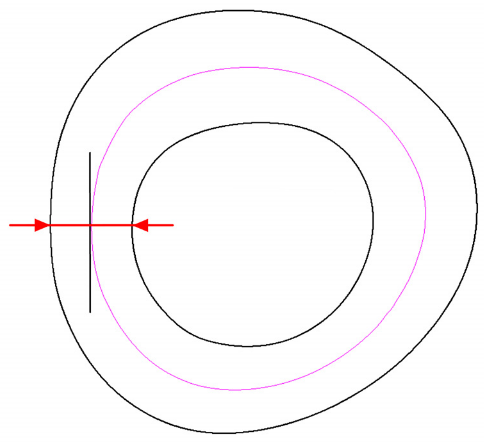


Figure 4.2. Wall thickness computed from the intersection of the centerline normal with the two contours (Gavrilescu and Manta, 2010).

4.2.2. Wall Thickening

This parameter constitutes the gain in thickness over a cardiac cycle. It reflects the extent to which a point or an area of the LV wall increases or decreases in thickness across all phases (Equation 4.1).

$$Wt = \max(WT_i) - \min(WT_i), \quad i = 1 \ldots P \qquad (4.1)$$

where Wt is the wall thickening, WT_i is the wall thickness in phase i, and P is the number of phases.

4.2.3. Moment of Maximum Thickness

This parameter is computed as follows: first the phase in which the LV wall reaches maximum thickness is computed for the entire data set. The resulting array of values is sorted and the median value is chosen. Then, the deviation from the median is chosen as the value of the parameter. The idea is to detect regions which are not properly synchronized with the rest of the wall, i.e. which attain maximum thickness earlier or later than the rest. Thus, any unsynchronized regions can be identified accordingly.

4.2.4. Wall Speed

This parameter denotes the contraction and dilation velocity of the wall. Either of the three contours, namely, the epi-contour , endo-contour and the centerline, can be used to compute the speed. We choose the centerline, since it constitutes a good compromise between the short motion range of the endo-contour and the much more dynamic epi-contour. The speed of a point on the centerline is computed as the total distance traveled during a cardiac cycle over the number of phases. Considering a point *C(x, y, z)* on the centerline, if *(x_i, y_i, z_i)* are the positions of the point in phase *i*, *i = 1..P*, then the speed S_C of the point is calculated as shown in Equation (4.2).

$$S_C = \frac{\sum_{1}^{P} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}}{P} \tag{4.2}$$

4.3. Visualization of the Parameters

Once parameter values are available for all slices, we use visualization methods to represent them, in order to facilitate their visual interpretation. As opposed to previous chapters, we do not rely on direct volume rendering, but rather on indirect representation of information from the data set via polygonal geometry. Specifically, the geometry used for representation is made up of two elements:

- two 3D surfaces, corresponding to the epicardium and endocardium, which are generated from the segmentation contours.

- a bull's eye plot, which is a 2D diagram commonly used in medical visualization for the representation of cardiac data (Termeer, 2009).

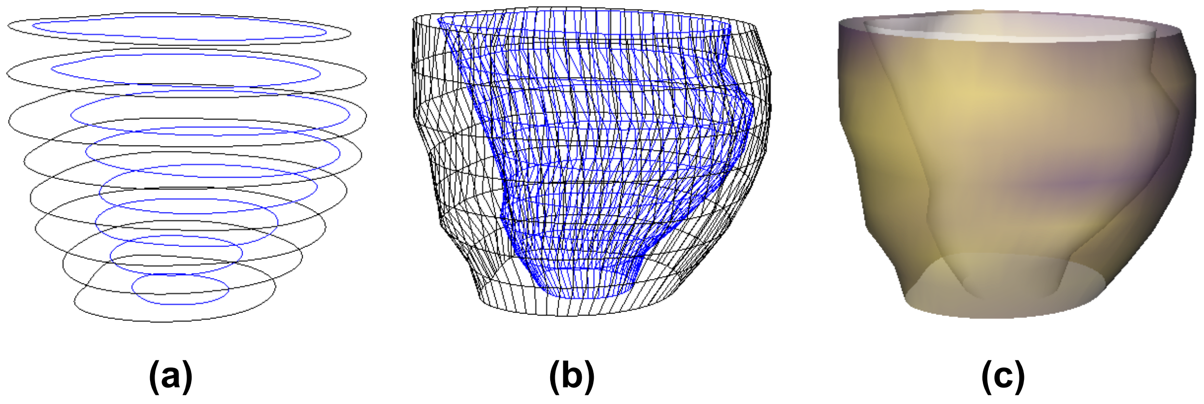


Figure 4.3. Generation of the surfaces corresponding to the epicardium and endocardium; (a) the segmentation contours from the data set; (b) wireframe meshes which correspond to the epicardium and endocardium; (c) the two surfaces with shading

The surfaces are generated as depicted in Figure 4.3. Starting from segmentation contours (Figure 4.3(a)), two meshes which approximate the shape of the LV wall are built (Figure 4.3(b)). Continuous surfaces are then generated from regular quads (Figure 4.3(c)).

The surfaces can now be used to display the values of the aforementioned parameters, as shown in Figure 4.4.

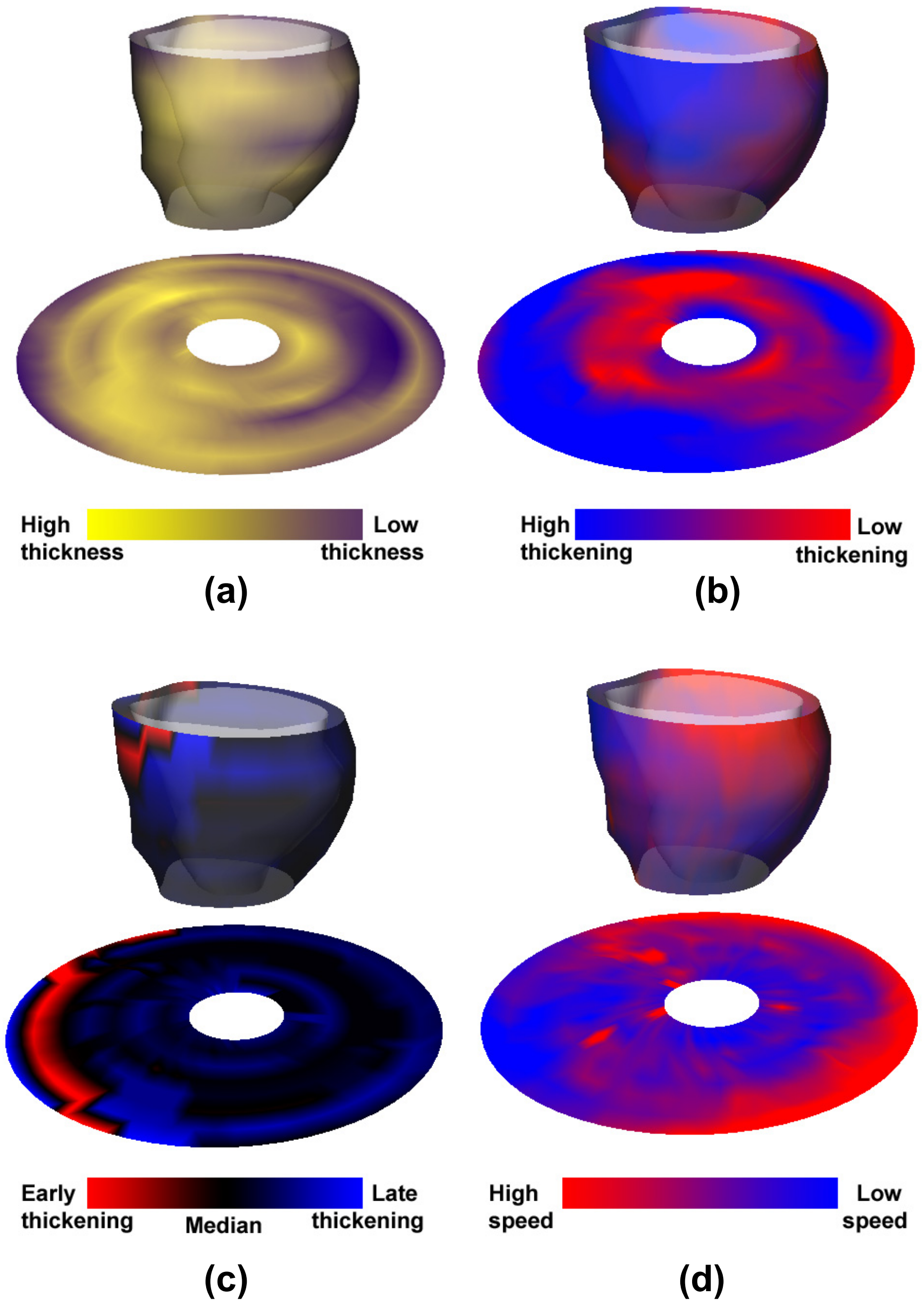


Figure 4.4. Visualization of cardiac parameters on epicardium and endocardium surfaces, and on bull's eye plots: (a) thickness; (b) thickening; (c) moment of maximum thickness; (d) speed.

The parameter values are mapped to colors and displayed on the epicardium surface and on the bull's eye plot. Thus, the distribution of parameter values can be visually assessed for all slices. We also use various color mapping schemes to better highlight details and transitions between parameter values (Figure 4.5).

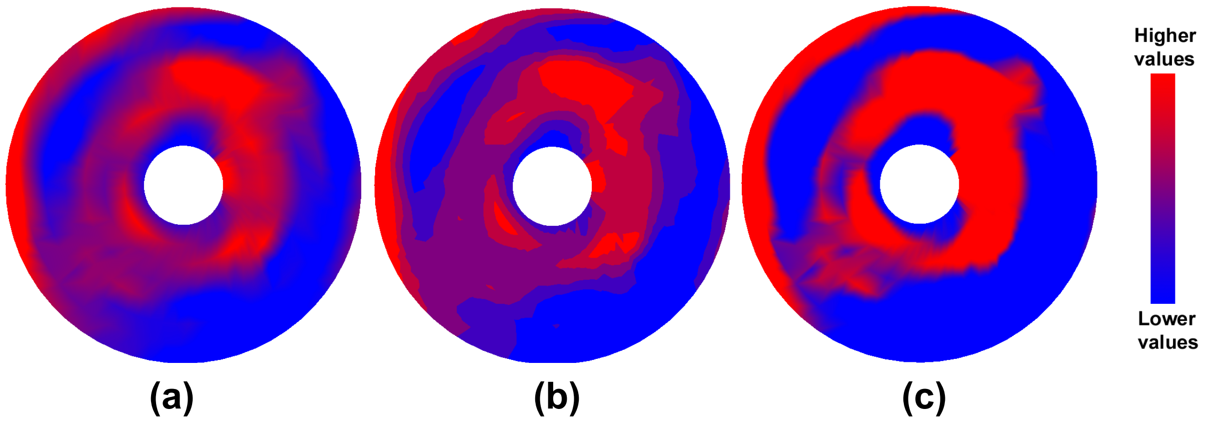


Figure 4.5. Various mappings used for assigning color to parameters: (a) linear mapping; (b) discrete mapping; (c) smooth step mapping.

Three types of color mapping are considered:

- linear mapping provides a smooth transition between colors. In Figure 4.5(a), wall thickening values are represented by linearly interpolating colors from a red-blue range; red regions correspond to high thickening values, while blue regions represent lower ones; one particular advantage of linear mapping is the compatibility with older graphics hardware which may not support more recent shader models;

- discrete mapping displays parameter values *p* using a set of color values $\{C_1, ..., C_N\}$, which are sampled from an established color range (Figure 4.6(a)). In Figure 4.5(b), five colors are sampled from a red-blue range, and wall thickening values are snapped to these colors when displayed. This mapping provides a discontinuous aspect of the rendered image. The advantage of such a mapping is that it emphasizes areas which may be difficult to notice if a smoother color transition were used, such as the red regions in Figure 4.5(b) which show areas of high thickening.

- smooth step mapping assigns color values using a Hermite curve interpolation (Gonzales and Woods, 2008), bounded by two constants (Figure 4.6(b)). Smooth step interpolation accentuates regions of transition between distinct parameter values, making them easier to analyze (Figure 4.5(c)). The bounds of the Hermite spline, p_{left} and p_{right}, can be adjusted in order to regulate the emphasis of transitional regions. If $p_{left} = p_{right}$, the smooth step becomes a regular step function and only the most significant transition is shown.

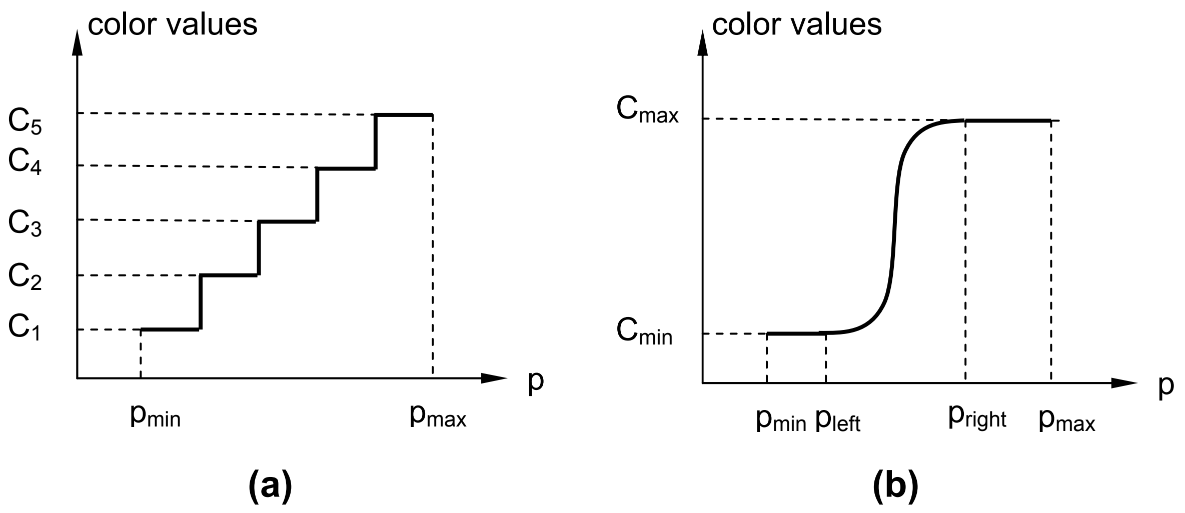


Figure 4.6. Two mapping functions used to assign colors to parameter values *p*. (a) discrete; (b) smooth step.

4.3.1. Concurrent Parameter Visualization

The simultaneous representation of multiple parameters in the same image is important for providing more thorough information on the nature of the data and the potential medical conditions that can be derived from observing the distribution of parameter values. However, the representation of more than one parameter should not overload the image with too much visual information, thereby making the data difficult to interpret. For example, many existing methods rely on glyphs to encode parameter values (Meyer-Spradow et al., 2008; Ropinski and Preim, 2008). We propose a different approach, which allows the concurrent visualization of two parameters using color-coding for both. Specifically, we use a lens tool to decide which parameters are represented on the epicardium and bull's eye plot. The lens is a circular region in which the displayed information may change in terms of parameters or the distribution of colors used. For our purposes, the idea is to display one parameter outside the lens, and the other inside of it, using different color ranges for coding the values. The concept is illustrated in Figure 4.7, where wall thickness and wall thickening are represented outside and inside the lens, respectively. The properties of the lens (its position and radius) can be controlled interactively, thus allowing for the real-time inspection of the parameters.

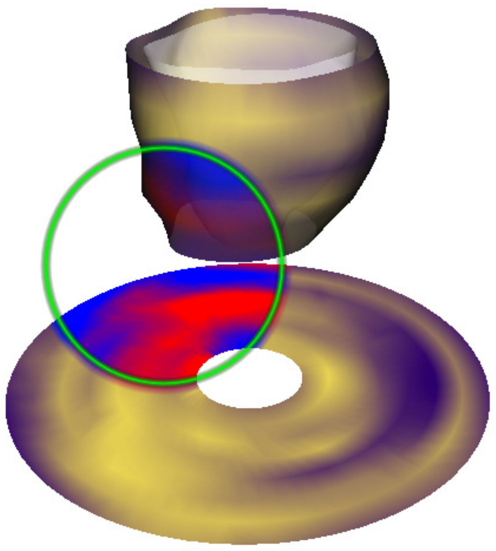


Figure 4.7. Visualization of two parameters using a lens tool and two different color ranges.

4.3.2. Visualization of Stress Levels

MRI stress imaging is capable of reproducing each position on the slices for several stress levels (Marwick, 1996). In each stress level, the LV wall is solicited at different extents to test for conditions such as abnormal blood flow, ischemic heart disease or the probability of myocardial infarction. Each slice in the data set is available over multiple stress levels, which constitute a fifth dimension of the data. In order to properly assess the influence of different stress levels on the functionality of the myocardial wall, a comparison between the wall parameters across the stress levels is required. However, examining stress levels individually can become a tedious task, due to the large number of slices which have to be examined (several slices per phase, several stress levels). It would therefore be useful if the stress levels could be examined simultaneously for each slice. Our solution involves the display of wall parameters for every slice, on a single surface. The data set depicted in Figure 4.8. contains three slices, and four stress levels per slice. In order to render the parameter values simultaneously and display them on a single image, we use *stress bands* to show the values for all stress levels, over all slices. Each slice has four bands drawn around it, while each band shows parameter values for one of the four levels. This way, the parameter can be comparatively analyzed across all stress levels and slices (Malik et al., 2010).

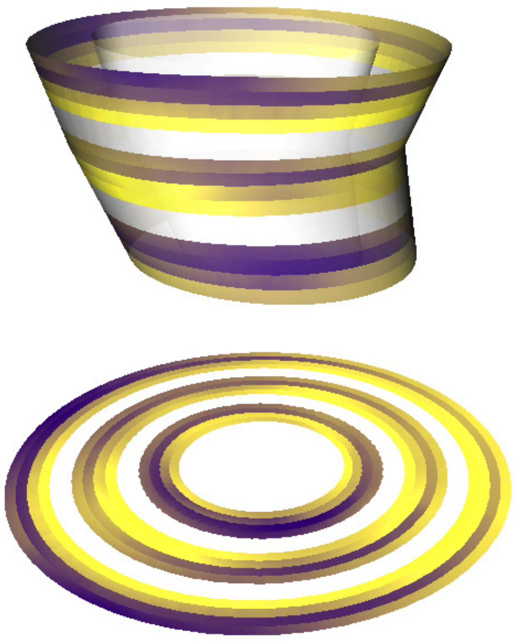


Figure 4.8. Visualization of wall thickness over multiple stress levels through the use of stress bands.

4.4. Visual Assessment of Parameter Data

The techniques presented so far allow for the analysis of LV parameters, which are represented continuously along a surface closely approximating the LV wall. A visual inspection of the distribution of color along the wall surface may point out problematic areas of the myocardium, which exhibit abnormal behavior. Such abnormalities are usually indicated by sudden color transitions, which signify a discontinuity in myocardium functionality. For example, in Figure 4.4(c), a red area can be seen, which stands out from the rest of the surface. The significance of the color imbalance in this case is that, in the red area, the myocardium is asynchronous with the rest of the wall, since it attains its maximum thickness much too early. Such dyssynchronies in wall behavior are significant causes of heart failure (Kass, 2002).

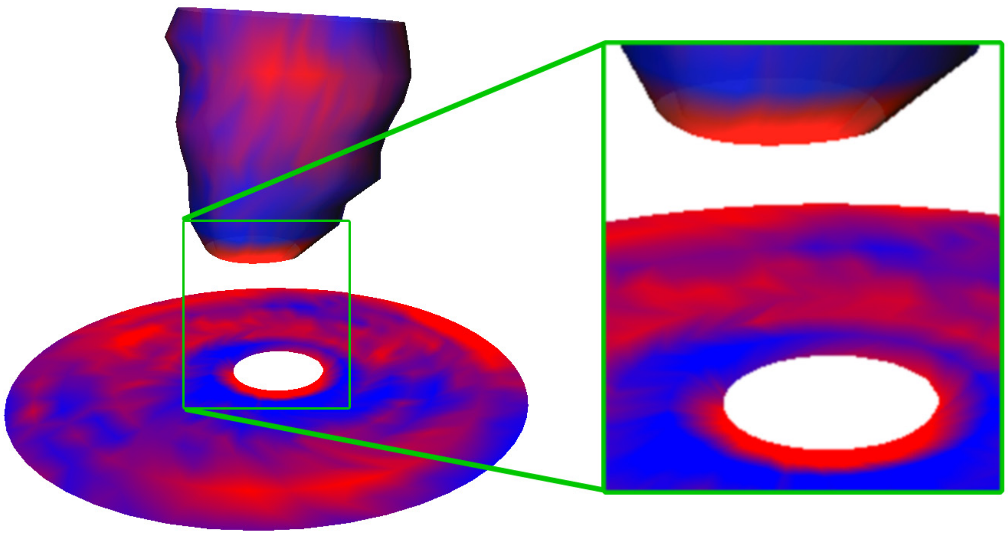


Figure 4.9. Epicardium surface and associated bull's eye plot depicting abnormal behavior in terms of wall speed.

Another example can be seen in Figure 4.9, which shows a representation of wall speed where a problematic area has been zoomed-in. A red ring is noticeable around the region corresponding to the bottom-most slice of the data set. This signifies that the wall around this slice moves at a significantly faster rate than its immediate neighborhood. Such a sudden leap in speed typically results from a shaking or flickering motion caused by arrhythmia (Wang, 2008) . Such a deviation from the normal cardiac cycle may well indicate cardiac conditions such as coronary artery disease, in which case areas of the myocardium exhibit fatigue or abnormal motion (Lipton et al., 2002; Khan, 2005).

Conclusions

The evaluation of myocardial parameters is important for the proper diagnosis of cardiac conditions. For this purpose, we proposed techniques for the visualization of 4D and 5D segmented MRI cardiac data. We computed several parameters which characterize the myocardial wall in the left ventricle, and subsequently represented their values on specially-constructed 3D geometry. Two 3D surfaces were generated from segmented contours, which approximated the shape of the epicardium and endocardium. Color ranges were then used to encode the parameters on the outer surface. A bull's eye plot was added to the geometry in order to display parameter information more comprehensively. We used a lens tool to simultaneously show two parameters on the same surface, and stress bands to plot parameter information across multiple stress levels. The distribution of color along the geometry was an indicator of potentially abnormal myocardial behavior. We provided examples of situations which reflected this. These methods of visually depicting the behavior of the myocardium make it possible to detect problems such as dyssynchronies or muscle fatigue, which may point to more serious cardiac conditions.

The techniques shown in this section are aimed at medical staff, who may benefit from a unified representation of cardiac parameters, as well as patients and, generally, users with little or no medical training, who may find these methods of displaying cardiac data easier and more intuitive to understand than using sets of individual grayscale slices. These techniques are therefore meant to aid in the diagnosis of heart conditions, medical training and patient information.

5. Parameter Assessment in Volume Visualization

The intricacy of volume data means that the methods, interfaces and applications involved in volume visualization have to be sophisticated enough so as to provide the quality, flexibility and efficiency which is often required when working with volume data sets. As the complexity and number of features of a volume rendering application increases, so does the number of parameters involved in the volume visualization process. In previous chapters, virtually every presented technique involved at least some minor parameter adjustment. From simple parameters such as the sampling rate, to more complex ones such as multidimensional transfer functions, parameter tweaking is a constant issue if any control over the output image is to be exerted. Generally speaking, since the meaningful information in volumes is often a subjective matter, many of the parameters involved in the visualization process are adjusted manually, allowing users to configure the rendered image to their liking. However, with so many parameters present in rendering algorithms, manual tweaking can become tedious. This is especially true since most interface elements used for parameter adjustment do not offer any a-priori information on the effect that parameters have on the resulting images. For most interface elements, this is difficult to anticipate in absence of any visual indicators.

Considering the above, the purpose of this chapter is to present newly-developed techniques for the analysis, evaluation and visualization of the characteristics and influence of parameters involved in volume rendering. The main purpose of this endeavor is the development of methods through which the effect of a parameter on the rendered image can be shown directly on the interface element which is used for adjustment. We present in each corresponding subsection techniques for the on-screen display of parameter behavior. This behavior is then represented through customized versions of interface elements commonly encountered in volume rendering applications, such as sliders or transfer function control widgets. Throughout this section, we present images of these interface elements and provide appropriate explanations of their usefulness and of the underlying algorithms.

5.1. Parameter Analysis Method

The methods presented throughout this section require a beforehand analysis on the effect of parameters, across part of, or their entire domain. By *effect* we mean the extent to which the rendered image changes on screen, if the value of a related parameter is modified. Such an analysis would provide at least some indication of how the image would change, before the parameter is actually adjusted.

The basic idea behind our analysis method is that the behavior of a parameter can be determined by comparing images rendered using values from part of, or its entire domain. As an example, let p be a parameter which accepts integer values from the set $\{0,...,9\}$. This parameter controls some aspect of the rendered image, in other words, its value is used at some point in the rendering pipeline. By generating images for each of the values of the parameter, 0 to 9, a comparison of the differences between these images would completely characterize the parameter and clearly map out any effect that it may have on the rendered image. With this information available, the adjustment of the parameter is significantly simplified, because it is now known what effect it would have on the output. Unfortunately, in real world volume rendering applications it is not possible to completely know the effect of a parameter in advance, since the parameters involved in volume visualization usually operate in a continuous domain. The solution in this case is to sample parameter values from its domain and compute the characteristics based on images obtained from the samples. The higher the sampling rate, the more accurate is the assessment, but the more computational resources are required.

The parameter analysis method comprises four main components:

- **the sampler** traverses the domain of the parameter and collects values at discrete points. The complexity of sampling rests mainly on the size and the number of dimensions of the domain in question. Often, the domain is so large, that the analysis has to be restricted to one or several narrower sub-domains.

- **the renderer** produces an image for each sampled parameter value. Rendering is done using the algorithms which exploit the parameter under scrutiny. The generated images are not displayed, but instead stored in memory for further processing.

- **the comparator** uses an appropriately-chosen metric to assess differences among the images obtained from the renderer. It usually generates an array of scalar values with the same number of dimensions as the parameter domain.

- **the interface** is the medium in which the scalar values generated by the comparator are represented, in order to illustrate parameter behavior. Such a medium is usually the interface element which is used to specify the parameters. This element is modified in some manner and new features are added to it which help with parameter assessment. One or several information visualization techniques are used to represent the scalar values associated with parameter behavior. These techniques may involve graphs, color coding, or some other related means. While the interface element is most commonly changed in terms of aspect only, its functionality may also be adapted to the newly-acquired knowledge regarding parameter behavior. For example, a non-uniform scaling of the parameter domain may improve the capabilities of the interface element and allow for more optimal parameter adjustment.

5.2. Sampling from the Parameter Domain

The sampler component of the assessment method has to traverse the domain partially or completely, and collect samples at discrete positions of the domain. We only consider uniform sampling for now, as other non-uniform or adaptive alternatives are more involved and the subject of future work. Depending on the nature of the parameter and on the type of interface used for its specification, we divide the parameters into the following categories:

- **line-based parameters**: these operate on one dimensional domains and are commonly adjusted via a slider widget. Sampling involves dividing the slider into several discrete positions using a sampling step (Figure 5.1(a)).

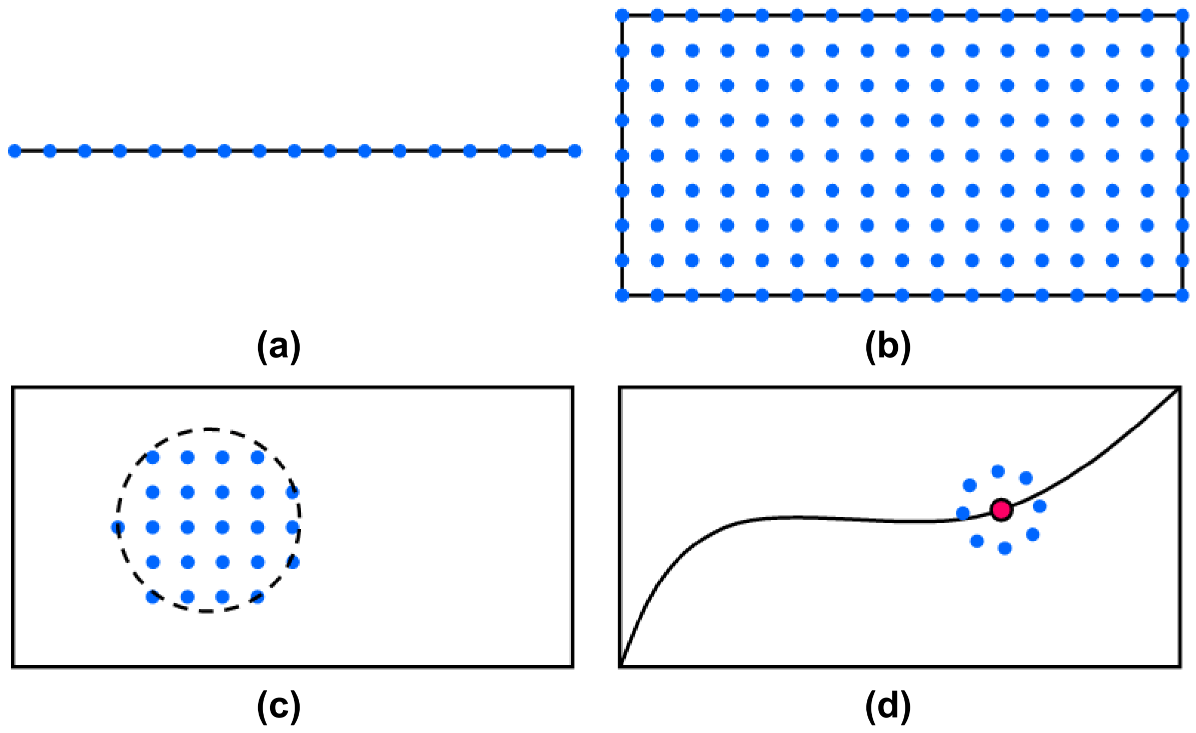


Figure 5.1. Sampling positions based on the type of parameter and associated interface element: (a) along the direction of a line-type widget; (b) on a regular grid; (c) on a sub-region within the operating area; (d) around a point on a spline.

- **area-based parameters:** these are adjusted by a two-dimensional widget. Sampling is carried out on a regular grid which subdivides the operating area (Figure 5.1(b)). The sampling positions may be restricted to a sub-region of the entire operating space (Figure 5.1(c)), if only a fraction of the parameter domain needs to be analyzed. If the sub-region is circular, radial sampling may also be used, though in this case the samples would be taken from unevenly-distributed positions. Some parameters may be associated with a position on a spline, as is the case for transfer function editors (Figure 3.3). Such a parameter may be mapped to a control point which, when moved, changes the shape of the spline. These parameters are treated distinctly due to the fact that certain sampling methods depend on the shape of the spline or on the position in relation to it. In the case of radial sampling, the samples are taken at a certain distance from a target point on the spline (Figure 5.1(d)). The sampling directions may be set as a function of the local spline tangent.

5.3. Rendering of Sample Images

In order to analyze the behavior and effect of the parameter, images need to be generated based on the sample values. For each sample, the parameter is first set to its corresponding value. The renderer then generates an image according to that sample. The image therefore reflects the specifics of the parameter in the position from its domain associated with that value. The images generated by the renderer may contain small differences among them which account for the changing value of the parameter. Figure 5.2. illustrates the image generation process for a line-type parameter. The same basic principle applies for all sampling methods from Figure 5.1.

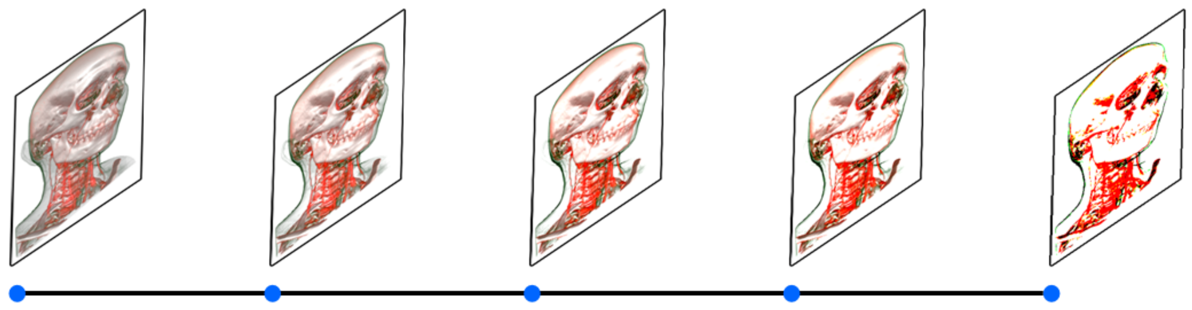


Figure 5.2. Generation of rendered images for samples taken from a line-based parameter.

Although Figure 5.2 may suggest that the images are generated and statically stored for all samples, in practice this is not the case. For illustrative reasons, only 5 samples are shown in Figure 5.2. In practice however, the actual number is in the hundreds for line-type parameters, and may even reach values in the tens of thousands for area-type parameters. Storing such a large number of images is therefore unrealistic. In the actual application, the images are generated using a buffer-type approach, in sets of several at a time. Once the current set has been processed, it is discarded and the next one is generated. Also, since the number of required images is quite high, the rendering process has a significant impact on performance. For example, one extra millisecond in rendering time for a single image might mean several minutes of waiting for an area-type parameter to finish processing.

5.4. Image Difference Computation

In order to analyze the changes induced by the variation of the parameter, an image metric is required to compare and compute the difference between rendered images. The concept is illustrated in Figure 5.3, where a differencing metric determines the distinctive elements between two images. For demonstrative purposes, basic pixel subtraction is shown.

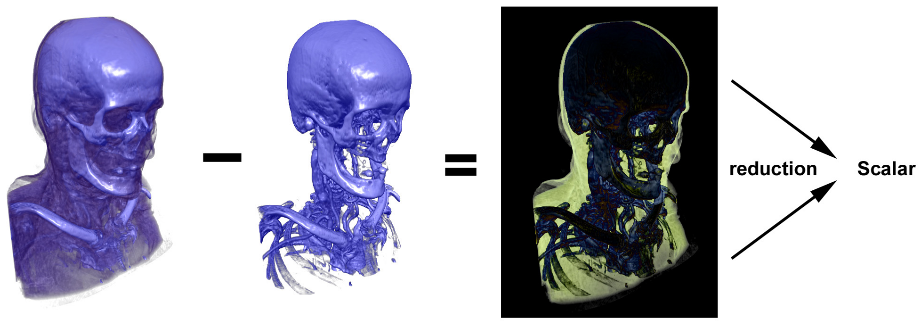


Figure 5.3. Image difference computation through basic pixel subtraction. The resulting image is reduced to a scalar value which indicates the amount of change between the two original images.

For the purposes of our method however, we have examined other, more complex metrics and developed our own custom one, which we further use for the assessment of parameters. Image comparison metrics generally work on a pixel-by-pixel basis. A differencing algorithm is used to compute the discrepancy between the corresponding color values of two images. Once a difference image has been obtained, it is used as is, or reduced to a scalar value, for example through pixel averaging. If a set of images corresponding to parameter samples are used, the differences between each pair of images or between each image and another reference image produces a set of scalars which quantify the amount of change induced by parameter variations. The most basic of such metrics are pixel subtraction (Figure 5.3) and the RGB distance, which applies the formula for Euclidian distance to R, G, and B values. Wilson et al. (1997) present other common metrics, among which are RMS (the Root Mean Squared Error) or SNR (Signal/Noise Ratio). They also develop their own metric, Δ_g, which draws upon the Sobolev norm and Hausdorff metric. Chan et al. (2004), introduce a method based on a Canny edge detector. Another class of metrics attempts to correlate the computed image differences with corresponding changes, as interpreted by the human vision model. We refer to these as *perceptually accurate*, since they mimic the human perception of brightness and color. The advantage of using such metrics is that the computed differences among image sets match what a human observer would perceive. This is especially useful when using an information visualization method to illustrate these differences. For example, when mapping the differences to color values, a human observer would expect that the changes in the rendered images visually correlate with the variations in the color spectrum used for coding the computed differences. The major downside of these methods is their significant negative impact on performance. The required computations are typically very taxing, making such metrics unsuitable for real-time applications. The performance is also highly affected by the resolution of the rendered image. Examples of such metrics are the ones developed by Titov (2000), Farrugia et al. (2004) and Pedersen and Hardeberg (2009). Their methods account for properties such as color distance in non-linear color spaces, the estimated viewer distance, or relative pixel position.

Considering related research and results found throughout the literature, we have developed our own metric, which attempts to mimic perceptional cues from human vision. The metric is built using a "bottom-top" approach, and comprises the following steps, considering that it takes two images as input and returns a scalar difference:

- background pixels which are common for both images are discarded, since they do not yield useful information.

- a slight low-pass filter is applied to the images. This removes noise and smoothes out the images without inducing a relevant loss of detail.

- the images are converted to CIE-Lab, using the standard D65 white point. This color space is perceptually uniform and approximates the manner in which human vision interprets color.

- a difference image is obtained by computing the color distance ΔE^{*}_{ab} between the two processed images (Ohno, 2000).

- a scalar is computed by reducing the difference image (via pixel averaging).

While the aforementioned metric is rather demanding in terms of the required computational power, we have found it to be a good compromise between performance and visual relevance. Unlike perceptually inaccurate methods, our approach generates scalar differences which correlate with the corresponding changes in the rendered images, while having a significantly lower impact on performance than other perceptually accurate variants.

5.5. Visualization of Parameter Information

Volume rendering applications contain multiple interface elements, which allow the manipulation of rendered images by modifying the values of their associated parameters. These controls vary in complexity, from simple buttons and sliders, to more complex transfer function specification widgets (Figure 3.3). While many efforts have been made to automate the parameter adjustment process (Sereda et al, 2006; Kohlmann et al., 2007; Zhou and Takasuka, 2009), virtually all volume rendering application allow an extensive amount of manual control of the parameters involved in the rendered output. Manual control is in many cases preferred, because it allows greater flexibility. An image containing meaningful information is therefore the result of parameter adjustment, which is carried out by means of various interface elements, such as buttons, sliders, graphs, area controls, and many other widgets. The disadvantage when presenting users with an extensive amount of manual control is that, unless they already know what to look for in the data set, the parameter adjustment process may become tedious and subject to a lot of trial-and-error. Part of the problem is that few if any interface elements provide feedback regarding the influence of potential parameter changes on the resulting images. We offer a solution to this problem: the incorporation of parameter information, as deduced in previous sections, into the layout of its associated interface element. This provides a-priori indications regarding the impact of the position, orientation and movement of interface controls on the on-screen image. Additionally, this allows the mapping of parameter stability by means of various visual indicators, thereby providing more in-depth information on parameter behavior.

Once the information on parameter-induced changes has been obtained via the methods described in previous sections, it is further used to modify and enhance the interface elements which control the parameter. Normally, an interface element uses a standard type of interpolation to specify parameter values in-between key positions of its controls. However, once the changes induced by the parameter are known in advance, the interface element can be non-uniformly scaled according to the parameter information. This results in interfaces which allow more sensitivity when fine-tuning parameters. The data regarding parameter-induced changes is shown using information visualization techniques such as graphs, arrows, or color mapping. In the following sections, these concepts are implemented for two types of interface elements frequently-encountered in volume rendering, in particular, and in software applications, in the more general case. The considered interface elements are slider-type widgets and 1D transfer function specification controls (Figure 3.3). These widgets are modified in such a way so as to readily display the impact of the changes incurred upon manipulating cursors and control points to modify parameter values.

5.5.1. Visualization of Parameters on Graph Sliders

The information derived from the parameter assessment methods should be represented in a comprehensive, intuitive manner. As mentioned, the proposed method involves the incorporation of parameter information in the design of common interface elements. Sliders are a frequently-used widget for the adjustment of one-dimensional parameters.

A regular slider (Figure 5.4(a)) allows the modification of a parameter whose domain is a continuous interval. In some cases, the values at the limits and at the current positions may be known, at most. In the case of Figure 5.4(a), the parameter associated with the slider is the step size used for sampling during ray casting (Figure 2.11(b)). A value of 1.0 means that one sample per voxel is taken along the corresponding ray. However, this slider provides no information on the changes that occur as a result of its manipulation. In some cases, the user may have some vague intuitive knowledge of how changing the parameter will affect the resulting image (for example: increasing the step size will eventually result in visible sampling artifacts in the image). Mostly however, this is a result of suppositions and guesswork. At this point, the only way to truly acquire an understanding of the influence of this parameter is to actually move the slider and visually evaluate the on-screen changes. In other words, no beforehand feedback is available regarding the influence of the slider on the output image.

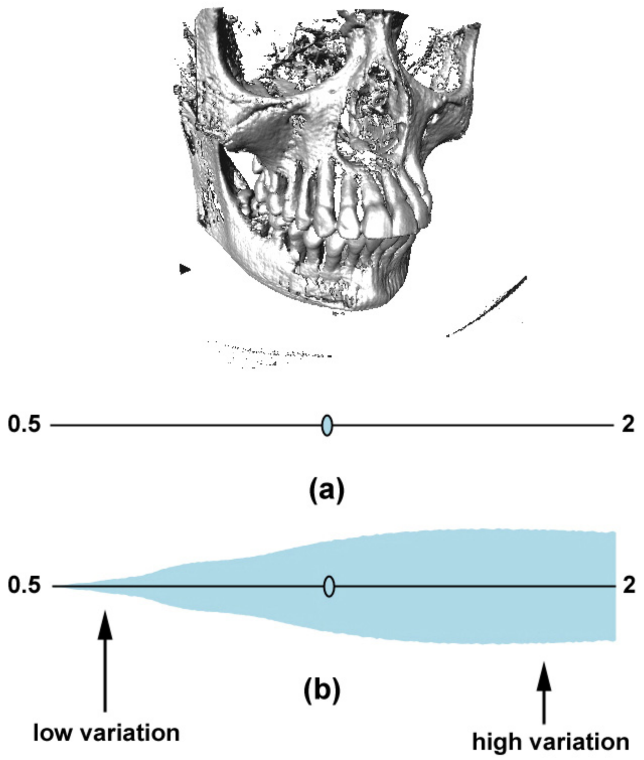


Figure 5.4. (a) a regular slider with no additional information**;** (b) a graph slider showing the magnitude of change induced in the rendered image when repositioning the cursor (Gavrilescu et al., 2010(a)).

The slider in Figure 5.4(b) has been augmented with a graph which circumvents the previously-described problem. We refer to this as *graph slider*. For every position along the slider, the graph shows the magnitude of change which would occur in the rendered image if the cursor of the slider were moved from that position to a neighboring one. The user can now make more informed decisions in terms of parameter adjustment. For instance, regions where the parameter has little to no effect could be ignored in favor of ones which exhibit greater change. Moving the slider in regions with low variation has minimal impact on the output.

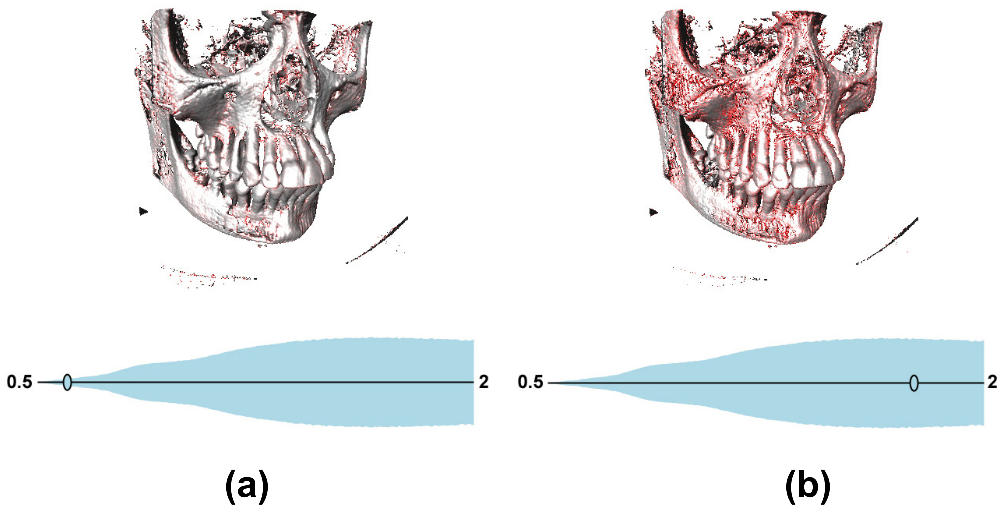


Figure 5.5. The amount of change in the rendered image: (a) when the slider cursor is slightly moved in low variation areas, little change occurs; (b) in higher variation areas the changes are a lot more numerous (Gavrilescu et al., 2010(a)).

The graph is generated by sampling parameter positions as shown in Figure 5.1(a). Images are then rendered for each position (Figure 5.2), and differences are computed between each pair of consecutive images. The result is an array of scalar values which is then used to plot the graph. The thickness of the graph translates directly to the corresponding scalar value, therefore it indicates the amount of change in the rendered image. Figure 5.5 shows the image-space changes which occur for two different positions on the same slider, when rendering the same dataset. The changes are highlighted in red, and indicate the variations which occur in the rendered image if the slider cursor is moved in the immediate neighborhoods of the positions in Figure 5.5. The changes seen in Figure 5.5(a) are minimal and correspond to positions of the slider cursor in low-variation areas, as depicted by the graph. With the slider in positions of greater variation (Figure 5.5(b)), the changes in the rendered image are a lot more numerous, and this is reflected by the thickness of the graph in the corresponding area. The graph thickness correlates with the amount of change seen in the image.

5.5.2. Graph Slider Scaling

The newly-displayed information on the graph slider allows for further improvements to the adjustment process. During parameter tweaking, a user may notice sub-regions of the slider which are of particular interest (for example, those corresponding to a certain amount of change, or where there is a sudden leap in the graph). While displaying the change-related information allows the identification of these sub-regions, the fine-tuning of parameter values is still difficult for narrow ones. We solve this problem by non-uniformly rescaling the slider, to allow for more cursor movement in the regions of interest. Rescaling is carried out in two ways: manual and automatic.

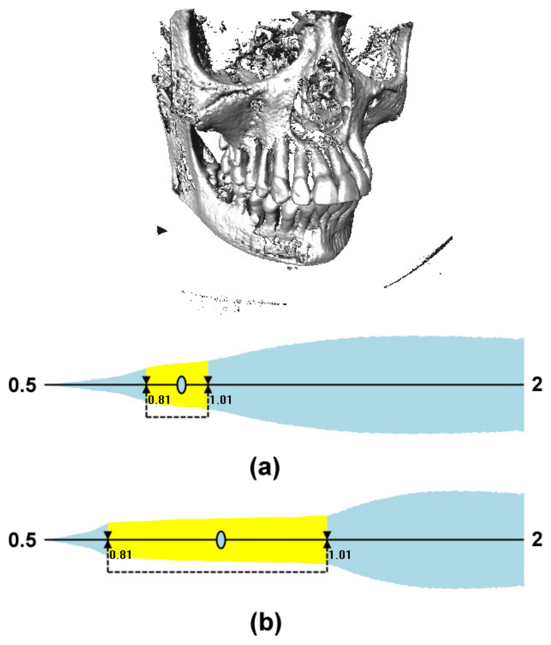


Figure 5.6. Graph slider which allows manual non-uniform rescaling: (a) a region is selected, and (b) is stretched so as to allow for better precision when fine-tuning (Gavrilescu et al., 2010(a)).

Manual rescaling involves further customization of the graph slider interface, in order to allow users to select their preferred region and scale it accordingly. The implementation of this concept is shown in Figure 5.6.

In Figure 5.6(a), a sub-region of interest is selected using two clamps. When the clamps are pulled apart (Figure 5.6(b)), the sub-region expands accordingly, filling a wider segment of the slider. It can be stretched as far as the entire slider length, and the clamps keep their values throughout the entire scaling process. Once the selected sub-region has been scaled, a lot more area is available, thus the cursor can be placed more precisely inside the sub-region of interest.

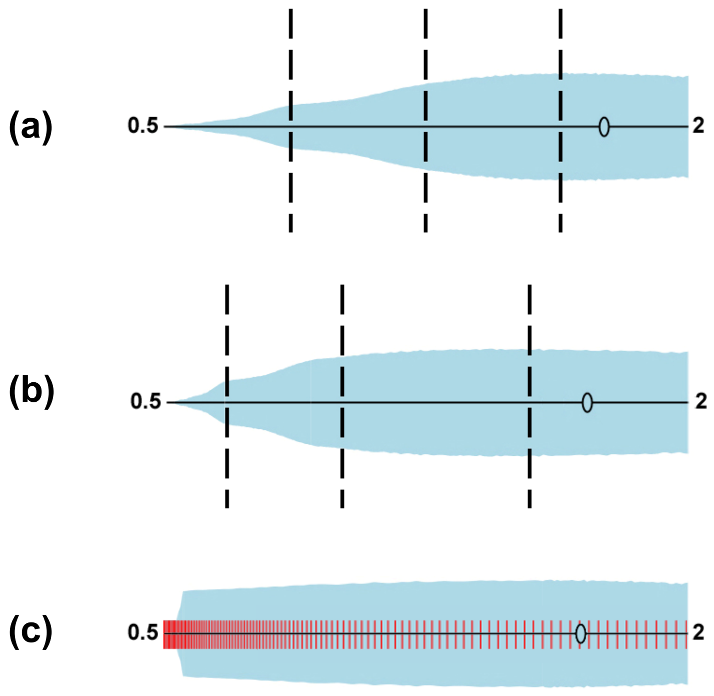


Figure 5.7. Automatic slider scaling: (a) the slider is partitioned into sub-regions; (b) the sub-regions are scaled according to the amount of change occurring between their bounds; (c) fully rescaled slider where the on-screen changes are proportional to the movement of the cursor. The red markers show the non-uniform scaling.

The automatic rescaling of the slider involves partitioning it into multiple sub-regions (Figure 5.7(a)), then resizing them according to how much potential change occurs within their boundaries (Figure 5.7(b)). In other words, upon rescaling, the sub-regions where the graph is thick will occupy a larger portion of the total slider area than sub-regions where the graph is narrower. Therefore, the slider is rescaled according to how much change it causes in the output. If the slider is partitioned sufficiently densely, upon rescaling the result is that the change perceived in the output image is proportional to the amount that the slider cursor is moved (Figure 5.7(c)). The amount of change induced by small cursor movements is nearly the same throughout the whole length of the slider. We refer to this as *perceptually uniform* behavior. The changes perceived in the output image match the gestures of the user when manipulating the slider. The advantage of this method is that, on the one hand, the need for fine-tuning is substantially reduced, since previously narrow sub-regions which generated a significant amount of change are now scaled to much larger areas on the slider. The movement of the cursor no longer reflects parameter values directly, but rather the variations induced in the output image. Another advantage is that "dead" regions on the slider, i.e. regions in which moving the cursor has little to no effect, are virtually eliminated. Thus, the manipulation of the slider is optimized to allow for as little tweaking as possible.

5.5.3. Augmentation of Transfer Function Interfaces

As seen in Section 3, transfer functions are an important parameter in volume rendering. The manual adjustment of transfer functions involves various interface elements such as the one shown in Figure 3.3. This widget was developed as part of UniVolume, but it is similar in design to those found in other volume rendering applications, where the manual adjustment of transfer functions is required. However, such editors suffer from drawbacks similar to the previously-mentioned sliders: they offer no beforehand information on the changes induced in the rendered image. Unless the structures and distribution of opacity within the volume are well known to the user, the adjustment of the transfer function shape is a trial-and-error process. Searching for the best, most revealing transfer function may therefore prove to be a time-consuming task. However, by using the previously-described parameter assessment techniques, new information can be added to the associated widget, so as to streamline the transfer function specification process.

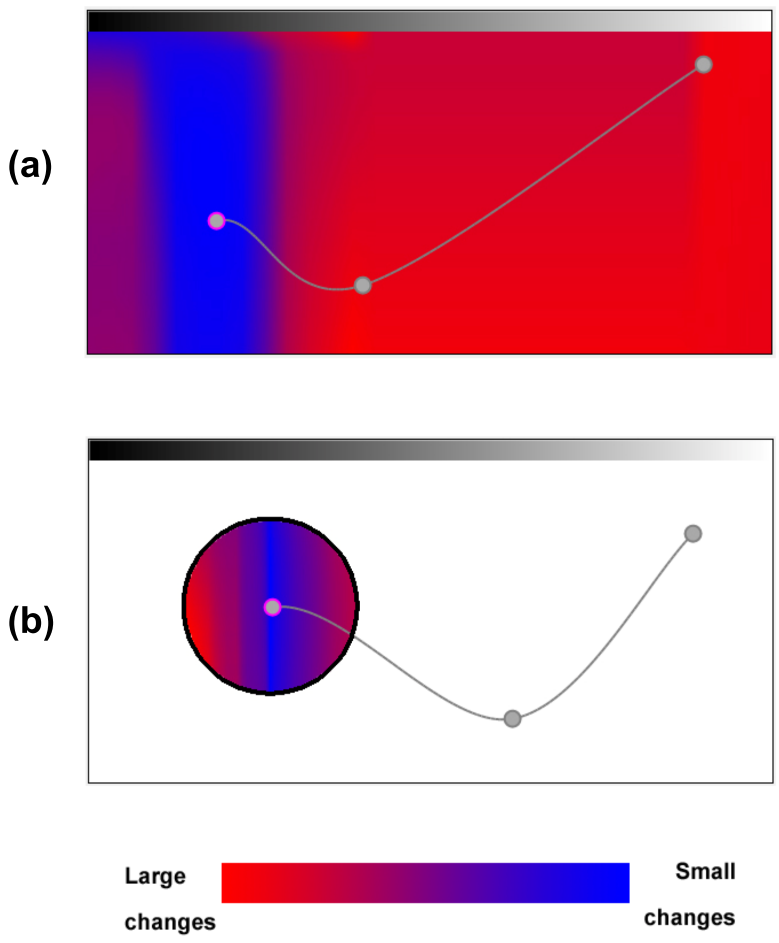


Figure 5.8. Enhanced transfer function specification interfaces where changes induced by the left-most control point are color coded onto the interface area. (a) the change magnitudes are shown for all positions of the control point; (b) the change magnitudes are shown in a sub-region around the control point.

The newly added information illustrates the magnitude of changes induced in the rendered image by changing the position and shape of the transfer function spline. We convey this information by means of various visualization techniques. The goals are to help with the transfer function specification process, to isolate useful transfer function shapes and to convey information regarding transfer function stability. We mainly focus our analysis on regions around the control points which are used to shape the transfer function spline, and illustrate the changes that occur in the output image for regions and sub-regions around these control points.

In Figure 5.8(a), the potential positions of the control point to the left are sampled as shown in Figure 5.1(b). Images are then rendered for the current position of the control point, and for each sampled position. The images corresponding to the sampled positions are then compared to the image generated with the control point at its present location. This results in an array of scalar values which show the magnitude of change that would occur in the rendered image if the control point were moved from its current position to the position of each sampled point. These values are represented on the interface via color coding. Bilinear interpolation is used to generate a continuous image. The result is a color map which indicates the behavior of the control point across the entire area of the interface. We refer to this as the *stability map* of the transfer function with regard to the highlighted control point. The more blue a sub-region, the less of a change is determined by the transfer function in that region (thus, the transfer function is stable), while more red regions point to significant changes (which denotes instability). The apparent vertical orientation of the map is due to the fact that changing the position of the control point horizontally has a more significant impact on the output than if the control point were moved vertically (changes in density affect the rendered image more than changes in opacity).

Alternatively, the sampled area can be restricted to a sub-region around the control point (Figures 5.1(c) and 5.8(b)). A narrower sampling area means that the information in the neighborhood of the control point can be represented with more precision. This is useful when fine tuning the transfer function, that is, moving the control point slightly, as opposed to along a larger area of the interface.

Stability maps suffer from the drawback that they require a substantial amount of sampling and rendering (Figures 5.1(b) and (c)), which translates to high computational requirements. For this reason, we have developed other, less taxing ways of showing the impact of transfer functions. The focus is sampling on narrow regions around the control points (Figure 5.9).

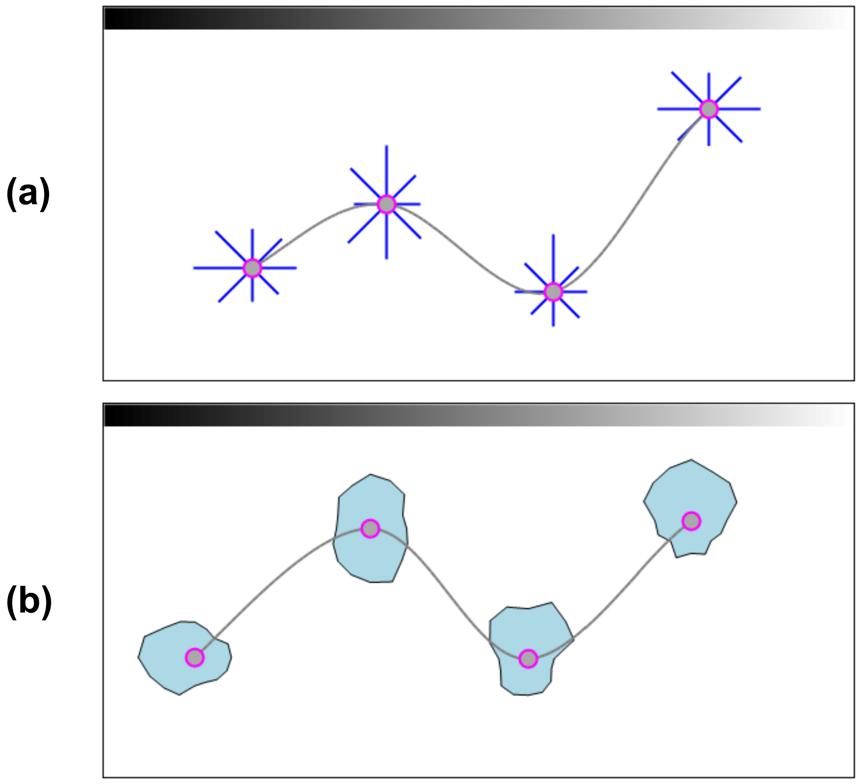


Figure 5.9. Transfer function interfaces which show the impact of a control point using (a) arrows and (b) deformable shapes

In the transfer function interface from Figure 5.9(a), the arrows protruding from the control points show the amount of change in the output image caused by moving the control points in the their respective directions. The transfer function is sampled as shown in Figure 5.1(d), and images are rendered for each sample position, and for the current position of the control points. The changes are computed by differencing the images generated from each control point and from each of its sampled positions. The resulting values are then used to set the length of the arrows accordingly. The advantage of this approach is that a small number of samples is sufficient to provide good feedback on the influence of the control point on the image. However, if multiple samples should be used, drawing arrows for each of them would clutter the area around the control point, and would thereby make the data difficult to interpret. A viable solution to this problem is to draw a polygon based on the tips of the arrows, instead of arrow lines (Figure 5.9(b)). The result is a deformable "blob"-type shape which bulges in the direction of greater change magnitude. Depending on the number of samples taken, this may be a more effective means of showing the impact of the control points, since the shape of an object is generally easier to visually interpret than multiple radially arranged lines. The data sets used to compute the information from Figures 5.8 and 5.9 are the same ones rendered in images from Chapters 2 and 3, for example in Figures 2.11, 3.1 or 3.3.

Conclusions

An assessment of the influence of various parameters on the images generated via volume rendering is beneficial to understanding the behavior of these parameters, and how they may be better adjusted in order to generate the desired results. To that end, we developed a method for assessing the influence of parameters on the changes which occur in rendered images. The approach was based on sampling parameter values from their domains, rendering images for each sample and then comparing these images in order to assess parameter-induced differences. These differences corresponded to the magnitude of change that the parameters induced in the rendered output. The differences were computed using our own perceptually-accurate metric, which ensured that they matched the visually-perceived changes in the rendered image.

The array of values resulting from the parameter assessment process was then used to enhance two types of interface elements commonly encountered in volume rendering: sliders and transfer function specification widgets. Sliders were enhanced by representing the difference values on a graph, which showed the magnitude of change induced by cursor movement. This meant that more informed decisions could be made in terms of parameter adjustment. The slider could also be scaled manually, which improved the precision when fine-tuning parameter values. Automatic scaling re-arranged sub-regions on the slider area so that the movement of the cursor would match the changes of the on-screen image.

Finally, we applied the same principle to transfer function widgets. By sampling control point positions and assessing the differences, we generated the stability maps of the control points. These specify the influence of the control point on the output image for either the entire area of the widget, or a narrower sub-region in the neighborhood of the control point. We then proposed additional methods for the visualization of parameter-induced changes. By appropriately sampling the control point and then drawing arrows which extend from it, we were able to show in which directions moving the control point would cause more change. The extension of this concept are deformable shapes, which extend in the directions of greater change and are more easily interpretable.

The methods developed in this section allow for the better understanding of parameter behavior and the customization of interface elements, so that they may convey beforehand user feedback or provide better means of adjusting parameters.

6. Final Conclusions

The thesis has thus far addressed multiple topics from the field of volume visualization. The concepts and results presented throughout the sections are the product of extensive research and documentation. This effort has resulted in the accumulation of expertise in volume graphics, which has allowed the development and implementation of techniques for the visualization, classification and analysis of volume data sets. The aims of this work were the gain of knowledge on fundamental notions, approaches and methods in volume visualization, as well as the development of new techniques in order to contribute to the progress of research in the field. The concepts and methods explained and developed throughout the thesis range from basic volume rendering, classification, the visualization of various data sets using multiple approaches, to the analysis of volume rendering parameters. We present, improve upon, and develop multiple techniques which allow the processing of volume data in multiple ways.

At the beginning of this work, in Chapters 1 and 2 we presented our objectives and thesis outline, as well as fundamental notions and methods used in state-of-the-art volume rendering. We introduced UniVolume, a prototyping framework where we implemented multiple algorithms for volume rendering, visualization, classification and analysis. Volume visualization techniques operate on data sets, which most commonly originate from various scanning devices such as CT or MRI. The data sets are subjected to reconstruction and filtering mechanisms. These generate a continuous volume which may be arbitrarily sampled. The visualization of volumes is carried out according to an optical model, where the most frequently-used one is emission-absorption. The rendering of volumes occurs according to the volume rendering integral, an equation which describes the accumulation of emission and attenuation effects on light rays during their propagation through multiple media. The representation of volume data presupposes the numerical computation of this integral. We therefore presented a few rendering methods used by state-of-the-art volume rendering applications. These are divided into object-ordered and image-ordered techniques. The most prevalent of these methods is the ray casting algorithm, which involves a direct evaluation of the volume rendering integral. The functionality of ray casting was elaborated upon and illustrated using images generated by our own ray caster.

In Chapter 3 we addressed two important topics involved in volume visualization: classification and the highlighting of features from volume data. We explained the concept of transfer functions and how they can be used to bring out features of interest from volume data, while discarding unimportant information. Transfer functions may operate on multiple voxel properties where they achieve various types of classification. Data-based transfer functions allowed the classification of volumes based on the scalar values within the data set. The gradient magnitude was used to better emphasize surfaces and to classify structures of similar density. Local curvature emphasized surfaces with valley- and ridge-type features. Distance-based criteria allowed the slicing and isolation of volume features based on the spatial position of voxels relative to a focal geometry. Visibility was used to highlight less apparent features in volumes. We proposed efficient ways to approximate parameters such as curvature and visibility and incorporated multiple classification criteria by means of component-based multidimensional transfer functions. We then used various instances of these functions to classify CT and MRI data, where we successfully removed obscuring features to reveal certain underlying tissues. Volume classification is one of the core features of our prototyping environment, and the implemented and developed classification techniques are being upgraded and improved continually.

The second topic of this chapter involves the development of techniques for the enhancement of features and details from volume data. The dot product between the gradient and viewing vector was used as a criterion to render outlines on the edges of shapes and surfaces. A method based on curvature control was employed to regulate the size of the outlines. Together with that, we also presented the possibility to smooth out the outlines, which better highlighted the finer features in surfaces from the volume. Subsequently, we proposed an image processing technique which was applied in the post-processing stage of the volume rendering pipeline. The technique was based on a combination of high-pass filtering and custom blending, and it was used to highlight edges and to provide sharper and clearer rendered images. The proposed methods work alongside classification to highlight and enhance the details of surfaces and structures within the volume.

Chapter 4 dealt with the computation and representation of various properties derived from cardiac MRI. Two sets of contours, which outline the epicardium and endocardium, were pre-segmented from a set of MRI slices representing the left ventricle. We used the segmented contours to construct surfaces which approximated the shape of the myocardial wall. The surface was then used to color-code various parameters which characterized the status and functionality of the left ventricle. The parameters in question were: the thickness of the wall, the degree of thickening, the moment of maximum thickness, and the wall speed. The colors indicated the distribution of parameter values along the left-ventricular wall, and could therefore be used to evaluate whether certain parameter values in a specific region of the wall constituted an indication of a cardiac condition. The parameters could be represented individually, or concurrently, using a lens tool, while various color interpolation methods could be used to emphasize regions of color transition.

We also developed a method for the representation of parameters from five-dimensional data, which included several stress levels. For this purpose, we used a technique which we referred to as "stress bands" to simultaneously display parameter values for all stress levels concurrently, on the same geometry. For comparison, we also displayed the data on bull's eye plots, a common means of representing cardiac data.

The methods developed for displaying the characteristics of the myocardial wall are meant to contribute to cardiac diagnosis and treatment. They may also serve as educational tools, or as intuitive ways of relaying information to patients who would not otherwise know how to interpret the grayscale slices used traditionally by medical staff.

In Chapter 5 we introduced a method for the assessment of parameter behavior in volume rendering applications. It involved sampling the parameter over part of or all its domain, then rendering a set of images using the sampled parameter values. Computing the differences among these images would then characterize the behavior of the parameter with regard to the changes induced in the output image. We concluded that, in order to compute image differences, a perceptually accurate image comparison metric was the most suitable choice. We therefore developed a custom metric, which encompassed several steps, including the conversion of the images to a perceptually-uniform color space. The difference values obtained for the parameter were then used to customize the interface elements used for its adjustment. Thus, we converted regular sliders into custom *graph sliders*, which, alongside the regular horizontal line and cursor, displayed the magnitude of change which would occur in the image for every cursor position. Furthermore, the graph slider could be non-uniformly scaled, allowing for the perceptually accurate adjustment of its associated parameter.

The parameter data was further used to modify transfer function editor interfaces. Specifically, we were interested in determining the stability of the transfer function in one of its sub-domains. For this purpose, we used several sampling schemes to compute the stability of control points using arrows and shapes, or throughout the entire surface area of the widget. This resulted in transfer function stability maps, which displayed the degree of change induced by a transfer function control point using color coding.

The techniques studied and developed for the elaboration of this thesis have provided substantial insight into the field of volume visualization. We therefore intend to continue improving upon the work described thus far. There are numerous directions in which progress could be made: from the enhancement of rendering speed and quality, to the development of new techniques for the analysis of medical data, to the further augmentation of user interfaces based on a-priori knowledge of parameter influence. Further contributions in the field would help advance and improve upon state-of-the-art volume visualization techniques.

References

Antoch G., Jentzen W, Freundenberg L. S., Stattaus J., Mueller S. P., Debatin J., Bockisch A., Effect of oral contrast agents on computed tomography-based positron emission Tomography, Investigative Radiology, vol. 38, no. 12, pp. 784-789, 2003.

Bary L., MR-based quantitative analysis of the local myocardial contraction - to assist cardiac resynchronization therapy, MSc Thesis, University of Technology Eindhoven, 2007.

Blankenberg F. G., Molecular imaging with single photon emission computed tomography, IEEE Engineering in Medicine and Biology Magazine, vol. 23, no. 4, pp. 51-57, 2004.

Bluth E. I., Ultrasound: a practical approach to clinical problems, Thieme Medical Publishers, New York, 2008.

Brown M. A., MRI: basic principles and applications, Wiley-Blackwell, 2010.

Bruckner S., Gröller E., VolumeShop: an interactive system for direct volume illustration, in Proceedings of Visualization 2005, pp. 671-678, 2005.

Bruckner S., Gröller E, Style transfer functions for illustrative volume rendering, Computer Graphics Forum, vol.26, no. 3, pp. 715-724, 2007.

Burns M., Haidacher M., Wien W., Viola I., Gröller E., Feature emphasis and contextual cutaways for multimodal medical visualization, in Data Visualization - EuroVis 2007, pp. 275-282, 2007.

Bushong S., Computed tomography, McGraw-Hill, USA, 2000.

Callahan S. P., Bavoil L., Pascucci V., Silva C. T., Progressive volume rendering of large unstructured grids, IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 5, pp. 1307-1314, 2006.

Caravan P., Ellison J. J., McMurry T J., Lauffer R. B., Gadolinium(III) chelates as MRI contrast agents: structure, dynamics, and applications, Chemical Reviews, vol. 99, no. 9, pp. 2293-2352, 1999.

Carlborn I., Optimal filter design for volume reconstruction and visualization, in Proceedings of Visualization 1993, pp. 54-61, 1993.

Catmull E., Rom R., A class of local interpolating splines. In Computer Aided Geometric Design, Academic Press, New York, pp. 317–326, 1974.

Chan W. C., Le M. V., Le P. D., A wavelet and Canny based image comparison, in the Proceedings of the. 2004 IEEE Conference on Cybernetics and Intelligent Systems, Singapore, pp. 329-333, 2004.

Correa C. D., Ma K. -L., The occlusion spectrum for volume visualization and classification, IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 6, pp. 1465-1472, 2009.

Correa C. D., Ma K. -L., Visibility histograms and visibility-driven transfer functions, IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 2, pp. 192-204, 2011.

Engel K., Kraus M., Ertl T., High-quality pre-integrated volume rendering using hardware-accelerated pixel shading, in Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, pp. 9-16, 2001.

Farrugia J. P., Albin S., Peroche B., A perceptual adaptive metric for computer graphics, in Proceedings of WSGC2004, Plzen, Czech Republic, pp. 49-52.

Feng W., Nagaraj H., Gupta H., Lloyd S. G., Aban I., Perry G. J. et al., A dual propagation contours technique for semi-automated assessment of systolic and diastolic cardiac function by CMR, Journal of Cardiovascular Magnetic Resonance, vol. 11, no. 1, pp. 1-13, 2009.

Gavrilescu M., Manta V, Volume visualization applied in medical imaging, Bulletin of the Polytechnic Institute of Iasi, LIV (LVIII), pp. 43-52, 2008.

Gavrilescu M., Manta V., Purgathofer W., Post-rendering enhancement of volumes, Bulletin of the Polytechnic Institute of Iasi, LV(LIX), 3, pp. 43-54, 2009.

Gavrilescu M., Manta V., Computation and representation of left-ventricular wall parameters from multidimensional MR cardiac data, in Proceedings of Computer Graphics and Imaging 2010, Innsbruck, Austria, pp. 128-135, 2010.

Gavrilescu M., Malik M. M., Gröller E., Custom interface elements for improved parameter control in volume rendering, in Proceedings of the 14th International Conference on System Theory and Control, Sinaia, Romania, pp. 219-235, 2010(a).

Gavrilescu M., Malik M. M., Gröller E., Enhanced interfaces for parameter adjustment in volume rendering applications, Bulletin of the Polytechnic Institute of Iasi, LVI (LX), 4, pp. 163-174, 2010(b).

Gavrilescu M., Manta V., Advances in the visualization of three-dimensional seismic volume data, Environmental Engineering and Management Journal, vol. 10, no. 4, pp. 567-578, 2011.

Gavrilescu M., Manta V., Gröller E., Gradient-Based Classification and Representation of Features from Volume Data, *15th International Conference on System Theory and Control (accepted),* Sinaia, Romania, 2011.

Gibson D., Spann M., Turner J., Automatic fault detection for 3D seismic data, in Proceedings 7th Digital Image Computing: Techniques and Applications, pp. 821-830, 2003.

Gonzales R. C., Woods R. E., Digital image processing, second edition, New Jersey: Prentice Hall, 2008.

Hadwiger M., High-quality visualization and filtering of textures and segmented volume data on consumer graphics hardware, PhD Thesis, Vienna University of Technology, 2004.

Hadwiger M., Sigg C., Scharsach H., Bühler K., Gross M., Real-time ray-casting and advanced shading of discrete isosurfaces, Computer Graphics Forum, vol. 24, no. 3, pp. 303-312, 2005.

Hadwiger M., Kniss J., Rezk-Salama C., Weiskopf D., Engel K., Real-time volume graphics, AK Peters, Wellesley, MA, USA, 2006.

Hashemi R. H., Bradley W. G., Lisanti C.J., MRI: The basics, 2nd ed. Maryland: Lippincott Williams & Wilkins, 2004.

Hautvast G., Segmentation of short axis cardiac MR using active contours, Msc. Thesis, Technical University of Eindhoven, 2004.

Hautvast G., Lobregt S., Breeuwer M., Gerritsen F, Automatic contour propagation in cine cardiac magnetic resonance images, IEEE Transactions on Medical Imaging, vol. 25, no. 2, pp. 1472-1482, 2006.

Haykin S., van Veen, B., Signals and systems, John Wiley and Sons, 1998.

Hege H. C., Hollerer T., Stalling, D., Volume rendering - mathematical models and algorithmic aspects, Report TR 93-7, ZIB (Konrad-Zuse-Zentrum), Berlin, 1993.

Herghelegiu P., **Gavrilescu M.**, Manta V., Visualization of segmented structures in 3D multimodal medical data sets, *Advances in Electrical and Computer Engineering,* vol. 11, no. 3, pp. 99-104, 2011.

Hladuvka J., König A., Gröller E., Curvature-based transfer functions for direct volume rendering, in Proceedings of Spring Conference on Computer Graphics and its Applications (SCCG 2000), Budmerice, Slovakia, 58-65, 2000.

Hsieh J., Computed tomography: principles, design, artifacts and recent advances, International Society for Optical Engineering, Bellingham, WA, USA.

Iaizzo P. A, Handbook of cardiac anatomy, physiology and devices, Hew Jersey: Humana Press, 2005.

Jeong W. -K., Whitaker R., Dobin M., Interactive 3D seismic fault detection on the graphics hardware, in Proceedings International Workshop on Volume Graphics 2006, Boston, MA, pp. 111-119, 2006.

Jonsson M., Volume rendering, MSc Thesis, Umea University, Sweden, 2005.

Kass D. A., Ventricular dyssynchrony and mechanisms of resynchronization therapy, European Heart Journal Supplements, 4 (Supplement D), D23-D30, 2002.

Khan M. G., Heart disease diagnosis and therapy - A practical approach, second edition, New Jersey: Humana Press, 2005.

Kindlmann G., Durkin J. W., Semi-automatic generation of transfer functions for direct volume rendering, in Proceedings of the IEEE Symposium on Visualization, pp. 79-86, 1998.

Kindlmann G., Transfer functions in direct volume rendering: design, interface, interaction, SIGGRAPH 2002 Course Notes, 2002.

Kindlmann G., Whitaker R., Tasdizen T., Möller T., Curvature-based transfer functions for direct volume rendering: methods and applications, in Proceedings of Visualization 2003, pp. 513-520.

Kniss J., Kindlmann G., Hansen C., Multidimensional transfer functions for interactive volume rendering, IEEE Transactions on Visuslization and Computer Graphics, vol. 8, no. 3, pp. 270-285, 2002.

Kohlmann P., Bruckner S., Kanitsar A., Gröller E., LiveSync: Deformed Viewing Spheres for Knowledge-Based Navigation, IEEE Transaction on Visualization and Computer Graphics, vol. 13, no. 6, pp. 1544-1551, 2007.

Krüger J., Westermann R., Acceleration techniques for GPU-based volume rendering, in Proceedings Visualization 2003, Seattle, USA, pp. 287 - 292, 2003.

Leven J., Corso J., Cohen J., Kumar S., Interactive visualization of unstructured grids using hierarchical 3D textures, in Proceedings of IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics 2002, pp. 37-44, 2002.

Lindemann F., Ropinski T., About the influence of illumination models on image comprehension in direct volume rendering, IEEE TVCG (Vis Proceedings) (accepted), 2011.

Lipton M. J., Bogaert J., Boxt L. M., Reba R. C., Imaging of ischemic heart disease. Berlin Heidelberg: Springer, 2002.

Malik M. M., Heinzl C., Gröller E., Cmparative visualization for parameter studies of dataset series, IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 5, 829-840, 2010.

Marschner S. R., Lobb, R. J, An evaluation of reconstruction filters for volume rendering, in Proceedings Visualization 1994, pp. 100-107, 1994.

Marwick T. H., Cardiac stress testing and imaging: A clinician's guide, Churchill Livingstone, 1996.

Max N., Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics, vol. 1, no. 5, pp. 99-108, 1995.

Meissner M., Huang J., Bartz D., Mueller K., Crawfis R., A practical evaluation of popular volume rendering algorithms, VVS '00 Proceedings of the 2000 IEEE Symposium on Volume Visualization, pp. 81 - 90, 2000.

Meyer-Spradow J., Stegger L., Doring C., Ropinski T., Hinrichs K., Glyph-based SPECT visualization for the diagnosis of coronary artery disease, IEEE Transaction on Visualization and Computer Graphics, vol. 14, no. 6, pp. 1499-1506, 2008.

Meyer-Spradow J., Ropinski T., Mensmann J., Hinrichs K., Voreen: a rapid-prototyping environment for ray-casting-based volume visualizations, IEEE Computer Graphics and Applications, vol. 29, no. 6, pp. 6-13, 2009.

Miller F. P., Vandome A. F., McBrewster J., CG (programming language), VDM Publishing House Ltd., 2011.

Mortenson M. E., Mathematics for computer graphics applications, second edition, Industrial Press, New York.

Neumann L., Csebfalvi B., König A., Gröller E., Gradient estimation in volume data using 4D linear regression, in Proceedings of Eurographics 2000, pp. 351-358, 2000.

Nguyen, H., GPU gems 3, Addison-Wesley Professional, 2008.

Ohno Y., CIE fundamentals for color measurements, in Proceedings of the 2000 IS&T NIP16 International Conference on Digital Printing Technologies, Vancouver, Canada, pp. 540-545, 2000.

Olowoyeye A., Tuceryan M., Fang S., Medical volume segmentation using bank of Gabor filters, in Proceedings of the 2009 ACM symposium on Applied Computing, pp. 826-829, 2009.

Patel D., Bruckner S., Viola I., Groeller E., Seismic volume visualization for horizon extraction, in Proceedings of the IEEE Pacific Visualization Symposium 2010, Taipei, Taiwan, pp. 73-80, 2010.

Pedersen M., Hardeberg J. Y., A new spatial hue angle metric for perceptual image difference, Computational Color Imaging, vol. 5646, pp. 81-90, 2009.

Petrou M., Petrou C., Image processing: the fundamentals, second edition, Wiley, 2010.

Pinto D. M., Freitas, C. M. D., Design of multi-dimensional transfer functions using dimensional reduction, in Proceedings of Eurographics/IEEE VGTC Symposium on Visualization 2007, EuroVis, pp. 131-138, 2007.

Prassni J.-S, Ropinski T., Hinrichs K., Uncertainty-aware guided volume segmentation, IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 6, 2010.

Rajan S. S., MRI: a conceptual overview, Springer-Verlag, New York, 1997.

Rieder C., Ritter F., Raspe M., Peitgen H-.O., Interactive visualization of multimodal volume data for neurosurgical tumor treatment, Computer Graphics Forum, vol. 27, no. 3, pp. 1055-1062, 2008.

Ray H., Pfister H., Silver D., Cook, T. A., Ray casting architectures for volume visualization, IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 3, pp. 210 - 223, 1999.

Rezk-Salama C., Kolb A., A vertex program for efficient box-plane intersection, in Proceedings Vision, Modeling and Visualization (VMV), pp. 115-122, 2005.

Rost R. J., Licea-Kane B., Kessenich J. M., OpenGL shading language, third edition, Addison-Wesley Professional, 2009.

Ropinski T., Preim B., Taxonomy and usage guidelines for glyph-based medical visualization, in Proceedings of SimVis'2008, pp.121-138, 2008.

Ropinski T., Rezk-Salama C., Hadwiger M., Geier G., Ljung P, GPU-Based Volume Ray-Casting with Advanced Illumination, Tutorial Notes, IEEE Visualization 2008.

Schpok J., Simons J., Ebert D. S., Hansen C., A real-time cloud modeling, rendering, and animation system, in Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 160 - 166, 2003.

Seeram E., Computed tomography: physical principles, clinical applications, and quality control, Saunders/Elsevier, 2009.

Sereda P., Vilanova A., Gerritsen F.A., Automating transfer function design for volume rendering using hierarchical clustering of material boundaries, in Proceedings of EuroVis 2006, pp. 243-250, 2006.

Sherbondy A., Houston M., Napel S., Fast volume segmentation with simultaneous visualization using programmable graphics hardware, in Proceedings Visualization 2003, Seattle, MA, USA, pp. 171 - 176.

St-Laurent S., The complete effect and HLSL guide, Paradoxal Press, 2005.

Szabo T. L., Diagnostic ultrasound imaging: inside out, Elsevier, Burligton, MA, USA, 2004.

Tappenbeck A., Preim B., Dicken V., Distance-based transfer function design: specification methods and applications, in Proceedings of SimVis 2006, Magdeburg, Germany, pp. 259-274, 2006.

Taylor K. J. W., Johnson S. A., Ayers W. R., Medical imaging techniques: A comparison. Journal of Clinical Ultrasound, vol. 9, no. 5, 1981.

Termeer M., Comprehensive visualization of cardiac MRI data, PhD thesis, Vienna University of Technology, 2009.

Titov S. I., Perceptually based image comparison method, Graphicon'2000, pp. 12-19.

Viola I., Kanitsar A., Gröller E., Hardware-based nonlinear filtering and segmentation using high-level shading languages, in Proceedings of IEEE Visualization 2003, pp. 309-316.

Wang P., Hsia H. H, Al-Ahmad A., Zei P. C., Ventricular Arrhythmias and Sudden Cardiac Death: Mechanism, Ablation, and Defibrillation, Wiley-Blackwell, 2008.

Westbrook C., Roth C., MRI in practice, Third Edition, Blackwell Publishing Ltd, Oxford, UK, 2005.

Wilson D. L., Baddeley A. J., Owens R. A., A new metric for grey-scale image comparison, International Journal of Computer Vision., vol. 24, no. 1, pp. 5-17, 1997.

Zhou J., Doring A., Toennies K. D., Distance based enhancement for focal region based volume rendering, in Proceedings of Bildverarbeitung für die Medizin, Berlin, pp. 199-203, 2004.

Zhou J., Takasuka M., Automatic transfer function generation using contour tree controlled residue flow model and color harmonics, IEEE Transactions on Visualization and Computer graphics, vol. 15, no. 6, pp. 1482-1488, 2009.

List of Publications

Publications in ISI - indexed journals and proceedings:

Gavrilescu M., Manta V., Computation and representation of left-ventricular wall parameters from multidimensional MR cardiac data, in Proceedings of Computer Graphics and Imaging 2010, Innsbruck, Austria, pp. 128-135, 2010.

Gavrilescu M., Manta V., Advances in the Visualization of Three-Dimensional Seismic Volume Data, Environmental Engineering and Management Journal, vol. 10, no. 4, pp. 567-578, 2011.

Herghelegiu P., **Gavrilescu M.**, Manta V., Visualization of segmented structures in 3D multimodal medical data Sets, Advances in Electrical and Computer Engineering*,* vol. 11, no. 3, pp. 99-104, 2011.

Publications in journals and proceedings indexed in other international databases

Gavrilescu M., Manta V., Volume visualization applied in medical imaging, Bulletin of the Polytechnic Institute of Iasi, LIV (LVIII), 3-4, pp. 43-52, 2008.

Gavrilescu M., Manta V., Purgathofer W., Post-Rendering Enhancement of Volumes*,* Bulletin of the Polytechnic Institute of Iasi, LV(LIX), 3, pp. 43-54, 2009.

Gavrilescu M., Malik M. M., Gröller E., Custom interface elements for improved parameter control in volume rendering. Proceedings of the 14th International Conference on System Theory and Control, Sinaia, Romania, pp. 219-235, 2010.

Gavrilescu M., Malik M. M., Gröller E., Enhanced Interfaces for Parameter Adjustment in Volume Rendering Applications, Bulletin of the Polytechnic Institute of Iasi, LVI (LX), 4, pp. 163-174, 2010.

Gavrilescu M., Manta V., Gröller E, Gradient-Based Classification and Representation of Features from Volume Data, 15th International Conference on System Theory and Control (accepted)*,* Sinaia, Romania, 2011.