UNIVERSITÄT SIEGEN
Computergraphik & Multimediasysteme
Theorie
und Praxis
für Karrieren
von morgen

Fachbereich 12 - Elektrotechnik und Informatik

Angewandte Informatik mit Anwendungsfach Medienwissenschaften

Diplomarbeit

Transfer Function Design Based on Statistical Shape Models

Design von Transferfunktionen basierend auf Statistical Shape Models

Peter Kohlmann

Matr.Nr. 591878

Betreuer: Prof. Dr. Andreas Kolb
Dr. Christof Rezk-Salama

Tag der Ausgabe: 01.08.2005
Tag der Abgabe: 31.01.2006

Abstract

This thesis examines Principal Component Analysis for transfer functions, based on an initially generated set of manually assigned transfer functions with respect to a very specific type of data set and a strictly delimited type of application. The process of transfer function design is decoupled from the specialized knowledge about the transfer function domain (intensity, gradient magnitude etc.). Transfer function design is difficult because of the high degrees of freedom and the lack of a truly goal-directed process.

Existing approaches have been developed for automatic and semi-automatic transfer function design. These can be categorized as image-driven and data-driven techniques. To concentrate on the anatomical or functional structures which are interesting for the user, an application-driven method is needed. For a well-defined application scenario it is possible to reduce the complexity of transfer function generation by restricting the classification process to structures of interest for a specific examination procedure.

First of all, transfer functions are manually generated for an initial collection of volume data sets that has been recorded for a specific clinical purpose. A single transfer function is represented by a set of parameters of geometric primitives (ramps or trapezoids). Each of these individually assigned transfer functions can be regarded as a point sample in the (high-dimensional) parameter space of the transfer function model. From this set of point samples in parameter space a statistical shape model is created by applying Principal Component Analysis. A higher-level transfer function model with only a very limited set of parameters based on this analysis is established. This makes the process of transfer function setup very simple and intuitive.

Acknowledgements

First of all, I would like to thank Christof Rezk-Salama for all his help and advise. He did a great job in his supervision in every respect. I also want to thank him for making my internship in Princeton at Siemens Corporate Research possible. A good part of the practical work at Siemens was also supported by Gianluca Paladini, Thomas Möller and Klaus Engel. Their ideas and the discussions with them were a great source of inspiration. It has been my privilege to work with such an experienced and capable group of people. Maik Keller was a great co-worker at the implementation of the transfer function editor. I also would like to express my gratitude to Andreas Kolb who gave me some important advice for writing about my work and some formal aspects. In addition, I would like to thank Miriam Boenheim and Bernd Kohlmann that they took the time to work through my manuscripts.

Finally, I want to thank my parents for their enduring emotional and financial support throughout the years.

Contents

Part I

English Part

Chapter 1

Introduction

The 3D visualization of medical data has become more and more powerful within the recent years because of considerable improvements of methods and applications for this process. However, in spite of a great benefit of the usage of these possibilities in the clinical environment, the tools and applications are not yet fully accepted by surgeons and radiologists. In general, there are two categories of techniques for the visualization of the structure of volume data sets. Indirect methods require a pre-processing step to transform the data values to surfaces descriptions before they can be rendered. A very popular indirect method is the *Marching-Cubes*-algorithm proposed by Lorensen and Cline [28] for the extraction of isosurfaces.

However, the most powerful approaches for the visualization of the structure of volume data sets are the techniques of direct volume rendering. In contrast to indirect methods there is no need for a surface extraction. The data values are mapped to a rendered image in a more direct way and time-consuming geometric calculations are avoided.

Classification denotes the process of mapping scalar values to optical properties and is done by transfer functions. The assignment of a good transfer function is the central step in achieving a high-quality rendering result for the specific purpose. A transfer function performs the mapping of the data values to optical properties (color and opacity) which are the bases for the displayed image. As the transfer function is of such importance, the process of finding a good one has to be improved. The development of an easy-to-use transfer function editor was the first step. In a second step, a higher-level transfer function model was implemented. The input for the calculations is a set of individually assigned transfer functions for an initial collection of volume data sets. With these information the method can produce a very limited set of parameters to reduce the complexity of the process of transfer function generation for the end user.

In the next section there is a focus on the importance of 3D visualization. Following, the goal of this work is described. A chapter overview finalizes the introductory part of this thesis.

1.1 Motivation

First of all, there is the question who need 3D visualization in the medical scope. Figure 1.1 compares a single slice image of a computed tomography (CT) recording of the human head (A) with a reconstruction of the data via volume rendering (B). It can be seen that the result of the volume rendering offers various possibilities which are not offered by the single slices.

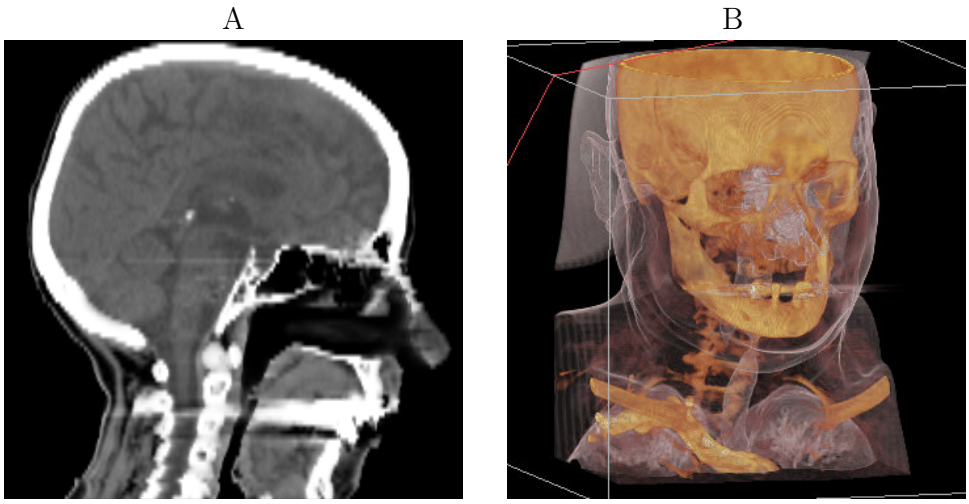

Figure 1.1: A typical slice image of a CT recording of the human head (A) and a volume rendering result of a data set (B).

The first group which can greatly benefit from 3D visualization in the different steps of an intervention are surgeons. They need to collect as much information as possible for pre-operative planning and assessment of the risks. During the intervention 3D visualization is a central part of computer-assisted surgery (CAS). CAS aims at an optimized synergy between people and machines in a combination of the techniques of advanced image processing and the recent developments in robotic and mechatronic surgical instruments. This technical assistance can help to achieve a highly accurate execution of surgical interventions. According to Joskowicz and Taylor [15], this integration of methods alters the procedures in the operating rooms of the 21st century fundamentally. After the intervention, 3D visualized data can support the post-operative evaluation.

The second group who benefit from 3D visualization are radiologists. Although they are used to work with slice images and they adopted a certain ability to reconstruct the 3D object mentally, there are still some complicated cases which will be improved by using 3D visualization. Especially tiny structures like vessels and nerves can be recognized and analyzed more accurately, due to the fact that 3D rendering results can show complex structures much better than the single slices.

3D visualization of medical data requires explorative methods and applications for the visualization process with a high degree of clear and simple user interaction.

The applications have to account for three principal aspects to ensure usability by physicians. At first, there is a high importance of the reproducibility of the whole visualization process. Without it, the physician will not rely on the results. Second, the speed aspect is very important. Especially the process of assigning color and opacity to the data values via transfer functions is a time-consuming process which needs to be facilitated. Besides the need of reproducibility and the speed aspect, the tools for the process have to be easy to use. The key to meet these demands is the development of intuitive user interfaces and the automation of certain tasks. Since the software which provides this visualization is often very complex, this complexity has to be hidden as much as possible from the user.

1.2 Goals

After this outline of the demands on the 3D visualization process, the goal of this work can be defined: The process of transfer function setup should become as simple and intuitive as the *greyvalue windowing* for slice images. Only if this can be guaranteed, there will be a high acceptance of these advanced image processing methods in the clinical process. For the transfer function design it is not sufficient to have a good knowledge of the volume rendering algorithm. In addition, it is important to know the interesting anatomical and functional structures which are contained in the data set.

A first step towards the realization of these demands is the development of a transfer function editor which is very easy to use. Although the transfer functions have high degrees of freedom, the transfer function setup can be managed with simple primitives such as ramps and trapezoids. After those primitives are added, they can be easily moved by picking the primitive or certain movable parts like its edges or points.

The second step is based on the generation of a set of manually assigned transfer functions for a specific type of data sets and a strictly delimited type of application. Through this goal-oriented process a basis for the reduction of the high degrees of freedom of the transfer function setup is produced. There are two underlying principles of the existing approaches for the transfer function design. *Image-driven* techniques are based on a trial-and-error generation of images to navigate the space of transfer functions. With this approach the user can examine the results to find the best rendering. *Data-driven* techniques are based on the analysis of the volume data itself. Neither choice integrates the knowledge about what anatomical or functional structures are interesting for the user. To overcome this drawback an *application-driven* method for the transfer function design is investigated.

As the user usually is interested in the generation of a transfer function for a specific examination procedure, the classification process has to be restricted in this method. This can be accomplished through the manual setup of transfer functions for an initial collection of volume data sets which is recorded for a specific clinical purpose.

With the developed transfer function editor the transfer function is represented as

a set of geometric primitives (ramps and trapezoids) which are positioned by the user. Thus, the transfer function can be parameterized that each of the adjusted transfer functions is regarded as a distribution of points in the high-dimensional parameter space of the transfer function model. The dimension of this space is corresponding to the number of modifiable parameters. This set of point samples is the input data for an algorithm which is based on the statistical methods of the *Principle Component Analysis* (PCA). The result is a higher-level transfer function model which provides a very limited set of parameters to navigate the space of appropriate transfer functions for the specific application.

1.3 Structure

Now the motivation and the goal of this thesis are outlined. Chapter 2 starts with some important topics of computed tomography. After pointing out how the data is acquired, an introduction of image reconstruction techniques is provided. Following, there is an outline of the principles how the data can be displayed. A description of the basic techniques for modern volume visualization closes this chapter.

In Chapter 3 the theory of one-dimensional and multidimensional transfer functions, as well as the design of transfer functions is introduced. As previous work and existing approaches are always a great inspiration for new ideas there is also a presentation of interesting methods for the transfer function design.

The results of the development of a sophisticated and easy-to-use transfer function editor are outlined in Chapter 4. The focus is on the architecture and the most important features of this application.

In Chapter 5 the mathematical foundations for the understanding of the applied statistical methods are provided. There is also a focus on the benefit of principal component analysis and on the application of this procedure to the investigation of transfer function design.

Following, in Chapter 6 the implementation of the PCA algorithms for the purpose of transfer function design is presented. Furthermore, it is described how an intuitive user interface facilitates the fast transfer function setup.

Chapter 7 shows the results of the done work for a specific clinical application scenario. Finally, there is a conclusion with a summary of the work and an outlook on possible future work for the improvement of the transfer function design in Chapter 8.

Chapter 2

Data Processing

It does not make sense to discuss the visualization of data without the basic knowledge about the data source and the data acquisition. This chapter has a focus on the basic principles of computed tomography and the techniques for volume visualization.

2.1 Computed Tomography

Before computed tomography was introduced, the classical x-ray imaging was the standard defect detection technique. Through creating a projection of the internal structure, the traditional method provided information for the determination of certain defects with the lack of detailed volumetric information. In 1972, the first practical implementation of computed tomography was presented by the English engineer G. N. Hounsfield [10] and in the late seventies there was a huge demand for CT scanners. In 2000, almost three decades after its invention, approximately 30.000 whole body CT scanners were installed in clinical environments. These scanners provide digital images of single discrete slices as a representation of the volume [18].

2.1.1 Measuring

The classical x-ray radiography records the relative distribution of the x-ray intensity to provide a greyvalue image for the diagnostic purpose. In CT, the intensity of x-rays is also recorded behind the object to determine the attenuation of intensity I caused by the object in addition to the initial x-ray intensity I_0. The basic equation for the attenuation is *Beer's Law* [19], defined as

$$I = I_0 \, exp\,[-\mu x],$$

where μ is the linear coefficient of the attenuation and x is the length of the x-ray path through the material. But as there are usually a number of different materials within the scanned object, the equation has to be expanded to

$$I = I_0 \, exp\Big[\sum_i (-\mu_i x_i)\Big],$$

where each i represents a single material with the attenuation coefficient μ_i and the path length x_i. In the general case the summation has to be replaced by the integral over μ along the ray path because the summation has to be carried out with very small increments. To get the final equation for attenuation, the integration over the range of the x-ray energy spectrum has to be included which lead to

$$I = \int I_0(E)\, exp\Big[\sum_i (-\mu_i(E)x_i)\Big] dE.$$

For the computation of high-quality images a high number of recordings from various directions is required. The covered angular space has to be at least 180°. In

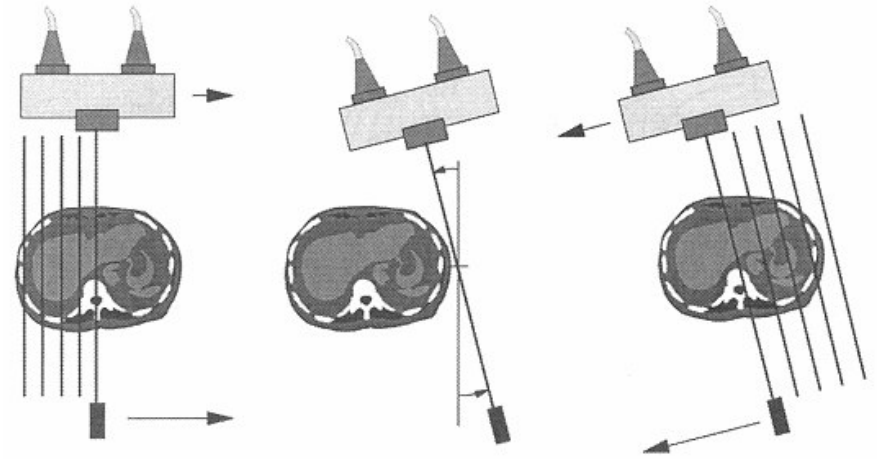

Figure 2.1: The measuring process of an object in CT [43].

Figure 2.1, a simplified measurement setup to illustrate the principle CT scanning-cycle is presented. For a given angular position a radiation source emits a pencil beam and the opposite placed detector measures its attenuated intensity. To cover the width of the object, the radiation source and the detector perform a translation tangential to the rotation circle. This set of parallel rays is recorded as an intensity profile. In the setup of early clinical CT scanners the usual procedure was the recording of 180 projections each with an 1° angular displacement and 160 measured data points per projection. A modern CT scanner covers an angular range of 360°. Within this range it fulfils the recording of $800 - 1500$ projections with $600 - 1200$ data points for each projection. The reason for this enormous increase of recorded data is the focus on an improved image quality and data sampling [18].

The *cutting edge* of CT scanner development is the multi-slice spiral scanner. This mechanism is able to collect four slices of data in 250 to 350 ms. With the rotation speed of 120 rpm (rotations per minute), the collection of data is performed in one eighth of the time previous spiral systems needed. The reconstruction of a 512×512-matrix image from millions of data points can be achieved in less than one second.

Multi-slice spiral CT scanning enables the recording of an entire human chest (forty 8 mm slices) in 5 to 10 seconds [12].

2.1.2 Image Computation

After the introduction how the CT scanner records the data there is the question how a CT image is computed. The distribution of the attenuation coefficients $\mu(x, y)$ is given as a set of projection values. An inverse transformation has to be performed to calculate this distribution. One way to do this is the solving of N_x independent equations to compute the N^2 unknown values of the matrix with NxN pixel-values. N_x is the product of the number of projections and the number of data points per projection. A solution is only possible if N_x is larger or equal than N^2.

This approach to find the solution to the problem of three-dimensional reconstruction from projections was introduced by Gordon et al. [4] in 1970 as the *Algebraic Reconstruction Techniques* (ART). A simple case for demonstration purpose with only four pixels (2×2-matrix) which present two measurements for two projections is shown in Figure 2.2. The values of A, B, C and D can be calculated by solving a

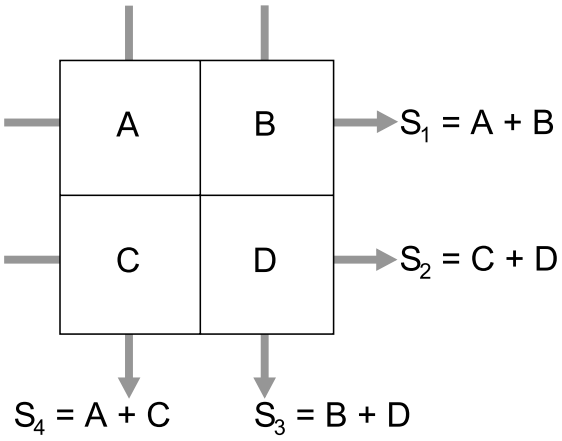

Figure 2.2: Algebraic procedures for the computation of a CT image.

system of linear equations. Assuming that $S_1 = 7$, $S_2 = 6$, $S_3 = 8$, and $S_4 = 7$, an equation can be arranged which reads

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{pmatrix} = \begin{pmatrix} 7 \\ 6 \\ 8 \\ 7 \end{pmatrix}.$$

This equation is of the form

$$K\,x = b.$$

The multiplication with K^T leads to

$$K^T K\,x = K^T b.$$

Finally, the multiplication with the inverted square matrix $[K^T K]^{-1}$ is performed to obtain

$$x = [K^T K]^{-1} K^T b.$$

The solution of this equation with the example values is $A = 2$, $B = 5$, $C = 4$ and $D = 3$.

By adding projections and data points the calculation effort increases immensely. As the CT recording usually generates much larger matrices, the equations have to be solved by the iterative application of standard methods for solving large matrix operation problems. This would lead to a high computational effort of ART for larger data volumes.

The second method of CT image computation is the *convolution-backprojection procedure*. Herman [7] describes the *summation* or *backprojection method* as the simplest algorithm for reconstruction. The sum of all rays through one point has to be added to estimate the density at this point. For the *simple backprojection*, the start is the allocation of an empty image matrix filled with zeros as initial values. Then, the matrix is filled by adding each projection value to all picture elements which are represented by the defined matrix along the direction of the measuring. As a consequence of this procedure an entry of the matrix represents not only the value for the desired image point but information about other parts of the image as well. The result is an unsharp image where the details of the scanned object can be recognized because of high intensities.

Figure 2.3 is based on an example of Herman [7] and illustrates the simple backprojection of a single point with a small number of projections. The result is not a single point but a star-shaped object with the original point in its center. Because

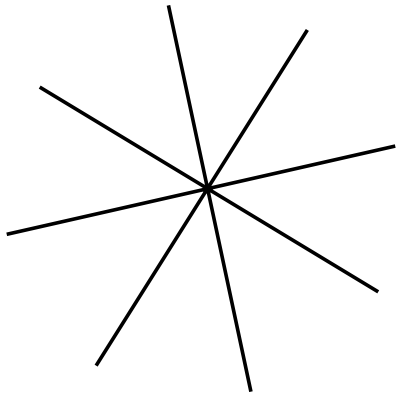

Figure 2.3: Backprojection of a single point.

of this phenomenon the simple backprojection alone does not work as a satisfactorily reconstruction method for computed tomography. The reason for this insufficient reconstruction is a wrong dimensionality of the values produced by the backprojection. To solve the problem of unsharp images, each projection needs to be convoluted with a so-called *convolution kernel* before its backprojection. These kernels can be used as high pass filters to increase or decrease the boundaries of the object. Beside its basic

function, the choice and design of the kernel influence the image properties and lead for example to a smooth or sharp result. A weak high pass filter reduces the noise and the spatial resolution of the image.

Besides the two presented reconstruction approaches the fourier methods have to be mentioned. These methods are mathematical equivalent to the convoluted backprojection and may become more widely used in the future. As the algebraic reconstruction techniques lost its importance with increasing resolutions, the convoluted backprojection is the dominant implementation for CT image reconstruction today [18].

2.1.3 Display

The result of the CT measuring and reconstruction process is a series of 2D images which are called slices. Because of the limited expressiveness of the attenuation coefficients $\mu(x, y)$, they are not stored directly. Instead, the attenuation of water serves as basis for the calculation of CT values. These values are specified in *Hounsfield Units* (HU). The definition of the CT value for a certain tissue t with μ_t as attenuation coefficient is described by the equation

$$CTvalue_t = (\mu_t - \mu_{water})/\mu_{water} \times 1000HU.$$

On the scale of Hounsfield Units which is shown in Figure 2.2, water has an attenuation value (HU) of zero. CT values of lung and fat are in the negative area because of their low density. As listed in Table 2.1, the density of most of the other body parts is higher than the attenuation of water. The numbers represent a certain grayvalue from white (+1000) to black (−1000). Although the usual range of this scale is from −1000 to +1000, some modern medical scanners provide a range of 4096 (= 2^{12}) values from −1024 HU to 3071 HU [18].

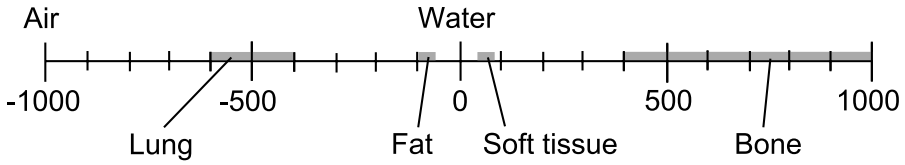

Figure 2.4: The Hounsfield scale based on [13].

Bone	+400 → +1000
Soft tissue	+40 → +80
Water	0
Fat	−60 → −100
Lung	−400 → −600
Air	−1000

Table 2.1: The range of Hounsfield Units for certain tissues and materials.

The 2^{12} CT values would lead to 4096 shades of grey but the maximum number which can be distinguished accurately by the human eye is $60 - 80$. Because of this fact, the various shades of grey are assigned to a certain window, representing the CT value interval of interest. A *window width* defines a number of HUs, and a central HU is represented by the *window level*. Then, a set of greyvalues is assigned to the covered HUs and all other values are displayed either black or white. How this *windowing* procedure can influence the displayed image is shown for the CT examination of the the human chest in Figure 2.5. If the soft tissue (mediastinum) is interesting for the observer, a window width of 350 and a window level of +40 will be a good choice (A). To get a meaningful result for the details of the lung parenchyma a window width of 1500 and a window level of -600 is assigned (B) [13].

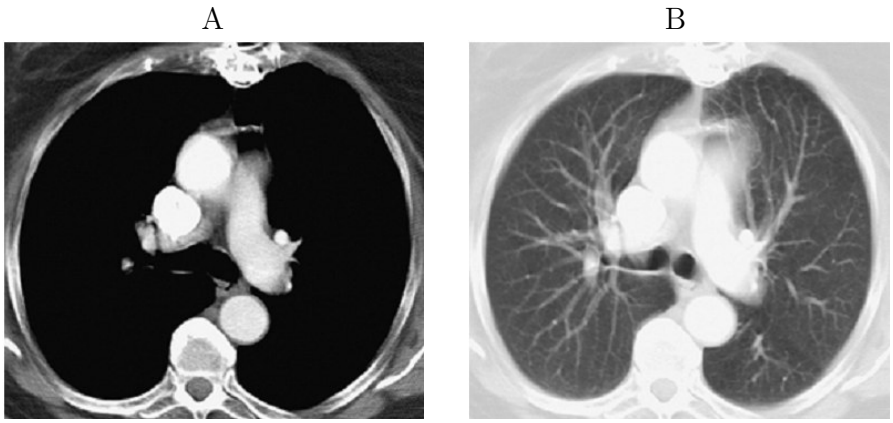

Figure 2.5: The result of windowing procedures to display CT images. In (A) the soft tissue is accentuated and the tissue of interest in (B) is the lung field [13].

2.2 Volume Visualization Algorithms

Within the last decade the visualization of volume data became more and more important for medical applications. Usually, the objects under examination are represented by a three-dimensional grid of volume elements (called *voxels*) measured by computer tomography (CT), magnet resonance imaging (MRI) or ultrasound (US). If the grid is uniform rectilinear, a three-dimensional array of cubic elements will represent the discrete volume data set. The great benefit of volume data is its ability to represent the inner structure of the objects. While the indirect methods for volume visualization like *isosurface extraction* compute a surface from the data, the direct approaches generate the image without the need for a surface representation. Popular algorithms for direct methods are *ray casting*, *texture slicing*, *shear-warp factorization* and *splatting*. Figures 2.6, 2.10, 2.11 and 2.12 are inspired by Pfister's description of the algorithms [34].

2.2.1 Ray Casting

Ray casting is based on the physical process of emission and absorbtion of radiative energy. This technique was introduced by Kajiya and Herzen [17] in 1984 and gained popularity because it achieves a very high image quality with a simple and optically correct algorithm. It is an *image-order* algorithm because the value for each pixel

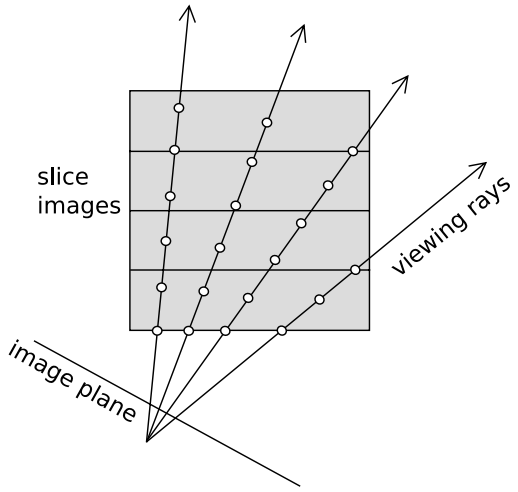

Figure 2.6: The principle of ray casting.

in the 2D image is calculated by the influence of multiple samples in the volume. Rays of sight are projected through the volume passing the pixels of the image plane as illustrated in Figure 2.6. Along these rays sampling points are calculated in a uniform distribution. The data values at these points have to be reconstructed which is usually done with trilinear interpolations.

A gradient vector is calculated at each sampling point for the classification and the shading. The process of classification maps physical properties of the volume such as density to optical properties (color and opacity). Depending on wether the classification is performed before or after the interpolation it is called *pre-classification* or *post-classification*. To achieve high-quality and more realistic images, a local illumination model can be applied to the volume points. The accumulation of all classified points along the ray of sight is the resulting value for one pixel. The final image is computed as the sum of all rays. An even better image quality can be produced by shooting several rays through one pixel and the computation of a combination of these rays.

Ray casting serves as a reference in terms of image quality for the other algorithms because of its modeling of the physical transport of light. The drawback of ray casting is that it is computationally rather expensive. With the implementation of ray casting on usual graphics hardware presented by Roettger et al. [42] in 2003, this algorithm is able to produce interactive frame rates [34].

2.2.2 Texture Slicing

Besides the performance, the quality of the images is highly important in the scope of scientific visualization. In most of the cases it is crucial that the information in the data set is displayed exactly. To meet these demands, the interpolation method and the sampling rate are important. To achieve good visualization results, especially for small structures like blood vessels or nerves, it is helpful to provide adjustable sampling rates. Texture slicing is a very popular volume rendering algorithm and there exist approaches with 3D-textures, 2D-textures and 2D-multi-textures. These methods use the abilities of the *graphics processing units* (GPUs) for hardware accelerated interpolation [39].

3D Textures

Cabral et al. [2] presented the standard approach for texture based volume rendering with the usage of 3D-textures in 1994. Their approach serves as a reference regarding performance and image quality. For this method a support of 3D-textures and trilinear interpolation by the GPU is necessary. Because the volume is saved as a 3D-texture, there is the possibility of calculating texture slices aligned to the image plane. Figure 2.7 shows that the polygon slices need to be changed as soon as the

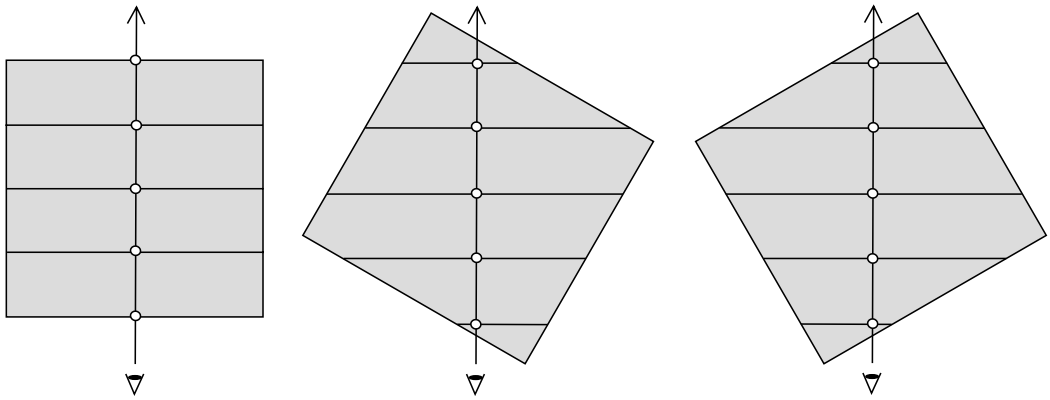

Figure 2.7: The principle of 3D-texture-slicing where the slices are aligned to the image plane.

eye position is changing in relation to the object. To meet this demand the textures have to be generated for each image. Interactive frame rates, even for scalar fields of high resolution, are achieved because the hardware supports operations for trilinear interpolation [39].

2D Textures

Although 3D-textures are supported by modern GPUs, 2D-texture-slicing is still an often used method of volume visualization. This method is based on the use of texture slices. As shown in Figure 2.8, a decomposition of the volume object into a stack of polygons which is aligned to the object has to be performed. The volume

has to be kept in the memory three times. The used texture stack is switched if the angle between viewing direction and the normal of the slice exceeds 45°. Caused by the change of this angle, the distance between the sample points changes by a factor between 1 and 2. Because of this effect, the integration of the sampling points

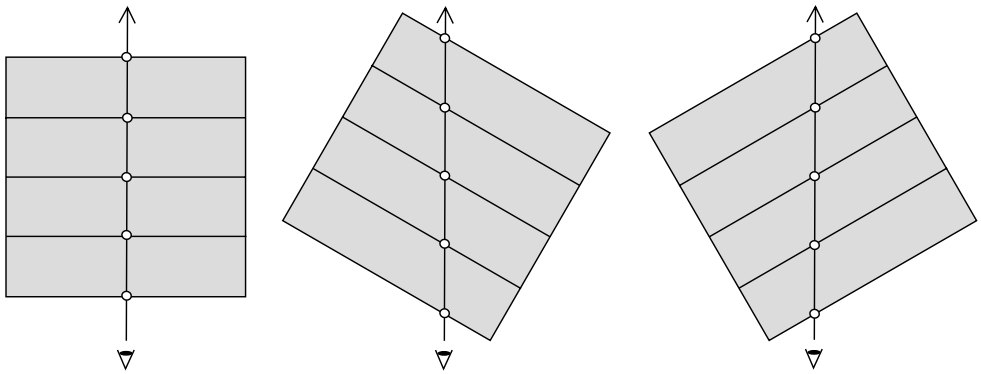

Figure 2.8: The principle of 2D-texture-slicing where the slices are aligned to the volume.

becomes incorrect. An adjustment of the sampling rate is not possible because of the fixed texture slices. This leads to visual sampling artifacts at the edges of the slices if a high zoom into the volume is performed. To get rid of these artifacts, Rezk-Salama et al. [38] introduced a method based on 2D-multi-textures [39].

Multi-Texture

This alternative approach of texture-sliced volume rendering allows trilinear interpolation for 2D-textures by the computation of intermediate slices on the fly. With this method visual artifacts caused by the fixed number of slices can be avoided as shown in Figure 2.9. Multi-texturing offers the possibility to assign multiple textures to one polygon within the rendering pipeline. With the use of these multi-textures the third interpolation step is performed by the rasterization hardware. The missing slice between two neighbored slices is computed by a blending operation of the surrounding slices S_i and S_{i+1},

$$S_{i+\alpha} = (1 - \alpha) \times S_i + \alpha \times S_{i+1}.$$

Bilinear interpolations are performed by the texture unit because the slice images are stored as 2D-textures. For trilinear interpolation a blending operation is computed for the two resulting *texels*.

Modern multi-texture hardware offers several possibilities of texture combination. The RGB texture-channels can be used for saving vectors and for the calculation of dot products. This allows an integration of the *Blinn-Phong illumination model* to texture based volume rendering [38, 39, 37].

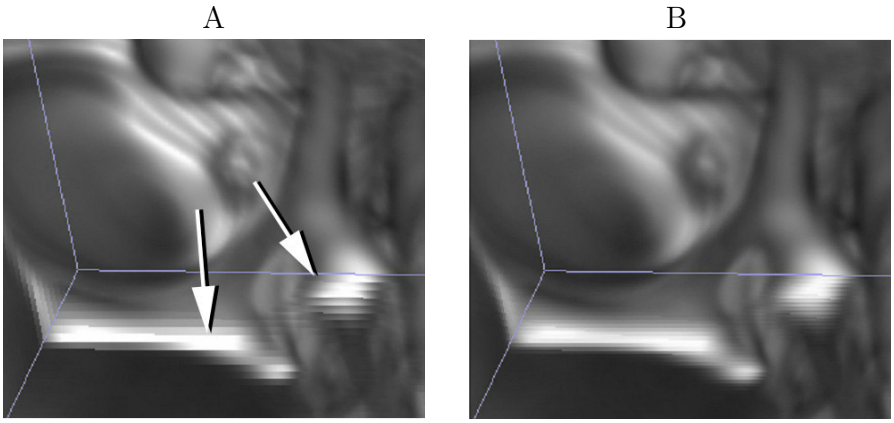

Figure 2.9: Without trilinear interpolation visual effects can be seen in the resulting image (A). Intermediate slices lead to much better results (B) [38].

2.2.3 Shear-Warp Factorization

Another very popular volume visualization algorithm is the shear-warp factorization. It was introduced by Lacroute and Levoy [25] in 1994 as a software solution but today it is implemented in hardware with high performance. In Figure 2.10 the

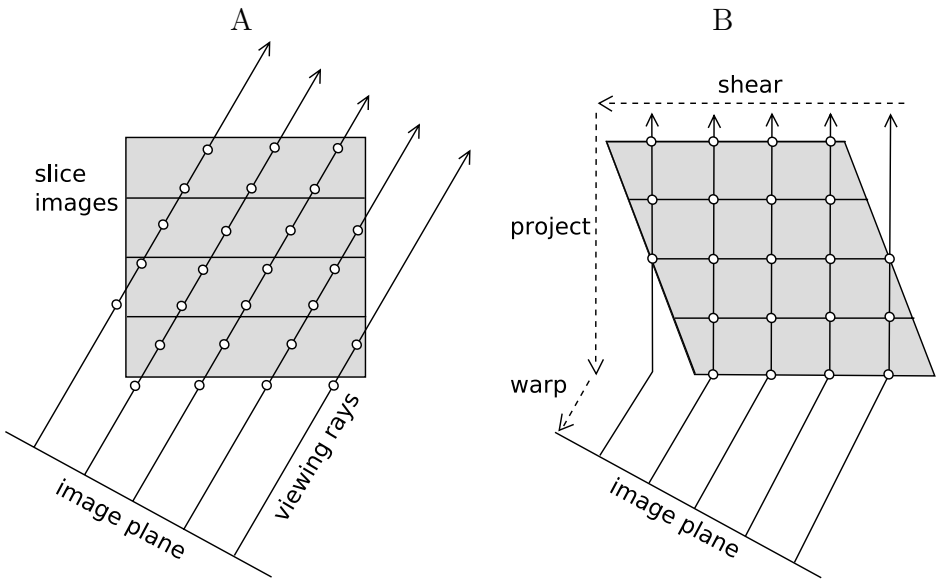

Figure 2.10: The principle of the shear-warp algorithm for parallel projection.

basic idea of the shear-warp algorithm for parallel projection is shown. First of all, the distribution of the sampling points along the rays has to be placed in a way that it matches exactly the slice images (A). This is necessary to replace the trilinear with bilinear interpolation. The projection of the volume slices can be factorized into a

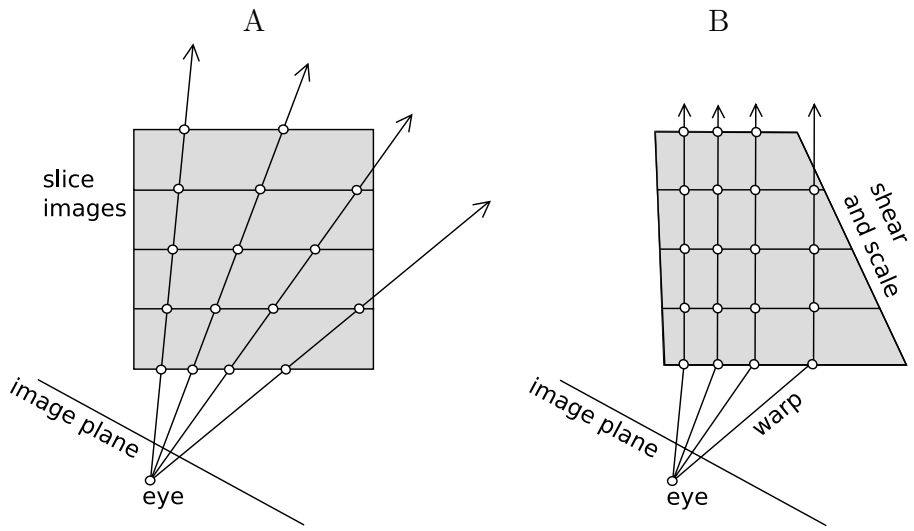

Figure 2.11: The principle of the shear-warp algorithm for perspective projection.

3D *shearing* and a 2D *warping* step to get the final image (B). Out of all algorithms for parallel projection the shear-warp factorization performs best.

If using the shear-warp factorization for perspective projection as illustrated in Figure 2.11, an additional scaling has to be applied to the slice images. In contrast to the method for the parallel projection, there are different angles between the viewing rays and the image plane. These angles have an additional effect on the distance between the single sampling points along the rays. The fact that the rows of voxels are aligned with the rows of pixels in the resulting image of the first factorization step (shear) leads to a fast computable interpolation process.

A drawback of the shear-warp algorithm is, that three copies of the volume have to be kept in the main memory to achieve an interactive rotation of the data set. The shear-warp algorithm is the fastest pure software implementation for direct volume rendering and achieves very high frame rates [37].

2.2.4 Splatting

Splatting is an *object-order* algorithm and in principle, it is an inversion of the ray casting algorithm. The ray casting approach calculates the value for each pixel in the 2D image by the influence of multiple samples in the volume. Splatting is based on the calculation of the influence of each data point on several pixels in the image. This method was introduced by Westover [45] in 1989.

For every data sample a convolution with a reconstruction kernel is applied with the purpose of getting a 3D image of this data sample. The convolution result is projected on the image plane and it contributes energy to to a number of pixels in the 2D image. Figure 2.12 illustrates the process of this projection and the generation of a color distribution on the image plane. As the reconstruction kernels are independent of the viewing direction and because they are the same for every data point, they can

be calculated in a preprocessing step. The projection of the energy distribution to the image plane is called the *footprint* of the data sample. To get the final image, all the footprints have to be integrated.

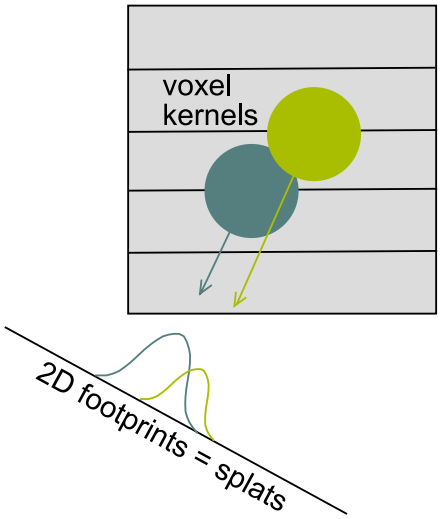

Figure 2.12: The principle of the splatting projection process.

Zwicker et al. [47] presented *EWA volume splatting*, a splatting method with a very high image quality. This approach integrates an elliptical Gaussian reconstruction kernel and a low pass filter. Aliasing artefact in the output image and excessive blurring are reduced with this method.

Chapter 3

Transfer Functions

In the previous chapter different methods for the visualization of volume data were presented. Now the focus is on transfer functions which are essential for the quality of direct-volume-rendered images. After the introduction of the principles of transfer functions the classification process is described. Multidimensional transfer functions are presented because they can be used for the generation of high-quality images. The chapter closes with the description of different existing approaches and chosen realizations of transfer function design.

3.1 Principles

The goal of volume rendering is the visualization of volumetric data. To discover the interesting structures inside the volume it is necessary to remove occluding elements. For a complete removal clipping planes and clipping primitives can be used. A more sophisticated approach is the application of transfer functions. They are required to make the scalar values in the volume data visible.

The rendering methods are based on the transport theory of light. He et al. [6] describe the integration of the effects of light interaction along viewing rays inside the data with the equation

$$I(a,b) = \int_a^b s(x)\, exp\Big[\int_a^x \alpha(t)\, dt\Big]\, dx.$$

$I(a,b)$ represents the intensity of a ray through the data set between the two points a and b. The source term $s(x)$ is the light which is added along the ray including self-emission and reflected light. Light attenuation along the ray is described by the absorption coefficient $\alpha(t)$.

This equation shows the requirement of a transfer function which maps the scalar value f in the data set to optical properties. The opacity transfer function is the function $\alpha(f)$ of the scalar value f that assigns opacities to the data values. Opacity is the most important optical property because it is possible to assign high opacities to the regions of interest. A transfer function $s(f)$ of the scalar f to get the source

term s can be used to assign color to the values in the data set. Colors are important to distinguish features in the data set.

Figure 3.1 illustrates the role of the transfer function in the process to get a 3D volume rendering from the volume data. With the set of slice images a three dimensional volume is generated. To get a visual representation with assigned colors and opacities the transfer function needs to be applied. There are also some alternative

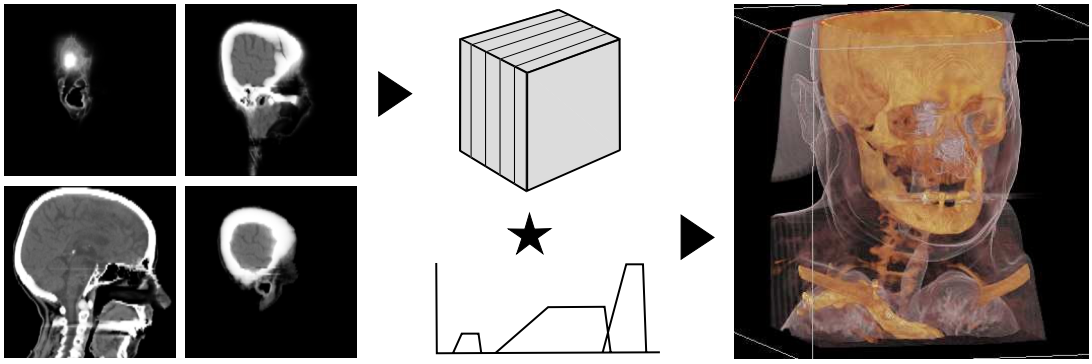

Figure 3.1: The simplified process to get a volume rendering result (right) via applying a transfer function to the volume data set.

techniques to create a three-dimensional rendering without numerical integration and transfer function adjustment. One way of simple compositing is the *Average Intensity Projection* (AIP). The resulting image is the average of the voxel-intensities along the corresponding rays and looks quite similar to x-ray photographs. This method is not very flexible. A similar compositing method is *Maximum Intensity Projection* (MIP). Instead of the average, the maximum intensity is used for the image generation. Although this technique leads to results with a confusing impression of depth, this method is helpful to visualize blood vessels with contrast medium. Examples for these alternative compositing methods are shown in Figure 3.2.

The way to assign the optical properties via transfer functions has the benefit of a high flexibility. Edges can be presented with a semi-transparency and certain areas in the data set can be displayed differently. A problem of transfer functions is their difficult and slow adjustment. Often the setup process is not very intuitive.

Kindlmann [20] describes the high flexibility of transfer functions both as a benefit and a drawback. The task of finding a good transfer function is extremely difficult and often a frustrating trial-and-error process. He states some major reasons for this. At first, it is because of the enormous degrees of freedom. Often this is too much for the user and he gets lost. Simple user interfaces use linear ramps for the definition of a transfer function. Even these geometric primitives add two degrees of freedom for each control point. Second, there is a lack of guidance because the user interfaces are not constrained or guided by a certain data set or a certain domain of interest. This leads to the trial-and-error process which becomes especially difficult if little changes of the transfer function lead to large changes in the rendered image. Because of these drawbacks it is very important to provide methods to make it easier to get good rendering results.

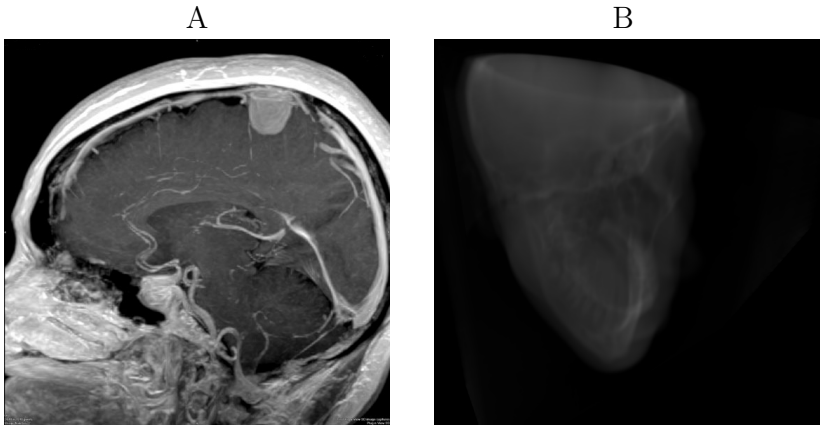

Figure 3.2: MIP can lead to good results for vessel examinations (A) [11]. The AIP method produces images which are similar to x-ray photographs (B) [46].

The way to realize this is the removal of unnecessary flexibility of the transfer function setup process and the providing of helpful guidance and information. A simple method to facilitate the process it the presentation of support data in form of a 1D histogram. The histogram provides information, for example on density of scalar values within the data set. This is useful to discover dominant structures which should be assigned with certain colors and opacities.

3.2 Multidimensional Transfer Functions

After the presentation of the principles of 1D transfer functions, the fundamentals of multidimensional transfer functions are introduced. Even though Levoy [27] already investigated the use of the magnitude of the gradient vector in 1988, the scalar value was the only dimension of interest to assign opacity and color until a few years ago.

Transfer functions of a higher dimension provide more flexibility to extract and visualize the features of interest in the volume data than 1D transfer functions. In medical scans recorded with CT or MRI techniques multiple materials have boundaries of a high complexity between each other. Because one data value often is connected with more than one boundary, there are problems to render the regions isolated with 1D transfer functions. Even if the boundaries are easy to identify in the spatial domain, an isolation in the domain of the transfer function is more complex. That is because the scalar value ranges of the interesting and the uninteresting regions within volume data can overlap.

One major challenge is to find an intuitive way for the transfer function setup because multidimensional transfer functions provide even more degrees of freedom than 1D transfer functions. Furthermore, it is a difficult task to adapt the implementation of 1D transfer functions as a *Linear Lookup Table* (LUT) to more than two dimensions. In general, there are two categories of multidimensional transfer functions. They are either applied to scalar data or to multivariate data.

3.2.1 Scalar Data

The first order derivative of a 3D scalar field $I(x, y, z)$ is the gradient which is defined using the partial derivatives of I in $x-$, $y-$ and $z-$direction as

$$\nabla I = (I_x, I_y, I_z) = (\frac{\partial}{\partial x} I, \frac{\partial}{\partial y} I, \frac{\partial}{\partial z} I).$$

An indicator for the local rate of change in the scalar field is the magnitude of the gradient, defined as the absolute value of ∇I with the formula

$$\|\nabla I\| = \sqrt{I_x^2 + I_y^2 + I_z^2}.$$

With the help of the gradient magnitude which represents an additional axis of the transfer function it is possible to distinguish regions of change (high gradient magnitude) and homogenous regions (low gradient magnitude).

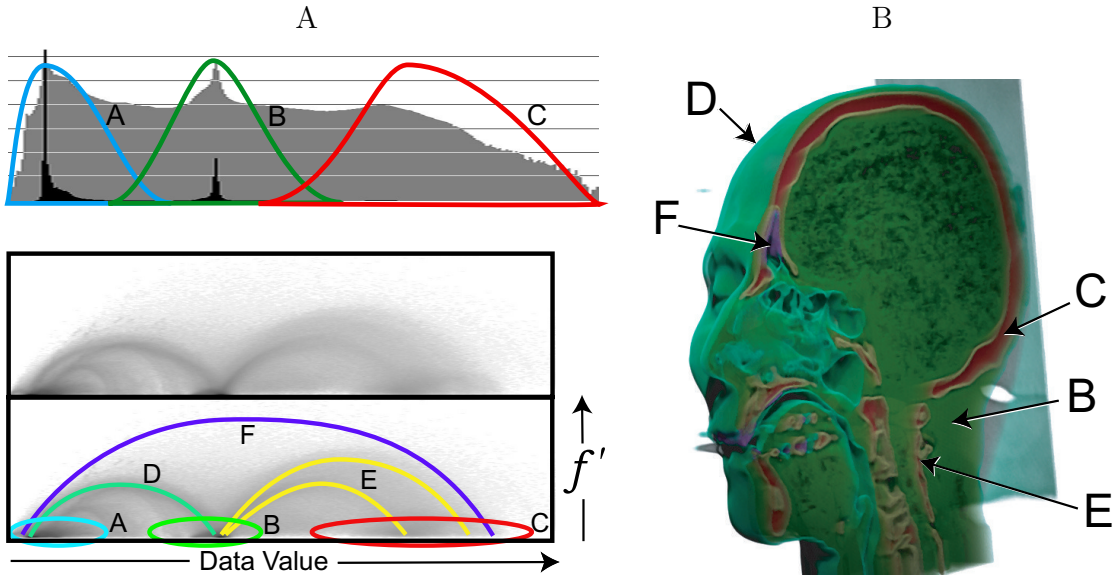

Figure 3.3: A 1D histogram (A,top) where the colored marks (A,B,C) identify the basic materials. A log-scale 2D joint histogram (A,bottom) with marked materials and material boundaries. The volume rendering of a head (B) is achieved with a 2D transfer function and shows materials and identified boundaries [24].

Figure 3.3 compares a 1D histogram and a log-scale 2D joint histogram on its left side. In the 1D histogram the three main materials in the CT data set of the head are identified as air (A), soft tissue (B) and bone (C). In the 2D histogram the circles A, B and C mark the materials with low gradient magnitude. The arches represent the high gradient magnitudes as the boundaries between air and soft tissue (D), soft tissue and bone (E) and air and bone (F). On the right side of Figure 3.3, a volume rendering of the head is shown with labels to the materials except air and the boundaries. With

the two dimensions, scalar value and gradient magnitude, it is possible to isolate the materials and boundaries. Especially the boundary between air and bone (F) cannot be isolated with transfer functions of solely one dimension.

But even the 2D transfer functions using the scalar value and the gradient magnitude are not always able to isolate the materials properly. Difficulties can appear if the arches in the 2D transfer function overlap. In these cases, the second order derivative of the 3D scalar field described by the Hessian matrix can help to put things right. With the three dimensions, scalar value, gradient magnitude and the magnitude of the second order derivative of the scalar field, it is possible to avoid an overlapping of the arches. This approach improves the isolation process of certain boundaries. The relationship between the scalar value and the first and second order

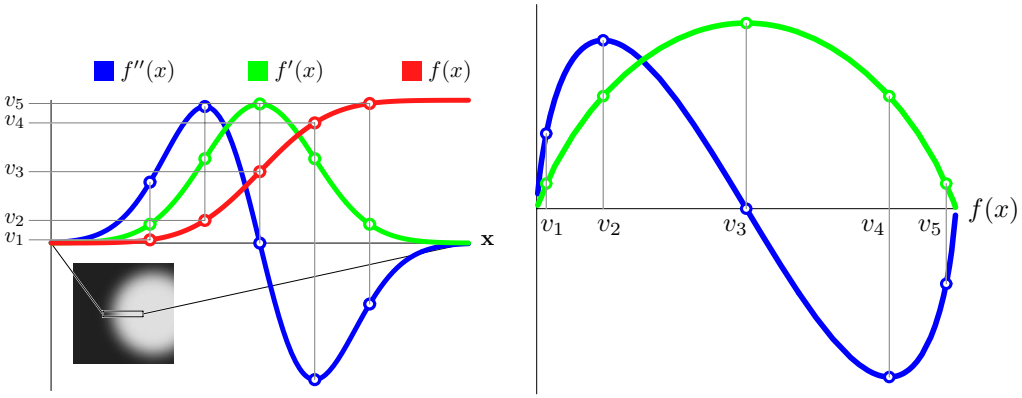

Figure 3.4: The relationship between scalar value and the first and the second order derivative of the 3D scalar field (left). Gradient magnitude and second order derivative as a function of data value (right) [24].

derivative of the 3D scalar field is illustrated in Figure 3.4 with an idealized material boundary as example. Most interesting for the boundary isolation is the maximum of the gradient magnitude where the second order derivative is zero (left). This point identifies the center of the boundary. On the right side of Figure 3.4, the gradient magnitude and the second order derivative are shown as a function of the data value. These are the curves which appear in the histogram for a 3D transfer function [24].

An approach introduced by Kindlmann et al. [22] uses the curvature information in multidimensional transfer functions for the improvement of non-photorealistic volume rendering.

3.2.2 Multivariate Data

Multidimensional transfer functions can be applied to multivariate data, too. Multivariate data contain more than one quantity at each sample point. These quantities can represent either measured or simulated values. MRI as a medical scanning method can record various tissue characteristics. It is also possible to compose data from different scanning methods such as MRI and CT to create a multivariate data set.

The different values can be represented by the axes of the transfer function. Multidimensional transfer functions can help to generate better images because they are more flexible in classifying the features of interest in the data set. Analog to the usage of the gradient magnitude for scalar values, the first order derivative is very useful to isolate and visualize the interesting materials and boundaries of multivariate data.

One field of application where multidimensional transfer functions offer good classification possibilities is volumetric color data. Various volumetric color data sets are available. The National Institute of Health's *Visual Human Project* [32] acquired color data by cryosection of male and female cadavers. These data sets are very good sources for the investigation of anatomy. A multidimensional transfer function can be utilized to assign opacity to different positions in the 3D space of RGB color. A good example for multivariate data aside from medical usage are weather data sets which can contain temperature, pressure and humidity as physical quantities [24].

3.3 Design

After the presentation of the theory of 1D and multidimensional transfer functions, it is time for a closer look into the design of transfer functions. As explained, the setup of a good transfer function is a very important but difficult task which becomes more and more complex with an increasing number of dimensions. Easy-to-use interfaces, as well as automatic and semi-automatic techniques can facilitate the transfer function setup. In the following sections, three classes of approaches, the techniques based on *interactive adjustment*, the *image-driven* and the *data-driven* techniques, are presented with example implementations.

3.3.1 Interactive Adjustment

Most scientific and commercial implementations of transfer function editors are based on a manual setup of transfer functions with the help of a visual editor. The implementations differ in the degrees of freedom and the kind of primitives that are used for the definition of the transfer function. Examples for used primitives are linear ramps, trapezoids, gaussian curves and splines. A precondition for the account of manual adjustment methods is an immediate visual feedback within the 3D viewer. Every small adjustment of the transfer function has to initiate a new rendering of the data set to allow goal-directed work.

To achieve the desired result with manual transfer function editors, some previous knowledge of the user about the specific data is helpful. As it is difficult to reproduce certain assignments for similar data sets, some implementation of transfer function editors provide templates for certain areas of examination. Such methods are well suitable for CT data sets because of the range of Hounsfield Units which defines the distribution of the scalar values for the different materials. After the application of a template the user can still make manual changes to get the desired result.

Kniss et al. [23] introduced a very sophisticated approach based on *direct manipulation widgets* for the interactive assignment of multidimensional transfer functions.

Direct Manipulation Widgets

This approach for interactive volume rendering with the use of multidimensional transfer functions provides direct manipulation widgets which are rendered geometric objects such as spheres, cylinders and cones. With these tools the user is able to select certain parts within the volume. The idea behind this method is to reduce the spatial problem. The assignment of opacities and colors with transfer functions is non-spatial because their domain does not include the spatial position as a variable. This leads to problems if the user wants to isolate spatial localized features within the volume which covers the same data values as other regions.

Three dimensions are used for the transfer function because of the benefit explained in the earlier Section 3.2. Beside the scalar value, the gradient magnitude is used as second dimension and the second directional derivative along the gradient direction is used as third dimension. This enables advanced possibilities for the isolation of the material boundaries. Direct manipulation widgets are the solution of Kniss et al. [23] to close the conceptual gap between the transfer functions and the spatial domains. In contrast to the comprehensible spatial domain which represents the 3D space of the rendered volume, the domain of the transfer function is more abstract. Direct manipulation widgets provide a bidirectional feedback to link interaction of the two domains. The transfer function widget is embedded in the main rendering window.

There are two different scenarios which describe the use of these widgets. At first, the user can start to move and rotate a clipping plane through the volume to get a good impression of the slices. If a certain area of interest is discovered he can click on the clipping plane. This action initiates a visual feedback in the domain of the transfer function which shows the data value and the corresponding derivative. Through mouse movements around the clipping plane the user is able to examine the changes in the transfer function domain and he can assign opacities to the regions of interest. In the volume rendering, all the voxels with similar transfer function values are displayed. This *painting* process of the transfer function can be repeated until all features of interest are discovered and visualized. In another scenario, the data set is initially visualized with a default or generated transfer function to display some possible features of interest. Then, the user can use the widgets to act like in the previous described scenario to work out a visualization of all the features he is interested in.

In addition to the transfer function widget and the clipping plane widget, a data probe widget, a classification widget, a shading widget and a color picker widget are provided to allow a *dual-domain interaction*. Figure 3.5 shows the use of direct manipulation widgets. In (A) the bone surface is emphasized with the use of a triangular classification widget. With the clipping plane a look at one slice inside the skull is made possible and the skin is captured with the data probe widget. The soft tissue is visualized with the application of the data probe widget (B) and the transfer function is shown as result of the dual domain interaction (C) [23].

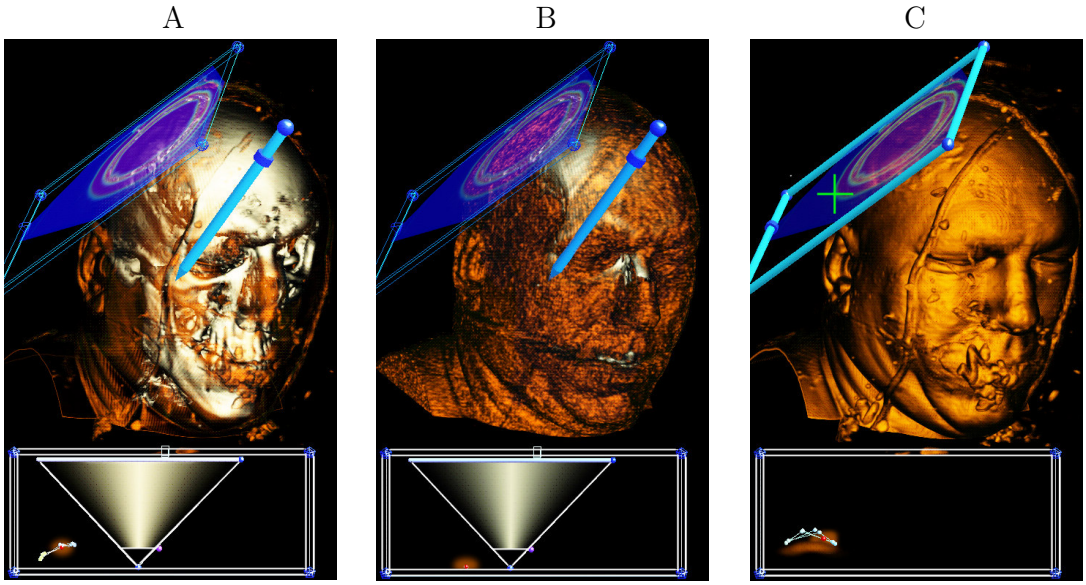

Figure 3.5: The usage of direct manipulation widget to emphasize bone (A) and skin (C). In (B), the data probe widget is shown in soft tissue [23].

3.3.2 Image-Driven Techniques

As a meaningful image is the most important result in the process of volume rendering, image-driven techniques for transfer function design are based on the analysis of image-inherent information. The analyzed images are generated with varying parameter settings. One method for setting visual parameters is an interactive exploration of the parameter space. Alternatively, a search for the good parameters can be performed based on an objective quality measure. A benefit of image-driven approaches is that even unexperienced users can generate pretty good images with the drawback that the results are often hardly reproducible. The reasons therefore are non-deterministic concepts used for algorithms which are responsible for the image generation.

Two interesting image-driven approaches are described in more detail. At first, the concept of He et al. [6] for semi-automatic transfer function generation based on stochastic search algorithms is presented. Second, the approach of *Design Galleries* introduced by Marks et al. [29] as a general concept of setting visual parameter in computer graphics is described.

Stochastic Search Algorithms

He et al. [6] introduced an approach for the assisted exploration of transfer functions. Stochastic search techniques are used to solve a parameter optimization problem which describes the search for a transfer function. A random or predefined set of transfer functions serves as initial population. A process of evolution of stochastic algorithms works on this initial population which involves the user to get meaningful

results.

In Figure 3.6 the visualization process is shown with the usage of semi-automatic transfer function generation. Based on a certain data set, the initial population of transfer functions is generated which leads to a first set of images. From this point in the visualization process there are two different approaches to go on. Either the user can evaluate the first population of images to initiate a new generation of transfer functions (A) or the user can define objective goals for the generation of new parameters (B). These goals can include entropy or histogram variance of the resulting image. Both processes are repeated until a satisfactory result is achieved. Stochastic

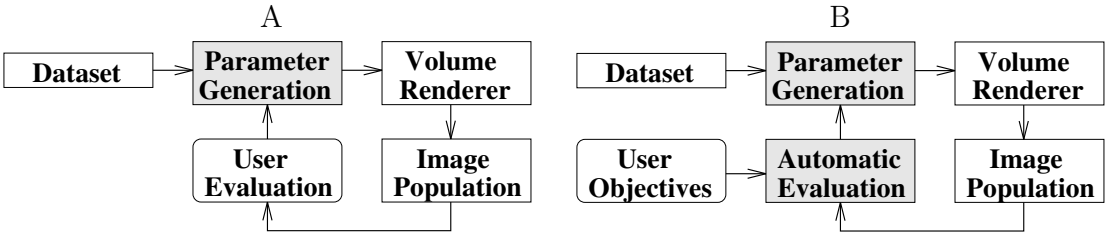

Figure 3.6: The visualization process with different options for user interaction to generate the final image [6].

search techniques are used for a variety of problems and they most likely produce an optimal global solution for a set of locally optimal solutions. These methods are applied to various intermediate images which are partially satisfying. The intuition is that these images have a positive influence on the results of the next iteration. For the implementation of stochastic search techniques an encoding for the solutions of the optimization problem is needed. This encoding is basically used for the mapping of structures $x(i)$ onto solutions. The population $P(t)$ where t represents a certain time step is defined as an s-dimensional vector of structures $x(i)$,

$$P(t) = \langle x_1(t), x_2(t), ..., x_s(t) \rangle.$$

In this equation, s is the size of the population and $P(0)$ as the initial population can be chosen randomly or with heuristic methods.

As the optimization techniques are based on evolutionary models the term *genotype* is used for the set of transfer functions. An algorithm for direct volume rendering translates the transfer functions into images which are regarded as *phenotypes*. In a user-controlled or automated process a fitness value is assigned to the set of phenotypes. Different stochastic search algorithms produce different results concerning the selection of an intermediate population and the generation of the solutions. He et al. [6] applied their algorithm on a transfer function which maps a single scalar value to a single optical property but their approach can be extended to multiple dimensions. The normalizing of the domain and range of the transfer function leads to the simple function

$$f : D \rightarrow V,\ D \in [0,1],\ V \in [0,1]$$

which defines the transfer function. To generate the initial population of transfer functions, a library of common initial functions as shown in Figure 3.7 is provided. These functions can be manual adjusted by the user. After the generation of the

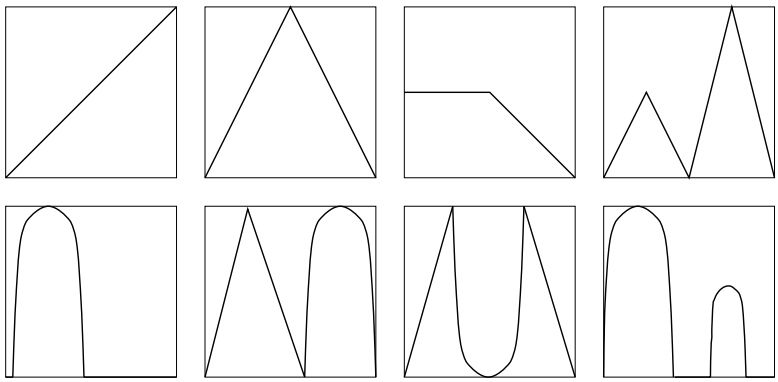

Figure 3.7: Provided example transfer functions for the generation of an initial population [6].

intermediate population and new solutions via mutation processes, the solutions can be evaluated. All transfer functions have an initial fitness value of 0. Depending on the selection of good images by the user, the fitness value 1 is assigned to the functions and a new intermediate population is set up.

The benefit of the application of stochastic search techniques is the hiding of the complexity from the user. An unexperienced user does not have to handle the abstract parameter space of transfer functions. Instead, he can just appoint the good images and initiate a new process cycle to get new results of a further evolution process [6].

Design Galleries

The approach of *Design Galleries* introduced by Marks et al. [29] provides an automatically generated and organized broadest selection of different graphics or animations. This selection is generated with a given varying input-parameter vector. The challenging task is the finding of a set of input-parameter vectors for a good variety of disperse output-value vectors. Furthermore, it is very important to present the output in a way that allows the user an intuitive browsing of the results. Beside solving several computer graphics problems like light selection and placement, *Design Galleries* are well suited for opacity and color transfer function specification.

A list of parameters forms the input vector which controls the generation of the output vector. The output vector holds the values that define the properties of the output graphic. A distance metric is used as an indicator for similarities within the set of output graphics. To ensure that the output vectors and the resulting output graphics are well-distributed, a dispersion method is used. Finally, an arrangement method is applied for presenting the graphics to the user. To keep the complexity away from the user, the creator of a *Design Gallery* system defines input vector,

output vector and the distance metric for a certain field of application. The user just has to select the best graphics from the gallery.

In their examples on using *Design Galleries* for opacity and color transfer functions, Marks et al. [29] developed interfaces for a CT data set of the human pelvis and the simulated electron density of a protein. At first, the input vector has to be defined. They parameterized the opacity transfer function for the protein data set which contains values in the interval [0, 255] by a low-pass filtered polyline with eight control points. Those are 16 values in total because of the x- and y-position of every control point. Five values are chosen for the color transfer function. The six resulting color ranges are yellow, green, cyan, blue and magenta. This configuration is shown in Figure 3.8. For the output vector eight pixel represented by 24 YUV values are

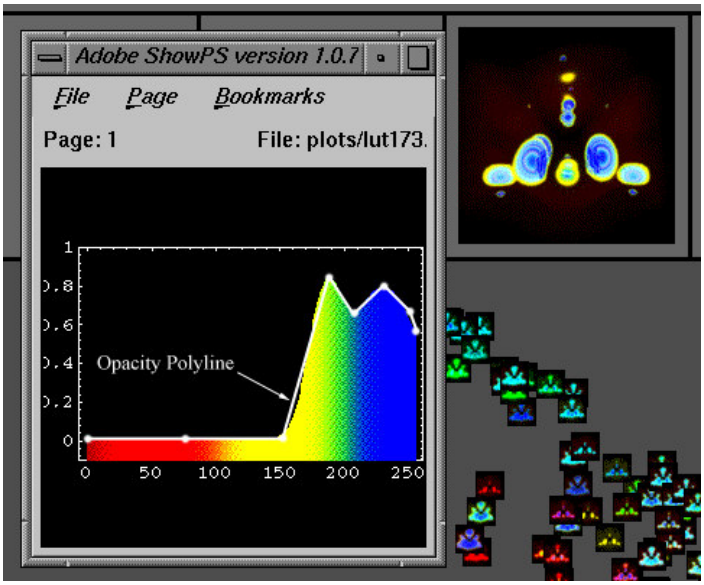

Figure 3.8: A pop-up window shows the graphical representation of the opacity and color transfer functions [29].

selected manually for a certain data set. Those eight pixels are a solid base for a set of images with an excellent dispersion. The standard euclidian distance is used as output-space metric. For an optimal dispersion of the output vectors an evolutionary strategy is applied. A randomly generated set of input vectors is perturbed to replace the existing vectors.

The arrangement of the resulting images is presented within an easy-to-user interface as shown in Figure 3.9. In the center of the panel are numerous thumbnails. These thumbnails as volume rendered images with a low resolution are representations of the output vectors. There is a correlation between the distance of the thumbnails and the distance between their output vectors. To give the user a better impression of the thumbnails a zoom of the center panel is implemented. The surrounding images are the full-size representations of selected images. Images of the surrounding gallery can be selected, too. This leads to the display of the color and the opacity

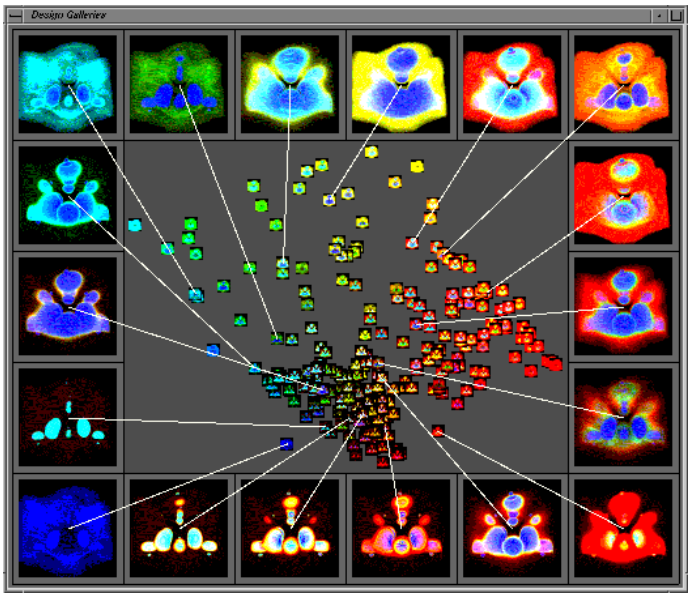

Figure 3.9: In the center panels are the graphical representations of the output vectors which are small renderings with different color and opacity transfer functions. The selected thumbnails can be placed in the surrounding image gallery by the user [29].

transfer functions as shown in Figure 3.8 to visualize the relation between image and data [29].

3.3.3 Data-Driven Techniques

The presented image-driven techniques are based on the analysis of rendered images. As a considerable amount of data has to be processed to render various opacity and color transfer functions, the focus of interest of the data-driven techniques is the original volume data set. Image-related parameters like pixel-resolution and viewing position have no influence on transfer function design anymore.

At the *Visualization 2000* conference, there was a panel discussion where four promising approaches of transfer function design have been compared. A data-driven technique presented by Kindlmann and Durkin [21] won this contest known as *the transfer function bake-off* [35]. Their approach bases on the first and second order directional derivatives of the scalar field and is described in the following section. Following, a technique for the visualization of aneurysms from medical image data with multidimensional transfer functions introduced by Vega et al. [8, 9] is presented.

Semi-Automatic Generation

Kindlmann and Durkin [21] assume that the boundary regions between relatively homogenous materials are the areas of interest in the scalar volume. As their goal is the visualization of material boundaries, they mention the problem of the band-limitation of the measurement process. The boundaries of real world data are blurred

by a gaussian. Gradient vectors have the property that they tend to point in the direction of the material boundaries. An examination of the scalar field (f) and its derivatives is used for the creation of an opacity function. This is done by following a path inside the volume along the direction of the gradient.

In computer graphics common edge detectors use the first (f') and the second directional derivative (f'') along the gradient direction to find edges. These techniques work in the spatial domain to locate the edges. To apply opacity functions to the volume boundaries, these regions have to be found with the help of a function of the data value. The relationship of f, f' and f'' is used to locate the edges within this domain.

In Figure 3.10 (A) f and f' are plotted as a three-dimensional curve. Its projections show the plot of data value versus position, first derivative versus position and data value in relation to first derivative. The last of these projections has no information about the position anymore. Figure 3.10 (B) shows the same projections for data value and its second derivative. A three-dimensional plot of the relation-

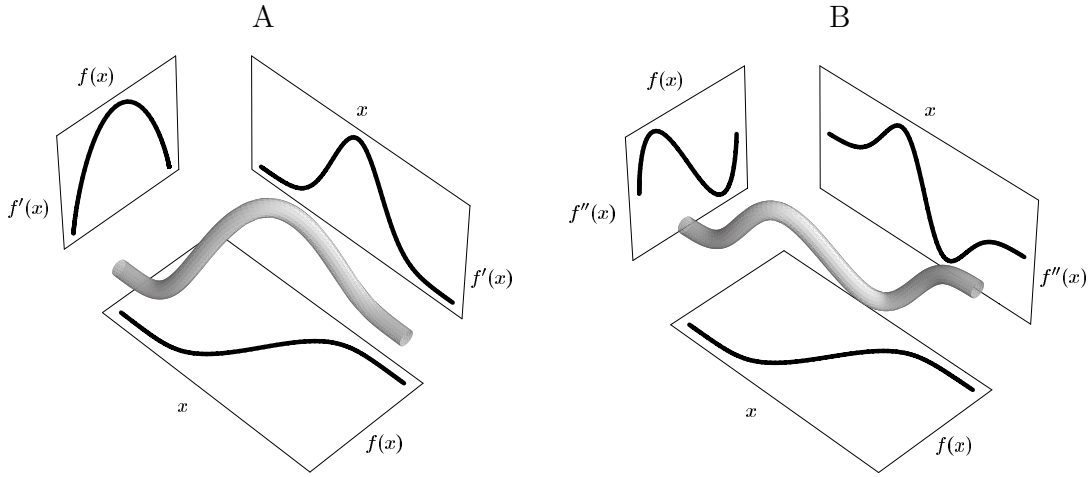

Figure 3.10: The relationship between f, f' and position x (A) and between f, f'' and position x (B) [21].

ship between f, f' and f'' where the position information is eliminated as shown in Figure 3.11 serves as an indicator for boundaries in the volume. The plotted curves are used for the generation of an opacity function. If a curve is similar to the one presented in Figure 3.11, it will be interpreted as a sign for a boundary in the volume. The specific features of the curves have to be analyzed by a detection tool to assign the highest opacity values to the center of the boundaries and to generate a rendering which shows the detected boundaries.

A three-dimensional histogram is generated with the three quantities f, f' and f'' as axes. This histogram is used to measure how the data values and its derivatives are related to each other. As soon as the histogram is generated it is possible to visualize the histogram volume to see the boundaries of the object. For volume rendering

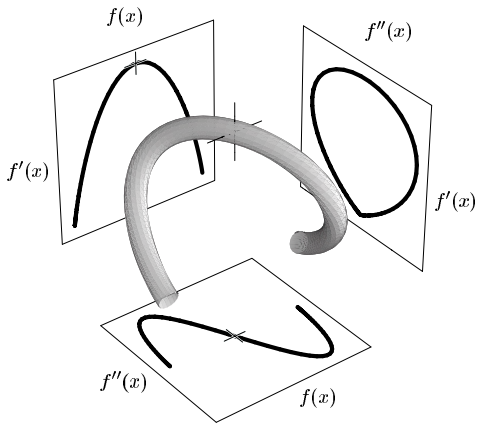

Figure 3.11: The relationship of f, f' and f'' where the position information is eliminated [21].

the histogram volume is not the best choice because of its noisiness. To achieve an improved rendering, scatterplots of f' versus f or f'' versus f are produced. This is done with a *summed-voxel* projection of the histogram volume where the projection can be performed along one of the two axis f' or f''. In Figure 3.12 the slice of a

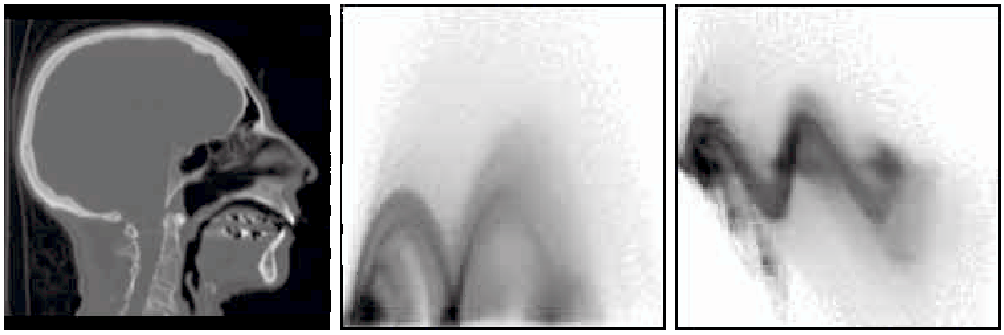

Figure 3.12: Slice of a head data set (left), f' versus f (middle) and f'' versus f (right) [21].

CT data set of a human head is shown with the relationship between f' and f and between f'' and f. Each displayed curve is a representation of a boundary between two materials. A *position function* $p(v)$ is used to compute the opacity transfer function,

$$p(v) = \frac{-\sigma^2 h(v)}{max(g(v) - g_{thresh}, 0)}$$

where the parameter σ controls the amount of boundary blurring and $g(v)$ and $h(v)$ represent the average of the first and second directional derivatives at value v. The functions $g(v)$ and $h(v)$ result from slices at value v of the histogram volume. As the gradient magnitude inside the materials has not very often the exact value of zero, a small threshold g_{thresh} is applied to determine zero-crossings of $p(v)$. A *boundary*

emphasis function b is needed for the calculation of the final opacity function $\alpha(v)$ with the equation

$$\alpha(v) = b(p(v)).$$

The shape of the boundaries is controlled by the boundary emphasis function. This function allows the user to define the appearance of the boundaries and the object interiors.

This algorithm reduces the parameter space of opacity transfer functions significantly. Instead of the exploration of all possible opacity transfer functions which is done with the image-driven techniques, only the opacity transfer functions which lead to the display of object boundaries have to be regarded. This approach offers a good possibility to find transfer functions that generate meaningful images for given data sets where the boundaries between materials are the interesting regions. No a priori knowledge about the spatial structures of the data set is necessary to achieve this goal [21].

Visualization of Aneurysms

An approach for the 3D visualization of intracranial aneurysms with multidimensional transfer functions is introduced by Vega et al. [8]. CT-angiography (CTA) data sets are the basis for the visualization of vascular structures. Common 1D transfer functions allow the mapping of measured scalar values to opacity and color values. Problems occur if the vessels are too close to the skull base because the differences of the data values of the vessels, filled with contrast agent, and the bone structures are not big enough. As seen in the previous sections, multidimensional transfer functions can help to get a good visualization of boundaries. Beside the introduction of multidimensional transfer functions to improve the described clinical problem, Vega et al. proposed tools for the manipulation of multidimensional transfer functions.

In addition to the data value, they use a border function based on first and second directional derivatives as introduced by Kindlmann and Durking [21] and the gradient magnitude. A good presentation of the data properties allows the user an easy identification of the interesting regions within the volume. The data value and the derivatives are the axes of a 3D histogram as shown in Figure 3.13 (A). Information about materials and the boundaries between them are contained in this structure. Figure 3.13 (B) shows a 2D scatter plot of the histogram, where the x-axis represents the data values and the gradient magnitudes are represented by the y-axis. This plot provides a meaningful area where the user can define a 2D transfer function. Individual colors and opacities can be applied for each material and the weight of the single materials can be controlled.

Geometric objects modeled with quadratic Bézier curves as seen in Figure 3.13 (B) are the designed widgets to paint the transfer function. Each control point can be moved by the user inside the painting area. In Figure 3.14 an image which is rendered with a common 1D transfer function (left) is compared to a result rendered with Vega's multidimensional transfer functions (right). As Vega's approach allows an exact separation of two tissues, different colors can be assigned to the skull bone

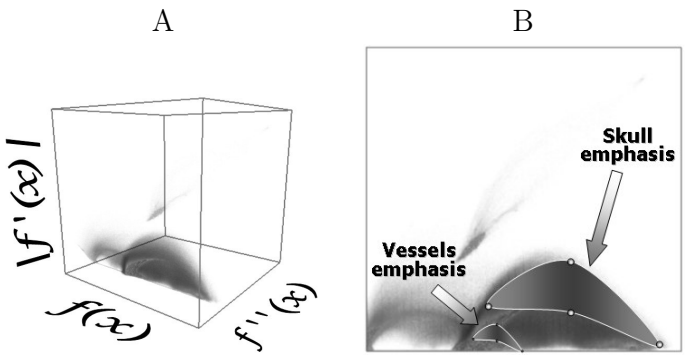

Figure 3.13: A 3D histogram generated from a CTA data set (A). The 2D scatter plot of the histogram with two widgets designed for an easy painting of the transfer function (B) [8].

and the vessels. One-dimensional transfer functions cannot offer these possibilities because the scalar values of the two materials are to close to each other.

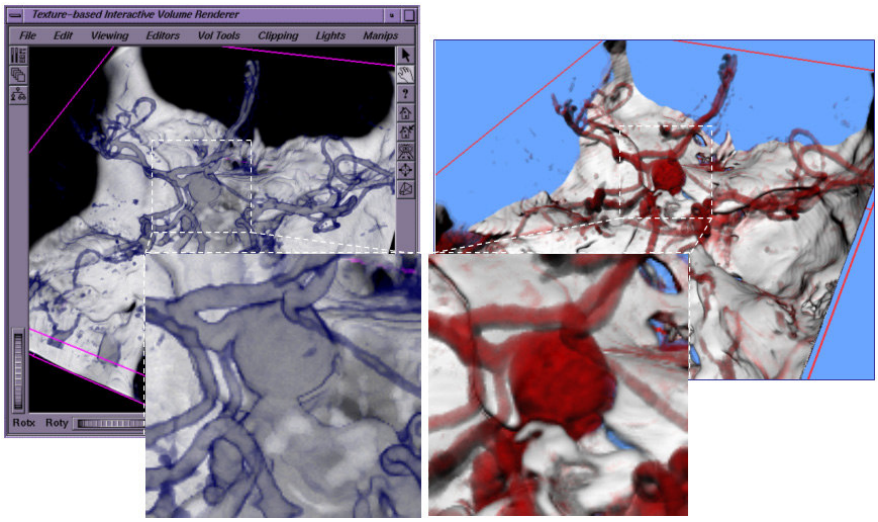

Figure 3.14: An image rendered with a common 1D transfer function (left) compared to an image rendered with Vega's multidimensional transfer functions (right) [8].

In further researches Vega et al. [9] presented a method for the automatic adjustment of bi-dimensional transfer functions for the visualization of intracranial aneurysms. Transfer function templates based on the information of a 2D histogram are used to achieve a clear visualization of the vessels. These templates are adjusted automatically and produce results which are comparable to manual created transfer functions [8, 9].

Chapter 4

Transfer Function Editor

To allow an intuitive and user-friendly setup of the transfer function a 1D transfer function editor was implemented. At first, the architecture of the editor is described. The graphics library *OpenInventor* is used for the implementation and the transfer function editor is realized with several OpenInventor nodes. These nodes are integrated in the *RadBuilder* environment. Advanced networks can be build with RadBuilder to realize medical applications. In the second part of this chapter, the most important features of the transfer function editor are described. The used geometric primitives for the transfer function setup are introduced and different options for the color assignment are presented.

4.1 Architecture

This section presents the framework for the realization of the transfer function editor. For its implementation the OpenInventor library is used. The transfer function editor is realized as a network of nodes which is integrated in the Siemens RadBuilder environment. Following, the framework and the most important implemented nodes are described. Furthermore, it is shown how the nodes can be connected to build advanced networks.

4.1.1 Framework

The transfer function editor is integrated in the RadBuilder of SCR (Siemens Corporate Research). This platform for application development is based on OpenInventor nodes. OpenInventor itself is based on *OpenGL*. Because of these connections there will be a brief description of OpenGL first. Followed by an introduction of the possibilities of OpenInventor and RadBuilder. The chapter closes with a description of the user interface development for RadBuilder which is based on *HTML*, *JavaScript* and *CSS*.

OpenGL

OpenGL was introduced in 1992 for the development of 2D and 3D graphics applications and became the number one graphics application programming interface (API) for this purpose. It offers numerous functions for rendering, texture mapping, special effects and visualization. As a main benefit, OpenGL is supported by all popular platforms for desktop and workstation. Language bindings are available for a lot of popular programming languages such as *C, C++, Fortran, Ada, Python, Perl* and *Java*. Because of its high performance and visual quality, OpenGL often serves as basic API for creating 2D and 3D graphics within the field of medical imaging. Main advantages of using OpenGL are that it is:

- an open, vendor-neutral, multiplatform graphics standard.
- stable because implementations are available since several years on a wide variety of platforms.
- evolving and allows a fast integration of new hardware innovations via the OpenGL extension mechanism.
- portable and produces a consistent visual result on different hardware and operating systems.
- scalable and the applications can either run on consumer electronics and personal computers or on workstations and supercomputers.
- well structured and offers an intuitive set of logical commands.
- well-documented and a lot of sample code is available.

Because high-level commands for describing three-dimensional objects are not provided by OpenGL, the models have to be created with a small set of geometric primitives like points, lines and polygons.

OpenGL works as a state machine. A state or mode is active until it is changed. For example, if the current color is set to the state *yellow*, every following object is drawn with this color until the state is set to another color. More OpenGL state variables are other material properties, current viewing and projection transformations, positions and characteristics of lights, polygon drawing modes, line and polygon stipple patterns, as well as pixel-packing conventions [33].

Open Inventor

As mentioned, OpenGL does not provide the high-level commands to build three-dimensional objects. OpenInventor is a library built on top of OpenGL to provide these features. It is a window system-independent library which offers a higher-level and object-oriented approach to create interactive 3D graphics applications. A set of building blocks reduces the programming effort while it taps the full potential of powerful graphics hardware features. Its library of objects can be either solely used,

or modified and extended. Examples for Inventor objects are shape, property, group, database primitives, engine objects, interactive draggers and manipulators, as well as several editors and viewers. Through providing a 3D interchange file format, 3D scene objects can be shared by users and a variety of programs.

To render an image, all information of a 3D object like shape, size, color, texture and position are stored in a scene database. While other 3D graphics packages offer the possibilities to create photorealistic images, a main benefit of OpenInventor is that it provides tools to add interactive elements to a scene. These can be used for example to change the viewpoint and to resize or move the 3D objects. Furthermore, the adding of new objects into an existing scene is easy to realize. In OpenInventor a scene database is set up with a number of nodes which represent shapes, properties or groupings. A sample scene graph as an ordered collection of nodes is shown in Figure 4.1 which is based on an example by Wernecke [44]. OpenInventor elements

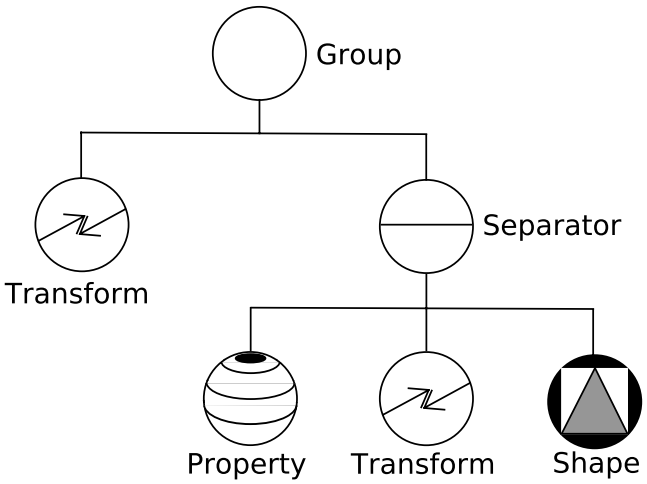

Figure 4.1: Example of a simple scene graph.

which were especially important for the realization of the transfer function editor are *fields*, *nodekits*, *draggers* and *sensors*.

A node contains a number of fields to hold its parameters. For example, the shape node *SoCube* has fields for its width, height and depth. The type of these fields is *SoSFFloat*, which means that each is able to store a single value of the type *float*. OpenInventor also provides *multifields* which can store more than one value of a certain type. Fields of different nodes can be connected through *field-connections* with the result that a change of the one field leads to a change of the connected field as well.

Nodekits are structured collections of nodes. They are working like templates and offer the possibility to add certain nodes if they are needed at a specified position. A good example of an OpenInventor built-in nodekit is the *SoShapeKit*. It offers a so-called *nodekit catalog* where all its parts are available. The programmer can choose the entries he wants to use and set their attributes. A cube is the default shape of the *SoShapeKit* and properties such as material and geometric transformations are

provided. These properties have a defined position in the nodekit and can be used and changed if required.

A nodekit is an efficient organization of nodes because one of its nodes is only created if it is needed. It is even possible that one nodekit contains other nodekits to build very flexible, complex and powerful structures. The *SoShapeKit* is an example of a nodekit provided by OpenInventor. It is possible to design own nodekits through subclassing. This mechanism was used consequently to build the transfer function editor.

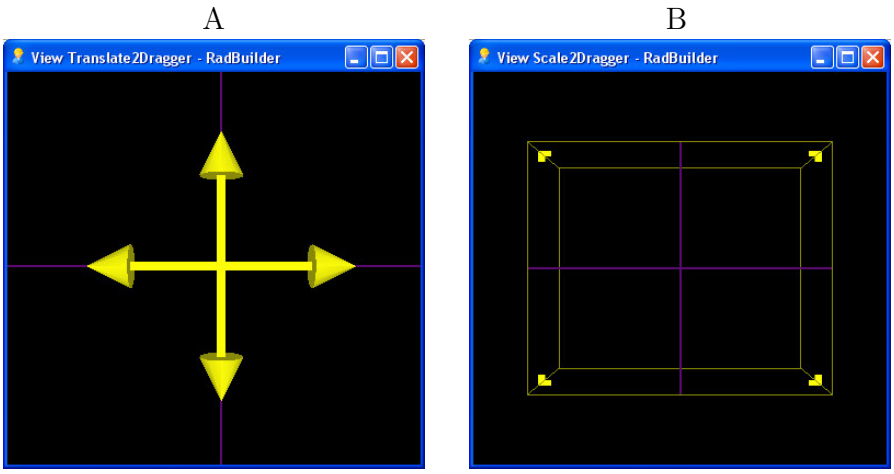

Figure 4.2: Two built-in draggers of OpenInventor. The *SoTranslate2Dragger* (A) can be picked and moved in x- and y-direction and the *SoScale2Dragger* can be scaled in the two dimensions.

Another type of objects which was necessary to realize the transfer function editor are the draggers. A dragger is a node in the scenegraph which provides interactions with the objects. They are configured with a built-in user interface. This geometric representation is used for picking actions and user feedback. There are various draggers for different purposes but all of them have some translation-, scaling- or rotation functionality. Some of them even combine these features.

Simple draggers for two-dimensional translation and scaling are shown in Figure 4.2. They are subclasses of *SoDragger* and they can be used in three different ways. One option is the connection of a dragger with fields or engines in the scene graph to establish dependencies. Another common case for the use draggers is their combination with *callback functions* which perform user-defined reactions on dragger-changes. Finally, simple draggers can be combined to build more complex draggers.

As explained, draggers are useful for user-interactions. They are intensively used within the graphical representation of the transfer function editor and they are applied to move primitives (especially trapezoids and their parts), to place colors and for zooming and scrolling purposes. Because a subclass of *SoDragger* is also a nodekit its catalog entries can be manipulated. For the draggers in the transfer function editor, the shapes of the draggers are changed to provide a user-friendly representation.

In OpenInventor sensors are used to watch out for certain events. If a specified event like the change of a field's value happens, user-defined callback functions which are connected to the sensor will be performed [44].

RadBuilder

The RadBuilder is a platform for the fast creation of medical applications developed at Siemens Corporate Research in Princeton, NJ (USA). By creating networks within a visual programming environment, applications can be built in an intuitive way. A large number of modules of visualization graphs and processing pipelines, as well as interfaces for external toolkits such as the *Insight Segmentation and Registration Toolkit* (ITK) and the *Imaging and Visualization Toolkit* (IVT) are provided.

Areas of development which are supported by RadBuilder are C++ modules for processing and visualization purposes, user interfaces in HTML and scene graphs, as well as pipelines to connect modules. Figure 4.3 shows the RadBuilder realization of the simple scenegraph in Figure 4.1 with the resulting rendering. All RadBuilder

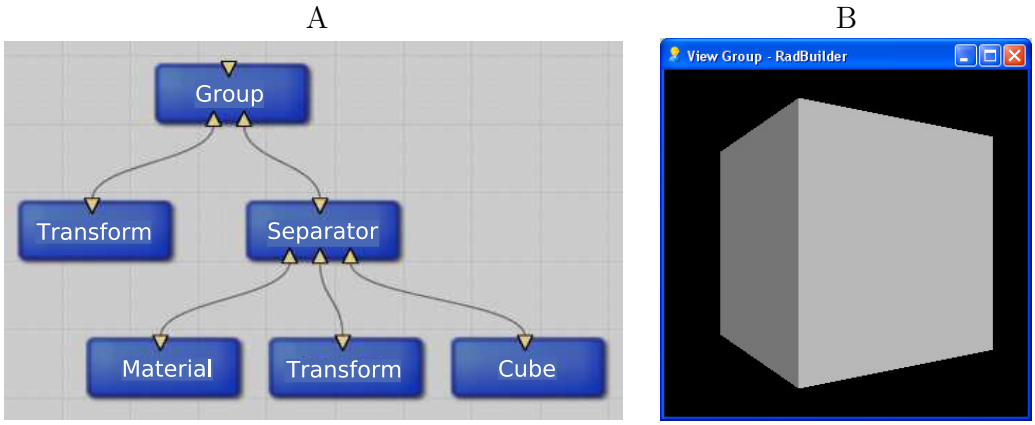

Figure 4.3: RadBuilder realization of a simple scene graph (A) and the resulting rendering (B).

modules represent either a rendering node or a processing engine that is based on the OpenInventor toolkit. While the rendering nodes are traversed top down from left to right to perform their actions, the engine updates its output fields if its input is changed. Nodes and engines can be combined in various ways. Field connections between modules can be created graphically so that field values are propagated to other fields.

Each module provides an automatic value-display of its fields and nodes. These panels can be used to change field-values of the nodes and to set up connections. More user-friendly interfaces can be built with HTML documents. With the help of a scripting language like JavaScript, it is possible to create objects like sliders, checkboxes or comboboxes. This offers a convenient way to display only the attributes which are interesting for the user of the application [30].

4.1.2 Network

A RadBuilder project consists of a network of single nodes which are connected to each other. This section describes the various parts of the transfer function editor and their connections. A central element for the interaction of the different nodes is a parametric description of the transfer function editor.

Nodes

As already mentioned OpenInventor is based on a scenegraph. A node is the fundamental element of the graph. It contains data and methods that define shape, property or grouping [44]. Each node is composed of a set of data elements (fields) which describes the parameters of the node. The transfer function editor is built with nodes and nodekits. In the following paragraphs the different parts of the transfer function editor are described.

SoRadTransferEditor and SoRadTFEEditorGUI: The *SoRadTransferEditor* node is the control unit of all public settings and of the used primitives (e.g. trapezoids) to set up a transfer function. This node provides the functionality to create and to remove trapezoids. In addition, the primitive's properties can be edited. *SoRadTransferEditor* contains fields to set up global options like the display of units such as Hounsfield Units or scalar values. All the functions the user may interact with for the definition of the transfer function are provided by this node. To simplify the user interface which is generated automatically by RADBuilder, the *SoRadTFEEditorGUI* node is a well-structured and clearly arranged user interface. The fields of the GUI node have to be connected to the corresponding fields of the *SoRadTransferEditor*. Furthermore, the user has the option to create *colorbars* or to save and load transfer functions.

SoRadTFEParams1D: The *SoRadTFEParams1D* node represents a parametric description of the transfer function and its settings. This node describes a complete transfer function with only a few parameters. The parametric description can be used for the distribution of certain tasks within a client/server-architecture. One important field of *SoRadTransferEditor* is called *params*. This field contains a *SoRadTFEParams1D* node which stores the state of the transfer function. For example, it keeps the parametric values of a *global colorbar*.

SoRadLUT1D: *SoRadLUT1D* has access to the parametric description to set up a color array and an alpha array. Both fields have to be connected to a node which generates a one-dimensional look-up-table based on these arrays. *SoRadLUT1D* has fields to control the render quality by using a *SoComplexity* node.

Connections between Nodes

After the *SoRadTransferEditor* and the *SoRadLUT1D* node are added to the scene, they have to be connected as shown in Figure 4.4. The RadBuilder application pro-

vides a *SoRadLut* node which is part of the volume renderer network. This node generates the texture for the volume renderer. If the input mode is set to *ARRAY*, the look-up-table will be based on the values of the *arrayColor* and *arrayAlpha* field. This two fields of *SoRadLUT1D* have to be connected to the corresponding fields in *SoRadLut*. In addition, the *bitsUsed* fields have to be connected to each other to guarantee the proper array lengths in *SoRadLUT1D*. The *SoRadLUT1D* node needs access to the parametric description to extract color and opacity information for its computations. To enable this access the *params* field of *SoRadTransferEditor* has to be connected to the *tfe_Params1D_Node* field of *SoRadLUT1D*.

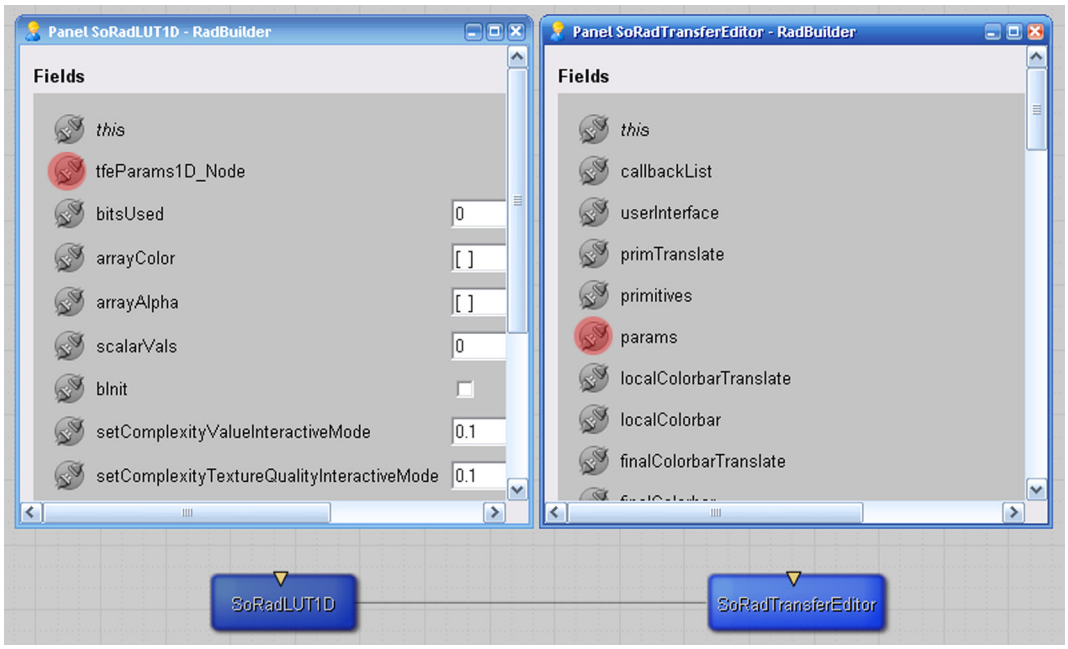

Figure 4.4: Automatically generated panels of *SoRadLUT1D* and *SoRadTransferEditor*. The red marked fields have to be connected to each other. The *params* field contains the parametric description of the transfer function editor.

Advanced Network

The basic network can be extended to a more user-friendly application as shown in Figure 4.5. The use of an orthographic camera and a *SoRadViewportGroup* enable the display of the transfer function in a desired layout. In addition, the *SoRadExaminer* node can be used for a three-dimensional view of the scene. The *SoRadTFEEditorGUI* node offers a well-structured user interface. To use this additional user interface, every field of the *SoRadTFEEditorGUI* node has to be connected to the corresponding field of the *SoRadTransferEditor* node. Furthermore, a *SoRadKeyRemoteExaminer* node provides zoom and rotation functionality for the rendering window and the transfer function editor window. The quality of the rendered volume image can be controlled by a complexity node. A switch node allows the rendering of the volume in a lower quality.

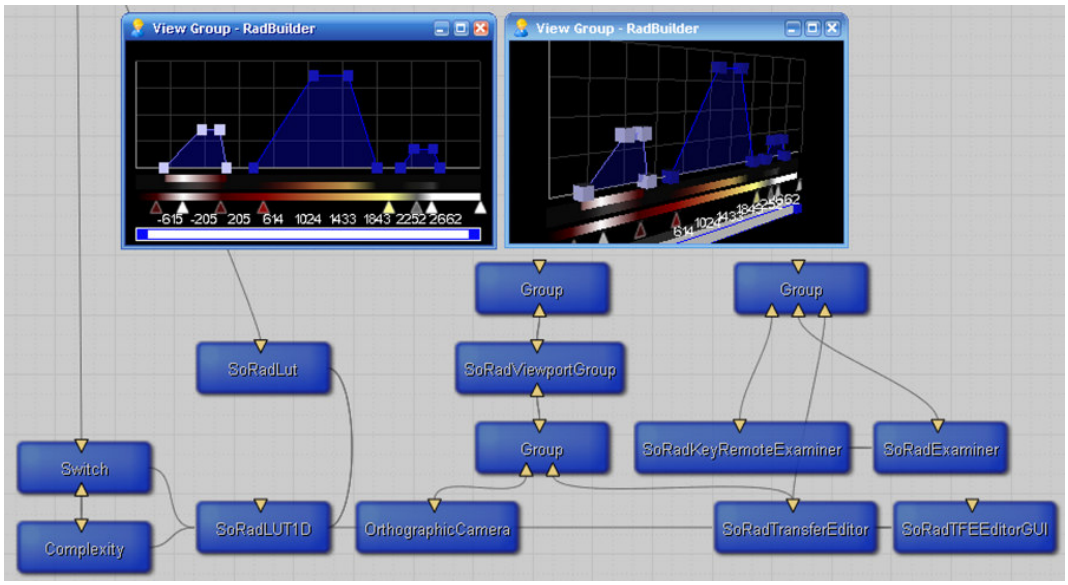

Figure 4.5: An advanced network with the transfer function editor in a 2D-view and a 3D-view.

Workflow

The panel of *SoRadTFEEditorGUI* and the automatically generated panel of *SoRadTransferEditor* provide the functions to add or remove primitives to the scene for the transfer function setup. These user interface panels are also used to modify the properties of the primitives and to adjust the colors. All modifications are stored in the parametric description of *SoRadTFEParams1D*. Because of a field connection, the *SoRadLUT1D* node has direct access to the parametric description and is able to calculate the proper values for the look-up-table which is passed to the *SoRadLut* node.

4.2 Features

In this section, the used geometric primitives for the transfer function setup and the parametric description are presented. It is also described how the primitives can be moved and adjusted and what restrictions are implemented. Following, there is a closer look at various color options. Besides the usage of a global colorbar it is shown how the colors can be defined for the single primitives. This section closes with the presentation of some global options.

4.2.1 Primitives

The transfer function setup can be facilitated with various primitive objects such as linear ramps, trapezoids, splines or gaussian curves. These objects can be adjusted manually by the user. The underlying rendering algorithm has to provide interactive

frame rates. An immediate visual feedback is very important to allow goal-directed work. As the motivation for the development of the 1D transfer function editor was the simplification of the transfer function setup, trapezoids and ramps are used. For the user, these primitives which are shown in Figure 4.6 are more convenient to adjust than splines or curves. The opacity is assigned in respect to the data value. Trapezoids are common peak shapes for the transfer function setup. Data values which are covered by the plateau of a trapezoid are mapped to full opacity. The range of values which is not covered is mapped to zero opacity and not displayed in the resulting image. A linear increasing or decreasing opacity value is assigned to the data values covered by the slopes on the left and the right of the trapezoid. If the plateau of the trapezoid is just a single point the primitive will looks like a

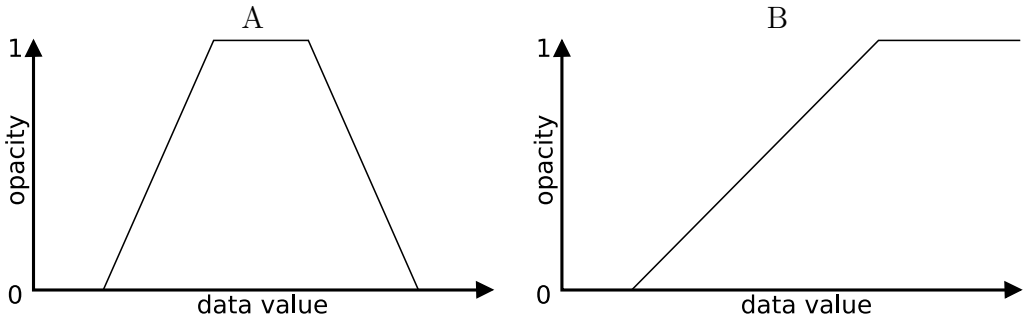

Figure 4.6: The provided primitives for the transfer function setup - a trapezoid (A) and a simple ramp (B).

tent. This shape can be used to display very narrow regions within the volume. A box shape, where the plateau of the trapezoid is as long as its base line usually leads to visual artifacts and should be avoided. Very simple primitives for the transfer function setup are ramps. The ramp offers limited degrees of freedom because only two of its edgepoints and its height can be adjusted.

Control

The user can add a number of trapezoids to define the transfer function. Depending on the zoom and the scroll position of the transfer function editor, the original width and position of the trapezoid are calculated. The trapezoid is originally placed centered into the visible area of the transfer function editor. To convert the trapezoid to a simple ramp the user has to choose *isSimple* at the trapezoid-properties. This feature leaves the two edgepoints on the left side of the trapezoid at its place and moves the right edgepoints to the very end of the range. To realize a user-friendly possibility to adjust the primitives, their edgepoints and their lines are built as OpenInventor draggers. Figure 4.7 shows how the elements of the primitives can be translated. Lines and edgepoints can be picked by the user and translated in the directions which are indicated by the blue arrows. The only limitation of a ramp is that two edgepoints are fixed to the right end of the range and cannot be moved in horizontal direction.

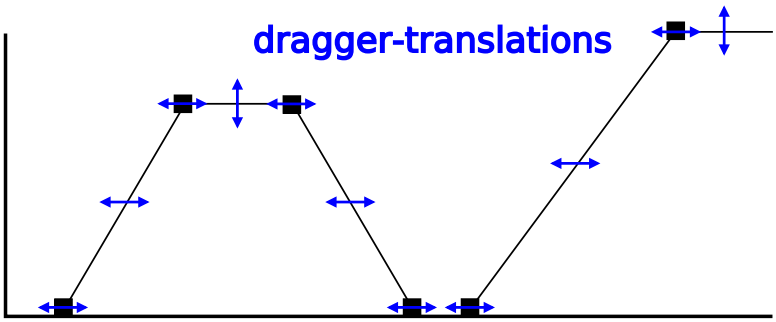

Figure 4.7: The lines and edgepoints are implemented as draggers. They can be translated in the directions indicated by the blue arrows.

Another property of a primitive determines whether it is active or not. Inactive primitives are not regarded for the lookup-table calculation. In addition, each added primitive can be removed if it is not needed anymore.

Restrictions

To avoid the generation of deformed trapezoids as shown in Figure 4.8 some restrictions of the dragger movement are implemented. The primitives can only be moved within a valid range. That means that their height cannot become smaller than zero or greater than one. This restriction is necessary because the opacity is also defined within this range. In addition, the primitives cannot pass the left or the right border of the data range. Another restriction is, that the lower edgepoints cannot enter the

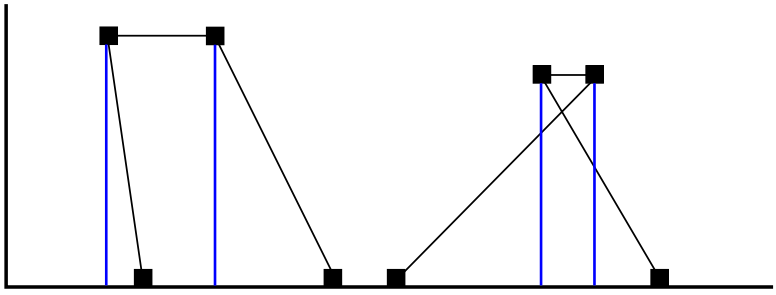

Figure 4.8: Deformed primitives which are avoided by implemented restrictions of dragger movements.

base-projection of the trapezoid's plateau as happened at the left primitive in Figure 4.8. Finally, the left upper edgepoint cannot pass the corresponding right one and vice versa as happened at the right primitive in Figure 4.8. In all the cases when a restriction is violated by dragger movements the last valid position is restored.

Parametric Description

The parametric description is a central part of the transfer function editor. In this element the complete transfer function is saved with a limited set of parameters. If

the transfer function setup, the lookup-table generation and the rendering process are realized within a client/server-architecture the parametric description can be sent over a network with very low costs. The most important nodes which are stored in the parametric description are of the types *SoRadTrapezoid2D* and *SoRadGlobalLut*. In the multifield with the *SoRadTrapezoid2D* nodes, all properties of the trapezoids are stored. These are for example the edgepoint-positions, the defined colors for the trapezoids, as well as information if the trapezoids are simple (a ramp) and active. The *SoRadGlobalLut* node contains the elements of the global lookup-table of the transfer function editor. These elements are a multifield with the colors of the lookup-table and a multifield with the corresponding x-positions. Filled with these entries, the field with the parametric description can be connected with all the nodes which need reading or writing access to trapezoid- or global lookup-table information.

4.2.2 Colors

The transfer function editor offers two different ways to assign colors to scalar values. At first, colors can be applied to the primitives. Second, the user can load a saved color table. To visualize the current state of the colors and to support the user with helpful information about primitives and their influence to the look-up-table, the transfer function editor provides up to three colorbars.

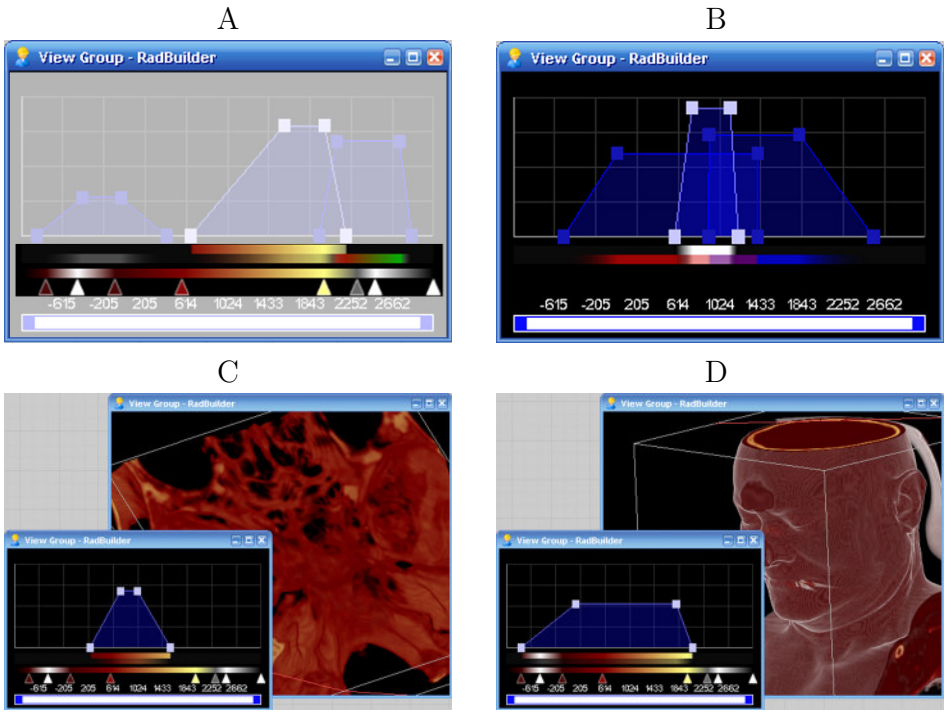

Figure 4.9: The transfer function editor provides up to three colorbars.

Colorbars

Figure 4.9 (A) shows the first colorbar which is positioned directly below the selected primitive. The colors are either individually assigned to the primitive or based on the colors in a global colorbar. The second colorbar represents the colors of the look-up-table which will be applied to the volume. Overlapping primitives influence the colors in this colorbar depending on the primitive's height at the certain scalar value as shown in Figure 4.9 (B). The third colorbar represents the global colorbar. The colors of this colorbar can be used by the primitives as illustrated in Figures 4.9 (C,D).

Primitive Colors

The multifield *currentTrapezoidColors* keeps the colors of the active primitive. For a trapezoid six different colors can be defined as shown in Figure 4.10 (A,B). In

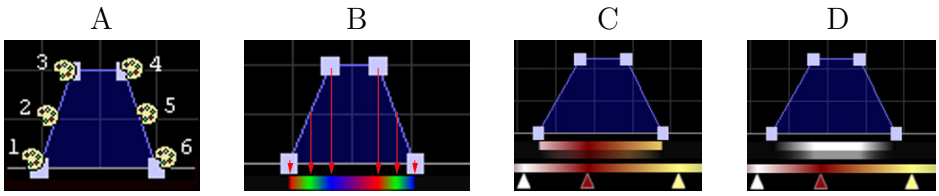

Figure 4.10: Colors can be defined at six places of the trapezoid (A,B). A trapezoid can adopt the colors of the global colorbar (C) or it can use its own colors (D).

this case the *currentTrapezoidColors* field has six entries. These values refer to the four edgepoints of the trapezoid and the midpoints of the trapezoid's slopes. The

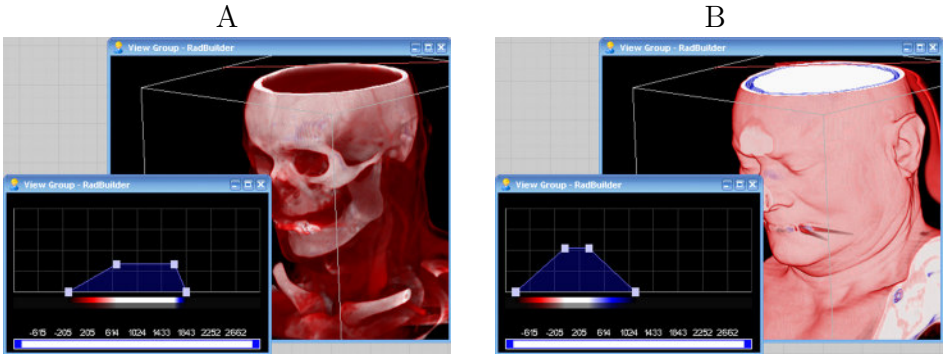

Figure 4.11: The trapezoid is moved to a different position while its colors remain the same.

first colorbar below the trapezoid visualizes these colors. The colors are interpolated between two defined colors. For example, if the colors of the upper edgepoints of the trapezoid are blue and red, then the colorbar below the trapezoid displays the transition of these colors as displayed in Figure 4.10 (B). The possibilities which are

offered by the definition of trapezoid colors is shown in Figure 4.11. A trapezoid can be moved while it keeps its colors.

Colorbar Colors

Figure 4.10 shows a trapezoid which adopts the color of the global colorbar (C) and another one which uses its own colors (D). The global colorbar can be filled with user-defined colors. This process is controlled by the fields *globalLUTColors* and *globalLUTPositions* of the *SoRadTransferEditor* node. Each color which is part of this colorbar needs an appropriate position where the color is placed. The position of a color can be adjusted by moving the triangle-shaped draggers which are positioned below this colorbar. In addition, the user can decide if the alpha values (opacity) of the primitives should be regarded for the display in the second colorbar and if the third (global) colorbar should be displayed separately.

4.2.3 Global Options

The *SoRadTFEEditorGUI* node offers a couple of options in order to control global properties. The user interface of the transfer function editor provides a zoom- and scrollbar which is located at the bottom of the transfer function editor window. In order to adjust a primitive very precisely, the user can use an up to 21-fold zoom into the panel of the transfer function editor as shown in Figure 4.12. It is also

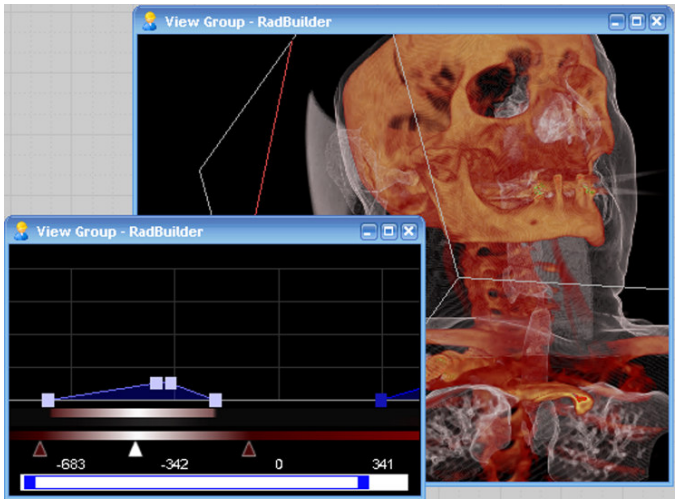

Figure 4.12: A zoom into the scene. The trapezoid controls the visualization of the light colored skin of the head.

possible to display different units such as Hounsfield Units, values from zero to one or different scalar resolutions to facilitate the transfer function setup. The application properties and the current transfer function settings can be saved as an inventor file (.iv). Colors, colorbars, primitives and positions are part of the parametric description which is written on hard disk.

Chapter 5

Mathematical Foundations

This chapter lays the foundations for the application of a statistical shape model to transfer function design. Principal Component Analysis (PCA) is used to reduce the number of parameters of a transfer function significantly. To understand how PCA works, some knowledge about statistics and matrix algebra is necessary. After the introduction of the most important mathematical background, the steps of PCA and its application to the research area of transfer function design are presented.

5.1 Statistics

PCA requires some calculations and quantities from the field of statistics. These are especially *random variables*, the *mean* or *expected value*, the *standard deviation*, the *variance* and the *covariance*. In general, statistics are useful to examine big sets of data. It often is interesting to analyze the relationship between individual data values. There are also powerful instruments to investigate multivariate data sets.

A random variable can be described as a function that maps events to numbers. If a person is selected randomly, then the height of this person will be an example of a random variable. The outcome is a numerical value that depends on chance because it differs for another chosen person.

The mean is the representation of the average value of a certain set of data. To show how the calculation of the mean is defined, an example set is helpful,

$$X = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6].$$

Following, X will is used as a reference for the entire set, n denotes the number of elements and $X_1 \dots X_n$ are the single elements of X. To get the mean $\bar{X}$ of the entire set X, all elements $X_1 \dots X_n$ have to be added and divided by n. The formula for this calculation is

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}.$$

In a mechanical interpretation the mean can be regarded as the center of gravity. Besides the indication of a middle point, the mean says nothing about the distribution

of the individual data values. The two sets of data [1 2 18 19] and [8 9 11 12] have the same mean ($\bar{X} = 10$), although the values of the sets are quite different.

Another important property of the data set is the *spread* of its data values. The measure of standard deviation tells the average distance of the values $X_1 \dots X_n$ from the mean. All the squared distances of the individual data values from the mean have to be added and divided by $n - 1$ before the positive square root is taken. The resulting formula for the standard deviations s of a sample is

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}}.$$

For the two sample data sets [1 2 18 19] and [8 9 11 12] the standard deviations are 9.8319 and 1.8257. That shows that the spread of the data values of the first data set is much higher than the spread of the second data set. If there is a set of data with only equal entries, then the standard deviation is zero because the individual data values correspond with the mean.

A very similar measure of the spread of the data values is the variance, defined as

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}.$$

The variance is just the square of the standard deviation and it is introduced because it is the basis for covariances which are needed for PCA. Standard deviation and variance are very useful measures for one-dimensional data sets like the height or the weight of a number of people. As there are data sets with more than one dimension it is necessary to have a measure for the statistical relationship between the dimensions. This allows to analyze if there are statistical dependencies between the height and the weight of the chosen set of people.

Therefore, the covariance can be calculated between two dimensions. It indicates the variation from the mean of the dimensions with respect to each other. For a three-dimensional data set ($x,\ y,\ z$), the covariances between x and y, between x and z and between y and z can be calculated. The covariance between one dimension and itself is equal to its variance. The formula of the covariance is quite similar to the formula of the variance,

$$cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}.$$

Because of the fact that the equations for variance and covariance just differs in the term $(X_i - \bar{X})^2$ which is replaced by $(X_i - \bar{X})(Y_i - \bar{Y})$, $cov(X, Y)$ is equal to $cov(Y, X)$. Regarding to the previous mentioned two-dimensional data set, the dimension H represents the height and the dimension W represents the weight of a selection of people. The sign of the resulting $cov(H, W)$ is most important. If the result is a positive value, there will be a positive linear correlation which means that both dimensions increase together. A negative result indicates a negative linear correlation which means that high values of one dimension are associated with low values of the

other dimension. If the covariance of two dimensions is zero, both dimensions will be independent to each other.

As seen before, the covariance is a value determined for two dimensions. If there are more than two dimensions all covariances can be calculated and put together in a matrix. For n dimensions there exist $\frac{n!}{(n-2)!*2}$ different values for covariances. The formula for the covariance matrix of a n-dimensional data set is

$$C^{n\times n} = (c_{i,j},\ c_{i,j} = cov(Dim_i, Dim_j)),$$

where the matrix $C^{n\times n}$ has n rows and n columns and the xth dimension is represented by Dim_x. The resulting matrix for a dataset with the three dimensions y, x and z is the square matrix with three rows and three columns

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}.$$

Covariance matrices are always symmetrical about the main diagonal. The values down the main diagonal represent the variances of the single dimensions [41].

5.2 Matrix Algebra

Besides a basic knowledge of matrices especially eigenvectors and eigenvalues of a matrix are important to understand the steps of PCA. This section shows how eigenvectors and eigenvalues are calculated and what they are good for. It is possible to multiply two matrices if the one on the left side of the multiplication is a $r \times s$-matrix and the one on the right side is a $s \times t$-matrix. Providing the matrix is of the right size, it can also be multiplied with a vector where the matrix is on the left side of the multiplication. A two-dimensional vector $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$ can be interpreted geometrically as an arrow pointing from the origin $(0,0)$ to the point $(2,3)$. If a square matrix A and a compatible vector $\vec{x}$ are multiplied in the form of $A\vec{x}$, then the matrix will be regarded as a transformation matrix. The result is a vector which is transformed from its former position. In 2D, the matrix of an angle α counterclockwise rotation about the origin is defined as

$$R = \begin{pmatrix} cos\,\alpha & -sin\,\alpha \\ sin\,\alpha & cos\,\alpha \end{pmatrix}.$$

The rotation is performed by the multiplication of the rotation matrix with a vector

$$\vec{x}' = R\vec{x}.$$

An eigenvector of a matrix is the vector which is changed in its length but not in its direction (but the vector can point in the opposite direction) when multiplied with the matrix on the left side of the multiplication.

The matrix A in the multiplication $A\vec{x}$ can be regarded as a function which takes the vector $\vec{x}$ as an input parameter and calculates a reflection of $\vec{x}$ on itself. All

multiples of $\vec{x}$ are also eigenvectors of the transformation matrix A because they just have a different length than $\vec{x}$. There are not eigenvectors for every matrix. The matrix has to be square that corresponding eigenvectors are possible, but not every square matrix has eigenvectors. If a square $n \times n$-matrix has eigenvectors, then there will be n of them. Another important property of eigenvectors is that all eigenvectors of a matrix are orthogonal to each other. This is very important for PCA because the data will be transformed from their original multidimensional space in a way that they are expressed in terms of the orthogonal eigenvectors.

As not the length of an eigenvector is important but its direction, the eigenvectors usually are normalized that they have the length 1. To normalize any vector, it has just be divided by its length. For small matrices it is not difficult to calculate the eigenvectors but if the matrices are larger, the eigenvectors will be computed by iterative methods. Eigenvectors and eigenvalues are closely related and for every eigenvector there is an eigenvalue or vice versa: There is an eigenvector for every eigenvalue. The task of finding a number λ and a corresponding vector $\vec{x}$ $(\neq \vec{0})$ for a square matrix that

$$A\vec{x} = \lambda\vec{x}$$

is called the eigenvalue problem.
In this formula the number λ represents the eigenvalue which can be a real or a complex number and the vector $\vec{x}$ is called the eigenvector of the matrix A. Every multiple $c\vec{x}$ (c can be any real number other than 0) of the vector $\vec{x}$ is an eigenvector, too. As it is not trivial to solve this equation because of the two unknowns $\vec{x}$ and λ, it is necessary to rewrite the equation to

$$(A - \lambda I)\vec{x} = \vec{0},$$

where I is the identity matrix filled with 1 down its diagonal and 0 anywhere else. If the matrix A is symmetric, then the resulting eigenvalues will be real numbers. At this point it is important to mention that PCA is based on the calculation of the eigenvalues and eigenvectors of the covariance matrix. The covariance matrix is square, symmetric and positive semidefinite.

As the eigenvalues of a positive semidefinite matrix are ≥ 0, the eigenvalues of a covariance matrix are real numbers ≥ 0. For the calculation of the eigenvalues the *characteristic polynomial* of A is needed which is defined as

$$p_n(\lambda) = det(A - \lambda I),$$

where n is the degree of the polynomial. Thus, it appears that the characteristic polynomial is calculated as the determinant of the resulting matrix where λ is subtracted of the main diagonal of A. The eigenvalues of A are the solutions for the equation

$$det(A - \lambda I) = 0.$$

In the following example the calculation of the eigenvalues of a 2×2-matrix is demonstrated.

$$A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

$$
\begin{aligned}
det(A - \lambda I) &= \begin{vmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{vmatrix} \\
&= (3-\lambda)(3-\lambda) - 1 \times 1 \\
&= \underbrace{\lambda^2 - 6\lambda + 8}_{\text{characteristic polynomial}} = 0
\end{aligned}
$$

It is possible to rewrite the equation

$$\lambda^2 - 6\lambda + 8 = 0$$

as

$$(\lambda - 2)(\lambda - 4) = 0.$$

The equation will be true if λ is 2 or 4 and thus, $\lambda_1 = 2$, $\lambda_2 = 4$ are the eigenvalues of the matrix A and the solutions of the equation $p_n(\lambda) = 0$.

For some kinds of matrices the calculation of the eigenvalues is not necessary. If A is a diagonal or triangular matrix, then the values of the main diagonal are the eigenvalues of A. Another fact is that the sum of the eigenvalues is equal to the sum of the values in the main diagonal of A. In the case of 2×2-matrices, it is enough to find one eigenvalue because the other is the sum of the values in the main diagonal subtracted by the found eigenvalue.

The corresponding eigenvector $\vec{x}_i$ to an eigenvalue λ_i is the solution of the equation

$$(A - \lambda_i I)\,\vec{x}_i = \vec{0}.$$

The first eigenvector can be calculated with the equation

$$(A - 2I)\,\vec{x}_1 = \vec{0}.$$

A possible solution of this equation is $x_1 = -1$, $x_2 = 1$ and the resulting eigenvector is

$$c \begin{pmatrix} -1 \\ 1 \end{pmatrix},$$

where c is a real number other than zero. The equation to find the second eigenvector is

$$(A - 4I)\,\vec{x}_2 = \vec{0}$$

with the possible solution $x_1 = 1$, $x_2 = 1$ and the resulting eigenvector

$$c \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The two eigenvectors of the matrix A are orthogonal to each other [1, 41].

5.3 Principal Components Analysis

After the introduction of the most important mathematical foundations to understand PCA, it is time for a closer look how this method works. PCA is used in various fields of applications like neuroscience, geoscience and computer science as a tool for data analysis. Beside its benefit for pattern identification it is very useful for the extraction of relevant information from large and confusing sets of data. In the case of multivariate data sets it is possible to reduce the number of dimensions without much loss of information. In the space of reduced dimensionality it is much easier to identify hidden dynamics within the data.

The major goal of PCA is to find the most meaningful bases for the given set of data. If the data is represented in terms of these bases, then the inner correlation of the data can be revealed. Often, not all data values are important because of high redundancy within the data set. PCA calculates the new bases which are linear combinations of the old ones. This precondition reduces the amount of potential new bases. Assuming that X is a $m \times n$-matrix which is linear transformed by P to be re-expressed as a new $m \times n$-matrix Y. The mathematical notation for this transformation is

$$PX = Y.$$

In this equation P is a rotation matrix which transforms X to Y. The following sections show that the rows of P are the principal components of X and the new basis vectors for the columns of X. As the problem is defined now, the task is to find an optimal way to re-express X. Therefore, it is important to choose the right transformation matrix P.

5.3.1 Matrix of Observations

In the last section the matrix X was introduced without an exact description what kind of values are the elements of this matrix. The input values of the principal components analysis are lists of measurements made of a certain set of individuals or objects. For instance, consider the measurement of the height h and the weight w of a collection of n individuals. Then, the matrix X is the matrix of observations with the columns as observation vectors $X_1 \dots X_n$. This matrix can be written as

$$X = \begin{pmatrix} h_1 & \cdots & h_n \\ w_1 & \cdots & w_n \end{pmatrix}.$$

Figure 5.1 shows a possible two-dimensional scatter plot of the observation vectors. The distribution of the points indicates the redundancy in the data. If the arrangement of the points looks like a circular-shaped point cloud, then there will be little or no redundancy. If the points are located around a line, then there will be high redundancy in the data. For example, a scatter plot of h and temperature most likely leads to a circular shape. In contrast, a scatter plot of weight in kilogram and weight in pounds results in perfectly aligned points. As the points of the scatter plot of w and h appear like an ellipse, there is some redundancy in the data. The idea behind

dimensionality reduction is the expression of redundant data as a linear combination of the original data. Thus, the data can be presented more meaningful [26].

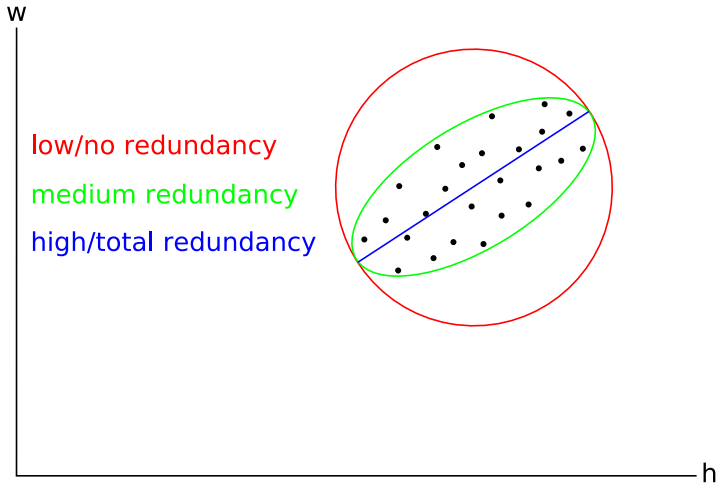

Figure 5.1: 2D scatter plot of the observation vectors.

5.3.2 Mean and Covariance

The next step is the calculation of the mean for the dimensions of the data set. With the definition of the observation vectors it is possible to calculate a vector M with the means of all dimensions as

$$M = \frac{1}{n}(X_1 + \ldots + X_n).$$

This mean vector M has to be subtracted from all observation vectors X_i for $i = 1 \ldots n$ with the formula

$$X_i' = X_i - M$$

to build a new matrix

$$D = (X_1' \ldots X_n').$$

Matrix D is the mean-deviated representation of the matrix of observations X. Figure 5.2 shows the two-dimensional scatter plot of the mean-deviated observation vectors. The mean vector M is the new center of the scatter plot. As a result the means of the single rows of the matrix D are zero. With the help of the mean-deviated matrix D, the covariance matrix can be calculated. The formula

$$C = \frac{1}{n-1} DD^T$$

calculates the positive semidefinite covariance matrix C.

C is a symmetric $m \times m$-matrix, where m is equal to the number of dimensions of the data set. For instance, the covariance matrix for the data set with the height

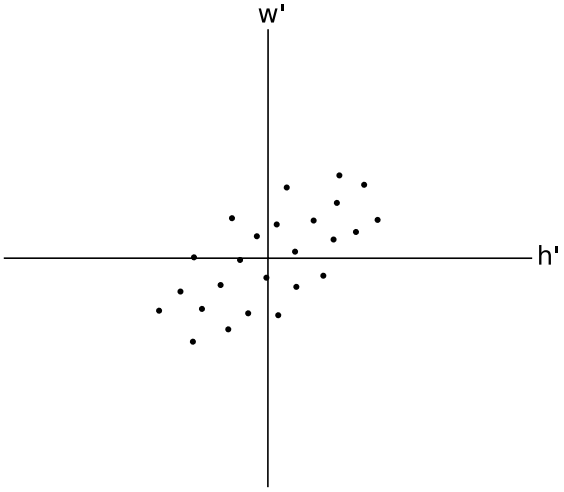

Figure 5.2: 2D scatter plot of the mean-deviated observation vectors.

and the weight is a 2×2-matrix. As introduced in the previous Section 5.1, each entry of C along the main diagonal represents the variance of one dimension and each other entry describes the covariance between two dimensions. Positive off-diagonal values of the covariance matrix indicate that the values of two dimensions increase together. Negative values show that the values of one dimension increase if the values in the other dimension decrease. Zero entries indicate that the two dimensions are independent of each other.

Another aspect which is important for PCA is the total variance of the data. This measure is calculated by the sum of the diagonal entries of the covariance matrix and is called the trace $tr(C)$ of the matrix [26].

5.3.3 PCA Core Calculations

As the goal of PCA is the removal of redundancy, the data have to be transformed in a way, that the corresponding covariance matrix is a diagonal matrix where all off-diagonal entries are zero. If this is achieved there will be no redundancy in the data anymore. This can be realized with the already mentioned change of the bases.

A matrix P is needed which transforms the mean-deviated matrix D with the equation

$$PD = Y$$

in way, that the covariance matrix of the new data Y will be diagonal. The rows of Y as the new dimensions of the data are arranged in order of decreasing variance and they are independent of each other. These tasks can be fulfilled by a matrix P where the rows are the unit eigenvectors of the covariance matrix of D. The eigenvectors are ordered by the values of the corresponding eigenvalues. This means that the transposed eigenvector in the first row of P is the one with the highest eigenvalue. This vector is called the first principal component. With the help of the first principal component, the first row or dimension of Y is calculated. The second

principal component is used for the calculation of the second row of Y, and so on.

Let u_1^T be the first row of the transformation matrix P which is the first principal component and y_1 be the first row of the new data matrix Y. Then y_1 can be calculated with the equation

$$y_1 = u_1^T D.$$

This shows that y_1 is a linear combination of the mean-deviated data values weighted by the principal component u_1. The computation of the other entries of Y can be done in an analog way. Thus, Y is the final data matrix with the dimensions in its rows and the data items in its columns.

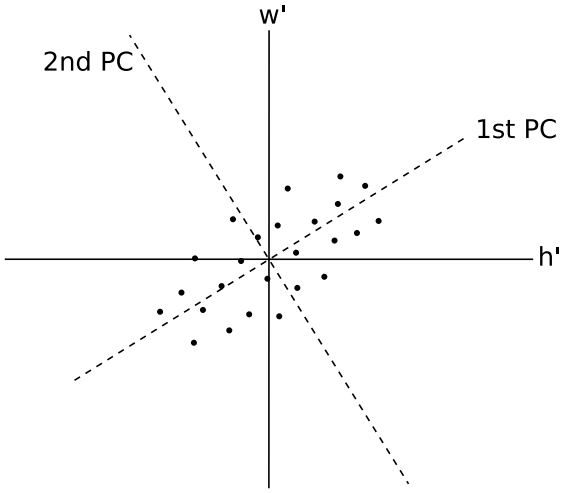

Figure 5.3: 2D scatter plot of the mean-deviated observation vectors and the eigenvectors of the covariance matrix.

Figure 5.3 shows the two-dimensional scatter plot of the mean-deviated observation vectors and the eigenvectors of the covariance matrix as dashed lines. The fact that PCA assumes that the direction with the largest variance is the most important one is illustrated. The first principal component is a line along the maximal variance. Smaller variances in a n-dimensional space are marked by the following principal components which are orthogonal to all other principal components.

It can be recognized, that the variance along the direction of the first principal component is higher than the variance along the second one. If there is a n-dimensional data set where $n > 2$, then the first few principal components will express most of the variance in the data. The last principal components point in the directions with less variance. After the transformation of the original data to the new data matrix Y, it is possible to represent the data in terms of the eigenvectors instead of in terms of their previous axes. The mean-deviated original data can be re-calculated with the equation

$$D = P^{-1}Y.$$

The inverse of P is equal to the transpose of P because the elements of P are the unit eigenvectors of the data set and the equation can be written as

$$D = P^T Y.$$

As D represents the mean-deviated data, the last thing to do to get the original data back is the addition of the mean vector M. Thus, the final equation for the re-calculation of the original data X is

$$X = (P^T Y) + M.$$

To summarize PCA, its steps are data acquisition, subtraction of the mean, calculation of the covariance matrix, calculation of the eigenvalues and eigenvectors, building of the transformation matrix, deriving of a set of transformed data and the recalculation of the original data [26, 41, 40, 14].

5.3.4 Dimensionality Reduction

In the last section the building of the transformation matrix P with the eigenvectors was shown. PCA can be used to reduce the dimensionality of the data. The principal component is the eigenvector with the highest eigenvalue. If the eigenvectors are ordered by the value of their corresponding eigenvalue, then the less important ones can be ignored without much loss of information. This can be done because the main variances in the data are expressed by the first few principal components. If there are originally n dimensions and only p eigenvectors are chosen to build the transformation matrix P, then the final data set will be reduced to p dimensions.

Now the interesting question is how the importance of an eigenvector can be determined. The percentage of the importance of an eigenvector is the corresponding eigenvalue divided by the sum of all eigenvalues. As the results are values in the range $[0, 1]$, they have to be multiplied by 100. In the extreme cases that the percentage is 100 or 0 percent, the corresponding eigenvector expresses the total variance of the data or no variance at all. The decision how many eigenvectors should be used can be made by the calculation of the cumulative percentage of total variance. Assuming, there are the five ordered eigenvalues

$$[6.2412 \;\; 1.2318 \;\; 0.9857 \;\; 0.3196 \;\; 0.1483].$$

Table 5.1 shows the percentage of variance of the sample eigenvalues and the cumulated percentages. A common way for the decision how many eigenvectors should be

eigenvalue	percentages	cumulated percentages
6.2412	70.55	70.55
1.2318	13.92	84.47
0.9857	11.14	95.61
0.2396	2.71	98.32
0.1483	1.68	100.00

Table 5.1: Eigenvalues, percentage of variance and the cumulated percentage for a sample set of eigenvalues.

used, is to set a certain goal-percentage. If the reduction of the dimensionality has

to preserve 95% of the data's variance in the example, then the eigenvectors for the first three eigenvalues will be taken to build the transformation matrix. The result is a reduction from five to three dimensions.

Other common methods for choosing the number of eigenvectors are the *Kaiser's rule* [16] and the *scree test* [3]. The *Kaiser's rule* keeps just the eigenvectors with eigenvalues > 1 and the *scree test* is based on the steepness of the plotted eigenvalues [14, 5].

Chapter 6

Implementation

Up until now the transfer function editor with its framework was presented and Principal Component Analysis was introduced. This chapter describes the implementation of a PCA node and its integration into the existing framework. There is also an introduction of the used library for matrix calculations and a presentation of the node's user interface. To close this chapter the typical workflow of PCA in combination with the transfer function editor is shown.

6.1 Matrix Library

As introduced PCA is based on several matrix calculations such as the identification of eigenvalues and eigenvectors, matrix multiplications or the building of the transposed matrix. The free *Newmat* C++ matrix library was the choice to facilitate the matrix calculations. It supports various types of matrices like *RectangularMatrix, UpperTriangularMatrix, LowerTriangularMatrix, DiagonalMatrix, SymmetricMatrix, BandMatrix, UpperBandMatrix, LowerBandMatrix, SymmetricBandMatrix, IdentityMatrix,* as well as *RowVector* and *ColumnVector*.

Newmat is supposed to help scientists and engineers with the manipulation of matrices by using standard matrix operations. The element type of the matrices has to be either *float* or *double*. Newmat especially is optimized for large matrices from 10×10-matrices upwards. Some of the supported operations are multiplication, addition, subtraction, concatenation, conversion between types, submatrix, determinant, singular value decomposition and eigenvalues of a symmetric matrix [31]. Another popular matrix library is the source code to the book *Numerical Recipes in C++* [36].

6.2 PCA Node

As the functionality of PCA is implemented in an OpenInventor node, it can be integrated in an existing RadBuilder network. The PCA node needs just a field-connection with the parametric description which was introduced in Section 4.2.1. This field is hold by *SoRadTFEditorParams1D*. Figure 6.1 shows how the PCA node is integrated in the network. If this field-connection is established, then the PCA

node is able to extract the relevant information for the dimensionality reduction. The access to the parametric description also allows the manipulation of this field. This offers the functionality that the PCA node can write back the result of its calculations. Changes in the parametric description lead automatically to a new rendering process and the adjustment of the transfer function primitives. To understand how the PCA

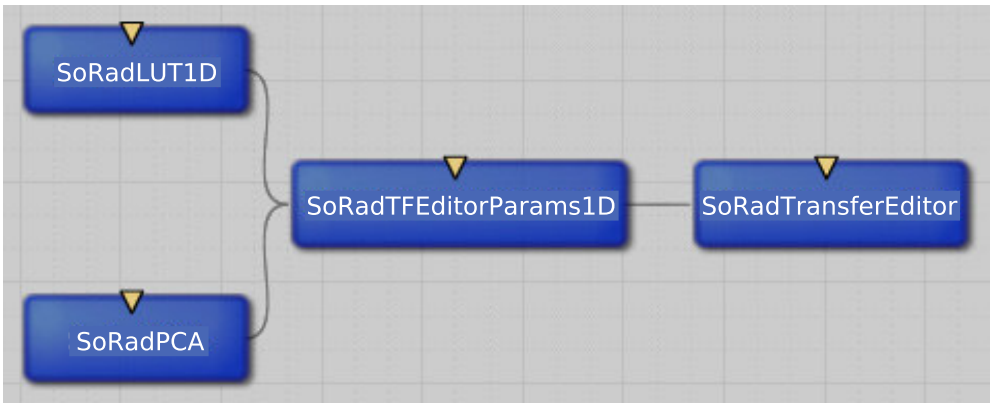

Figure 6.1: The integration of the PCA node into an existing network.

node works, it is helpful to have a closer look at its most important fields.

SoSFNode tfeParams1D_Node: This field is the most important one because it is the source of the input data and the target for the output data. *SoRadTFEditorParams1D* is connected with the parametric description in *SoRadTransferEditor* and provides this connection for *SoRadLUT1D* and *SoRadPCA*. *SoRadPCA* extracts the parameters of the transfer function from the parametric description. These parameters are the x-positions of the trapezoid-points as well as the heights of the trapezoids. After the calculation of the new transfer function, the new parameters are written back to the parametric description. This leads to a remote control of the trapezoids defining the transfer function and initiates a new rendering process.

SoSFInt32 numTFs: The number of transfer functions which are used for the PCA is stored in this field. This value represents the number of data items for the matrix of observations. Thus, the value accords with the number of columns of the matrix of observations.

SoSFInt32 numParamsPerTF: The number of rows of the matrix of observations is defined by the number of parameters per transfer function. Usually, the number of parameters is the number of trapezoids multiplied with the number of parameters per trapezoid. One trapezoid is defined by five parameters which are its height and the x-coordinates of its four edge-points. This means that the value of *numParamsPerTF* accords with the dimensions of the data set. All input transfer functions for the PCA have to be equal in the number of their parameters.

SoSFFloat informationPercentage: This field contains a value between 0 and 1 which is crucial for the dimensional reduction. The value represents the cumulated percentages of the data's variance which has to be preserved in the transformed data. If a value of 0.95 is chosen, the final data will contain 95% of original data's variance.

SoMFFloat percentages: The percentages of the contribution to the total variance of all eigenvectors are stored in this field in decreasing order. That means that the number of entries is equal to the number of eigenvectors and the first entry indicates the amount of variance which is represented by the first principal component. The cumulated percentages are calculated with the values in this field.

SoSFInt32 numEigenvectors: Based on the cumulated percentages, the necessary number of eigenvectors is stored in this field. The final data will contain as much dimensions as this field indicates. With the number of eigenvectors, the data's variance with at least the percentage which is stored in *informationPercentage* is ensured.

SoMFFloat data: The parameters of all attached transfer functions are saved in this field. Thus, the field is the base building the matrix of observations.

SoSFFloat minHeight and SoSFFloat maxHeight: Without little adjustments the output transfer function can cause trapezoids with a height of more than 1.0 or less than 0.0 for some configurations. To avoid this, the maximal height and the minimal height of the trapezoids of all input transfer functions are stored. If the height of an output trapezoid is out of this range, it will be adjusted to the value of *minHeight* or *maxHeight*.

SoMFFloat mean: This multifield contains the means for each dimension of the data set.

SoMFFloat dataAdjust: The *mean* is subtracted from the corresponding values in the *data*-multifield and the results are stored in *dataAdjust*. This field contains the values of the mean-deviated data matrix.

SoMFFloat eigenvalues: This field stores the calculated eigenvalues of the covariance matrix for the mean-deviated data matrix.

SoMFFloat eigenvectors: The values of the corresponding eigenvectors are stored in this multifield.

SoMFFloat featurevector: Only the eigenvectors which are necessary to preserve the required *informationPercentage* are contained in this field. That means the number of eigenvectors is determined by *numEigenvectors*.

SoMFFloat finalDataMins and SoMFFloat finalDataMaxs: As it is not the goal to calculate back a data set which contains the same amount of transfer

functions as the number of input transfer functions, it is not necessary to use the total set of the final data. It is exactly one transfer function needed. These two fields save the minimal and the maximal value of each row of the final data matrix. The number of entries of each multifield is equal to the value in *numEigenvectors*.

SoMFFloat finalDataAdjust: For the calculation of the output transfer function *finalDataAdjust* represents the selection of values from the final data matrix. The values are user-defined within the range of the corresponding values in *finalDataMins* and *finalDataMaxs*.

SoMFString TFnames: Every attached transfer function can get a user-defined name. This provides the possibility to apply an already defined transfer function to a the connected data set.

After this presentation of the node's elements the steps of the PCA implementation are described. At first, the data matrix has to be built. Therefore, it is necessary to parameterize the transfer functions. In the presented implementation each trapezoid of a transfer function has five parameters, namely the x-positions of its edgepoints and its height. These parameters are regarded as the random variables.

Assume x_{nm} are the elements of the data matrix where n is the number of the parameters of one transfer function and m is the number of different transfer functions. Then, the data matrix is defined as

$$X = \begin{pmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \end{pmatrix}.$$

The second step is the calculation of the means for each row of the data matrix. It is necessary to subtract the means from the corresponding entries in X to get the mean-deviated data matrix D. After this step the covariance matrix C of D is calculated. The covariance matrix is a symmetric matrix with n rows and n columns. With the support of the matrix library the eigenvalues and eigenvectors of C are calculated. Now the number of eigenvectors is chosen, based on the cumulated percentages of the corresponding eigenvalues. If i eigenvectors are chosen with $0 < i \leq n$, then the transformation matrix P will be a $n \times i$-matrix with the eigenvectors as columns. The two matrices for the calculation of the final data matrix Y are built and the formula

$$Y = P^T D$$

produces a matrix with i rows and m columns. In other words, the number of rows is equal to the number of chosen eigenvectors and the number of columns is equal to the number of transfer functions.

Following, the two multifields *finalDataMins* and *finalDataMaxs* are filled with the minimal and the maximal values of each row of Y. Let $n_1 \dots n_i$ be the entries in *finalDataMins* and $m_1 \dots m_i$ be the entries in *finalDataMaxs*. Then, it is up to the

user to define an i-dimensional vector $\vec{v}$ where the first entry is a value in the range $[n_1, m_1]$, the second entry is a value in the range $[n_2, m_2]$ and so on. How the user can define this vector with i sliders is shown in the following section.

The last step is the calculation of the vector $\vec{t}$ which contains the values for the output transfer function with the formula

$$\vec{t} = P\vec{v}.$$

As all the input transfer functions have n parameters, $\vec{t}$ is a n-dimensional vector. The values of $\vec{t}$ can be written to the parametric description which is connected to the PCA Node. The change of the parametric description initiates an adjustment of the trapezoids and a new rendering process.

6.3 User Interface

The user interface of the PCA node is kept clear and simple. Figure 6.2 shows the two appearances of the interface. Initially, the windows display the state shown in

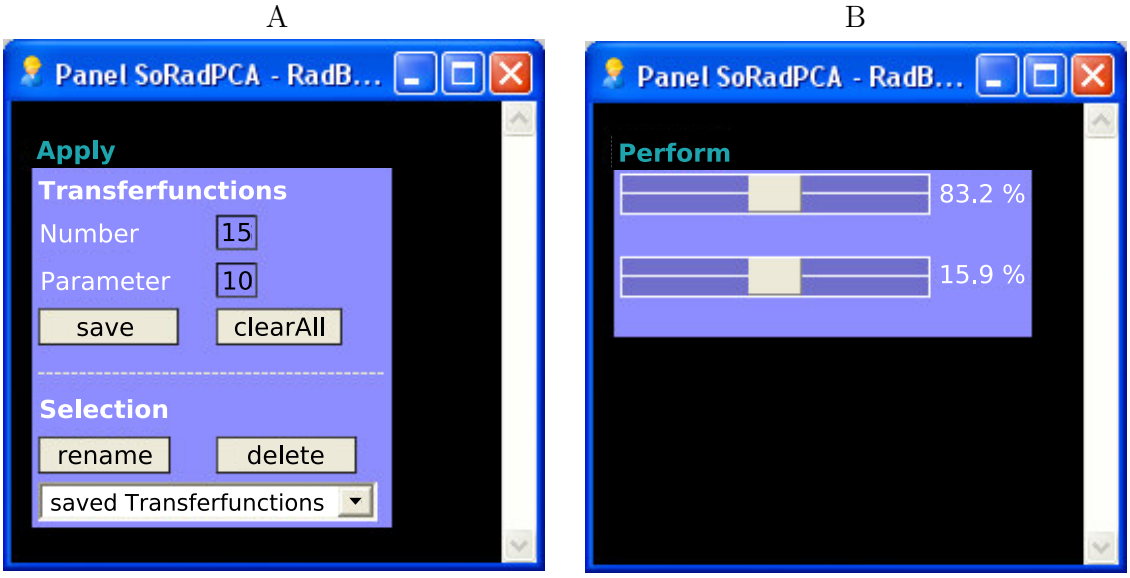

Figure 6.2: The user interface of the PCA node. Initially, the window for the management of the transfer functions is displayed (A). The window with the sliders (B) facilitates the generation of the output transfer function.

Figure 6.2 (A), with zero entries in the fields for *Number* and *Parameter*. If a transfer function is connected to the PCA node, it will be saved by a click on the *save* button. The *Number* and *Parameter* fields will display 1 and the number of the transfer function's parameters. Following transfer functions will be only saved if the number of parameters corresponds to the number of parameters of all previous saved transfer functions. The implementation of the *save*-functionality takes care about the number

of parameters, the connection to a valid transfer function, the naming of the transfer functions and avoids the multiple saving of identical transfer functions. All saved transfer functions can be deleted with the *clearAll* button.

The lower part of the window provides buttons for the renaming and the removal of saved transfer functions. With the combobox for the saved transfer functions the application of each saved transfer function is possible. This feature especially is interesting if a new CT data set is loaded because it is very easy to apply transfer functions which were designed for other data sets. Following, the selected transfer function can be adjusted to show the features of interest of the new data set optimally.

If at least two transfer functions are saved, the *Apply* link can be clicked to display the slider window shown in Figure 6.2 (B). The number of sliders is equal to the number of eigenvectors which is determined by the cumulated percentages. At the right of the sliders the percentage of information of the individual eigenvector is displayed. This it the percentage of the original data's variance which is preserved with this eigenvector. The position of the slider represents a value in the range $[0, 1]$. Let s be the value of the first slider, min the first entry of *finalDataMins* and max the first entry of *finalDataMaxs*. As the slider is used for the definition of a value between two corresponding entries in *finalDataMins* and *finalDataMaxs*, the first value v_1 for the vector $\vec{v}$ that is inroduced in the last section is calculated with the equation

$$v_1 = min + s \times (max - min).$$

The other sliders are used for the calculation of the further elements in $\vec{v}$ in the same fashion. A click on the *Perform* link leads back to the transfer function management window and it is possible to define more transfer functions.

6.4 Workflow

Two possible scenarios of using the PCA node are imaginable. For both scenarios it is necessary to load at first a data set with the *SoRadLoadRaw* node. This node needs information about the file location, the used bits, the voxel type (e.g. unsigned short) and the resolution of the data set. Furthermore, a model matrix needs to be defined which is derived from the resolution and the slice thicknesses.

The next step is the setup of a transfer function with an arbitrary number of trapezoids. If a good transfer function is defined which shows the features of interest in the data set, the transfer function can be saved with the PCA node as described in the previous section. Following, it is possible to define more transfer functions for the same data set which leads to the display of different features of interest. If a certain number of defined transfer functions is saved, the PCA node can perform its calculations. As a result, it is possible to navigate through the space of appropriate transfer functions for the one data set with a significant reduced set of parameters. This navigation is done with the user friendly sliders. Of course, it also is possible to discover similar data sets with the recorded transfer functions and the slider control.

The other way of using the PCA node's functionality is to define transfer functions for different data sets which are recorded for the same examination purpose. In the

following chapter, a couple of CTA data sets of the human head are taken. For each of the data sets one transfer function is defined which shows existing aneurysms in an optimal way. After the PCA calculations are performed, it is possible to visualize the aneurysms in a data set of the same type with only one single slider. The physician can use the slider for the setup of the transfer function. This process is now as simple and intuitive as the greyvalue windowing for slice images.

Chapter 7

Results

In this chapter the results of the presented work are shown. First of all, the specific goal is formulated. Followed by a description of the examined data sets. The transfer functions and the corresponding renderings of the input data sets are displayed. After the PCA node has performed its calculations the results are applied to independent data sets. The visualization of these data sets is presented for different slider positions. Decimal numbers in this chapter are rounded to four numbers behind the decimal point.

7.1 Goal

The visualization of intracranial aneurysms is the clinical examination scenario. A set of 14 CTA data sets is used to show how the Principal Component Analysis is used to reduce the number of parameters of the transfer function immensely. Before the acquisition of CTA data, a radiopaque substance is injected for the visualization of the blood vessels. This is necessary to isolate the blood vessels because they have the same density as the surrounding tissue. An aneurysm is a localized abnormal dilatation or ballooning of a blood vessel. Aneurysms often occur in the arteries at the base of the brain.

As the major goal is the simplification of transfer function setup it is interesting to determine how many parameters have to be adjusted by the user to navigate through the space of appropriate transfer functions. Because there is a lot of redundancy in the input transfer functions most of the variance of the input data can be expressed with the help of one slider in the investigated setup. As the input transfer functions for a given clinical scenario can be produced by experts, the physician just has to move this slider to generate the most meaningful image for a given data set. With this technique the transfer function setup is not a time-consuming process for the physician anymore.

7.2 Input Transfer Functions

The input data are 12 CTA data sets with the resolutions and used bits as presented in Table 7.1. For each of the data sets a transfer function which visualizes the bones and the blood vessels is defined. Two trapezoids are used to visualize these features of interest. The trapezoid for the bones has the same position for all transfer functions but the trapezoid for the blood vessels is individually adjusted.

data set	resolution	bits
CTA_1	512 × 512 × 121	12
CTA_2	512 × 512 × 246	12
CTA_3	512 × 512 × 121	12
CTA_4	512 × 512 × 138	12
CTA_5	512 × 512 × 189	12
CTA_6	512 × 512 × 167	12
CTA_7	512 × 512 × 62	12
CTA_8	512 × 512 × 85	12
CTA_9	512 × 512 × 31	12
CTA_10	512 × 512 × 173	12
CTA_11	512 × 512 × 64	12
CTA_12	512 × 512 × 109	12

Table 7.1: The resolutions and bits of the input data sets.

Figure 7.1 shows the trapezoid which assigns opacity and color to scalar values which represent the bones. It covers the range of Hounsfield Units from 886 to 3072

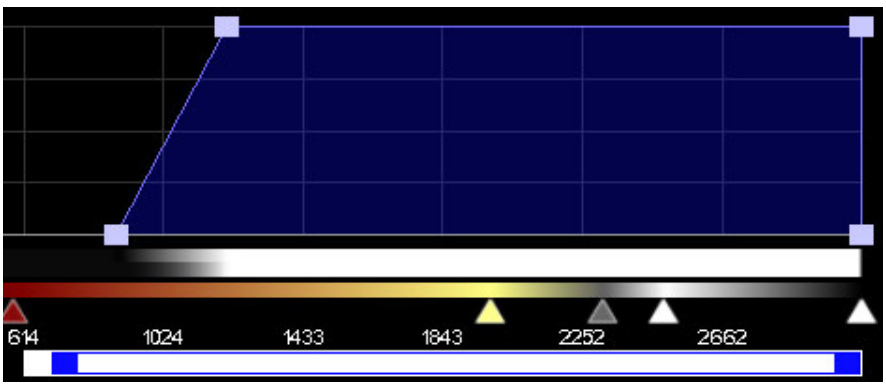

Figure 7.1: The trapezoid assigns opacity and color to scalar values which represent the bones.

and is parameterized as [0.4662, 1, 0.5451, 1, 1]. The parametrization of a trapezoid has five entries. It starts with the two lower edgepoints, followed by the two upper edgepoints and the height. This order is chosen arbitrarily and can be replaced by any other consistent parametrization. The value for the trapezoid's height represents

the opacity that is assigned to the covered scalar values. With the parametrization value of the edgepoint p_e, the corresponding Hounsfield Unit HU_e can be calculated with the formula

$$HU_e = -1024 + p_e \times 4096.$$

The trapezoid is using its own colors as described in the earlier Section 4.2.2. These colors are assigned to achieve a transition from black to white along the left slope.

In contrast to the trapezoid that represents the bones, the one that represents the blood vessels differs for all input transfer functions. The exact parametrization of this trapezoid in the transfer function definition for each data set is listed in Table 7.2. A transition from black to red along the left and the right slope of this trapezoid is achieved with the assigned colors. Each of the input transfer functions is defined

dataset	parametrization of the *blood vessels trapezoid*				
CTA_1	0.2801	0.3051	0.2894	0.3005	0.2274
CTA_2	0.2893	0.3656	0.3198	0.3243	0.1130
CTA_3	0.2705	0.3069	0.2830	0.2935	0.2012
CTA_4	0.2790	0.3253	0.3023	0.3113	0.3498
CTA_5	0.2743	0.3243	0.2914	0.3077	0.3761
CTA_6	0.2745	0.3431	0.2967	0.3243	0.2449
CTA_7	0.2926	0.3589	0.3198	0.3243	0.1399
CTA_8	0.2856	0.3202	0.2924	0.3103	0.2099
CTA_9	0.2773	0.3178	0.3023	0.3113	0.5772
CTA_10	0.3097	0.3599	0.3198	0.3292	0.1662
CTA_11	0.2949	0.3635	0.3198	0.3243	0.1487
CTA_12	0.2794	0.3132	0.2926	0.3003	0.1853

Table 7.2: The parametrization of the trapezoid that assigns opacity and color to scalar values which represent the blood vessels. This trapezoid is individually defined for each data set.

with two trapezoids and thus, it has ten position parameters. In Figure 7.2 and in Figure 7.3 the images for the input data sets, rendered with the corresponding transfer functions are presented. These images provide a good visualization of the blood vessel and in most of them aneurysms can be recognized clearly. All images are rendered with a ray casting implementation.

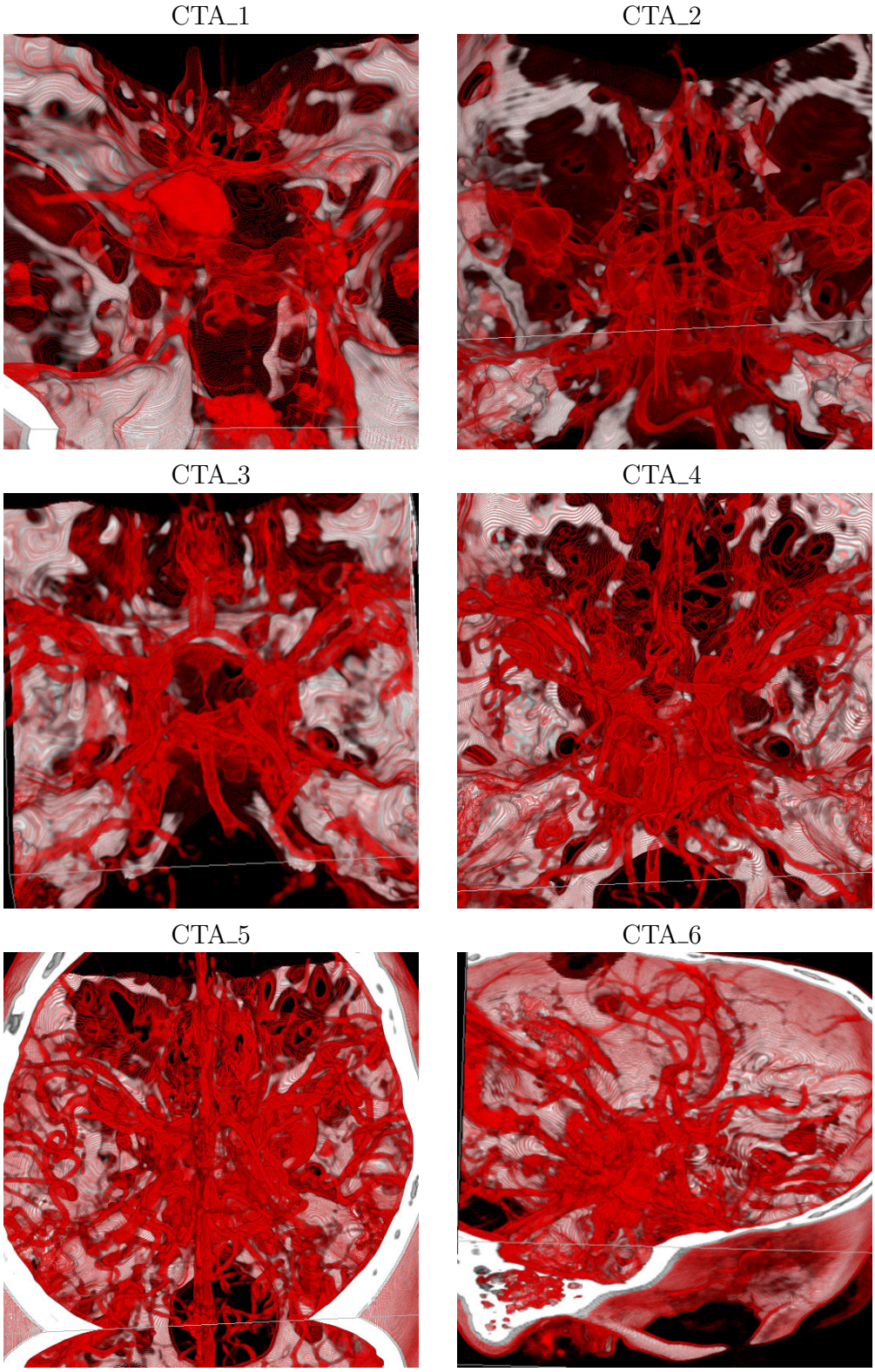

Figure 7.2: Images for the data sets CTA_1 - CTA_6 rendered with the presented transfer functions.

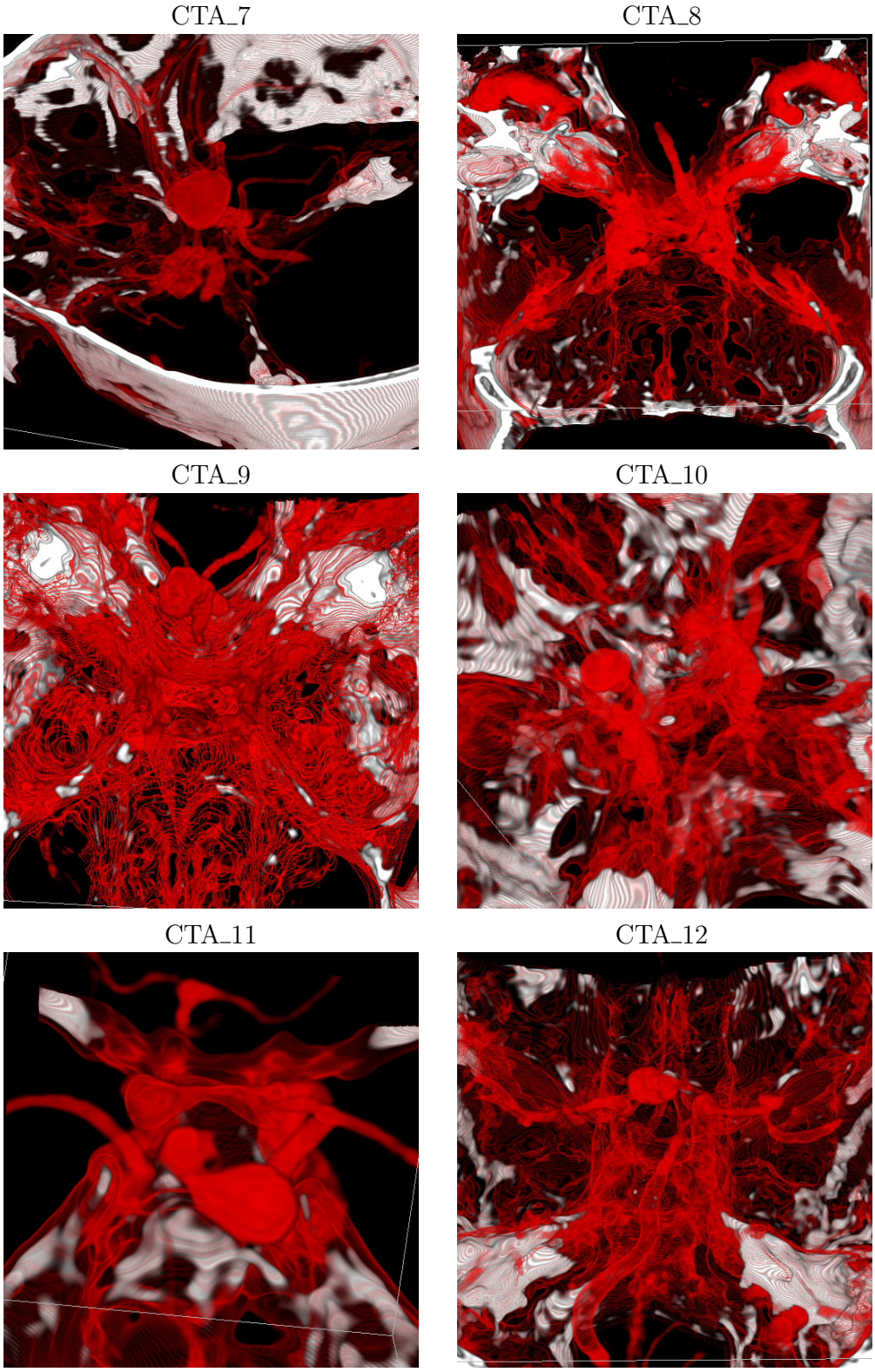

Figure 7.3: Images for the data sets CTA_7 - CTA_12 rendered with the presented transfer functions.

7.3 Dimensionality Reduction

The presented parameterizations of the input transfer functions are the elements of the matrix of observations. After the calculation of the covariance matrix and the eigenvalues, the cumulated percentages are calculated to determine how many eigenvectors have to be used for building the transformation matrix. Because each input transfer function is represented by ten parameters, the covariance matrix is a 10×10-matrix with ten eigenvalues and ten corresponding eigenvectors.

For the presented example five of the eigenvalues are non-zero and their eigenvectors describe the variances in the data. The percentages of variance which are represented by the five most principal components are listed in Table 7.3. More than

principal component	percentages	cumulated percentages
1^{st}	95.6247	95.6247
2^{nd}	3.9745	99.5992
3^{rd}	0.2855	99.8847
4^{th}	0.0933	99.9780
5^{th}	0.0220	100.0000

Table 7.3: The percentage of the variance which is represented by the first principal components.

95% of the variance can be expressed by the first principal component. With this result it is sufficient to provide one single slider for the user. This single slider facilitates the transfer functions setup extremely. Very meaningful images for the visualization of aneurysms are achieved by the generated output transfer functions.

7.4 Visualization Results

The slider is used to generate the output transfer functions for the two independent data sets listed in Table 7.4. Independent means that they are different than the data sets which are used as input. Because it is not possible to show the results of the

data set	resolution	bits
CTA_13	$512 \times 512 \times 101$	12
CTA_14	$512 \times 512 \times 77$	12

Table 7.4: The resolutions and bits of the data sets for the output images.

slider movement, the generated images are shown for six equidistant slider positions as illustrated in Figure 7.4. The positions are labeled with the letters A-F according to the headline letters of the Figures 7.5 and 7.6. This means that the image A is generated with the output transfer function for the slider position A. Figure 7.5 presents the images for data set CTA_13 rendered with the different output transfer

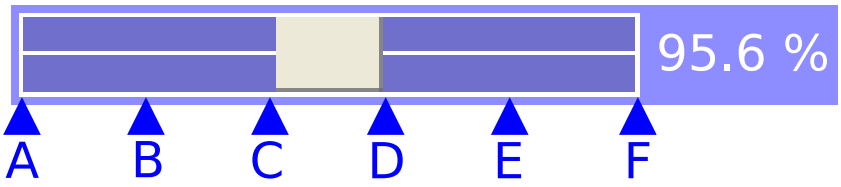

Figure 7.4: Output transfer functions are generated for six positions of the slider.

functions. The same transfer functions are used to render data set CTA_14. Figure 7.6 displays the resulting images.

In the case of both data sets meaningful results are generated. The aneurysms can be identified clearly in each image, but the opacities and the range of displayed scalar values differ slightly from one image to the next. The physician can select the result he likes best with an easy movement of one single slider.

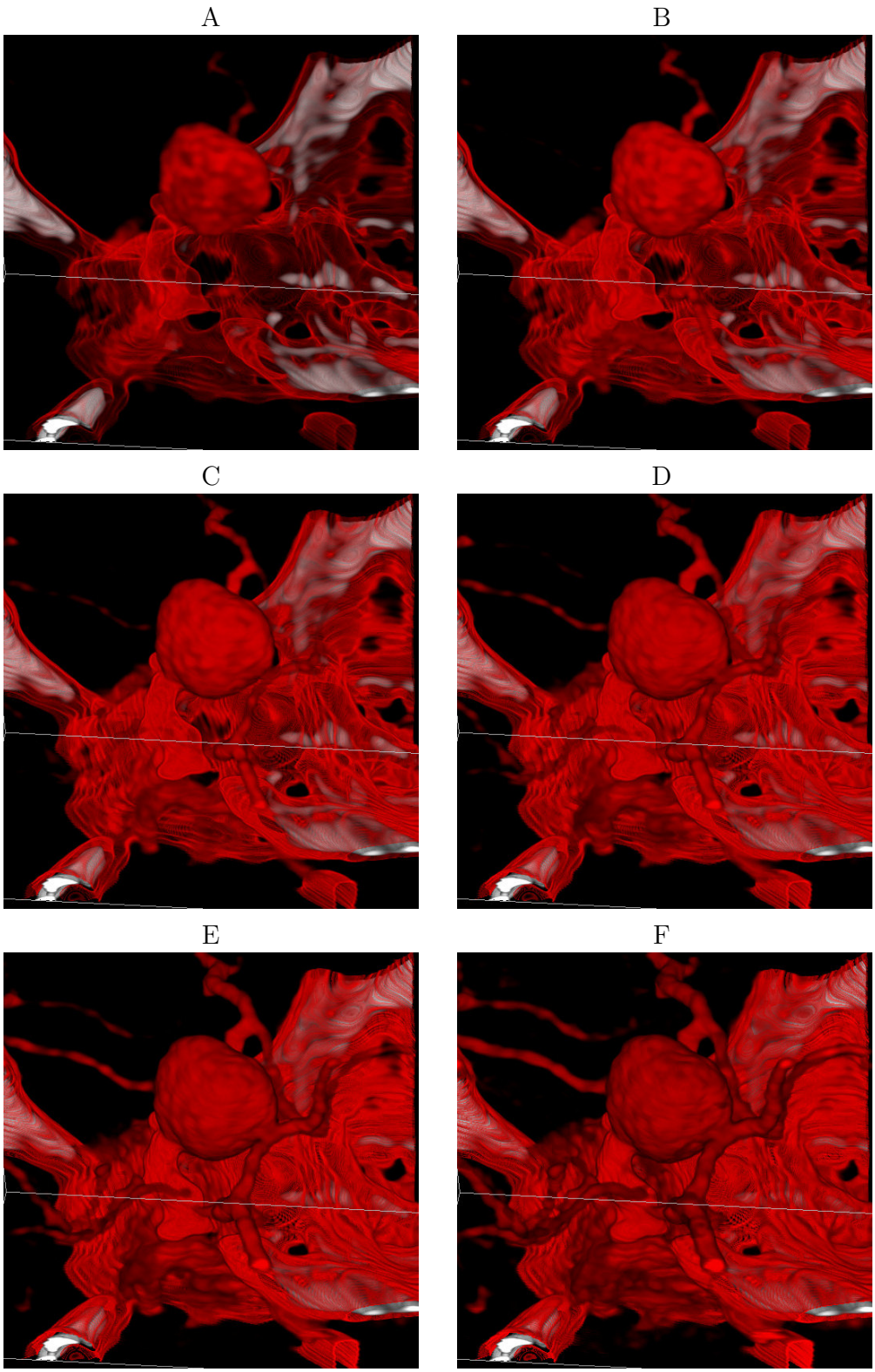

Figure 7.5: The generated transfer functions are used to render data set CTA_13.

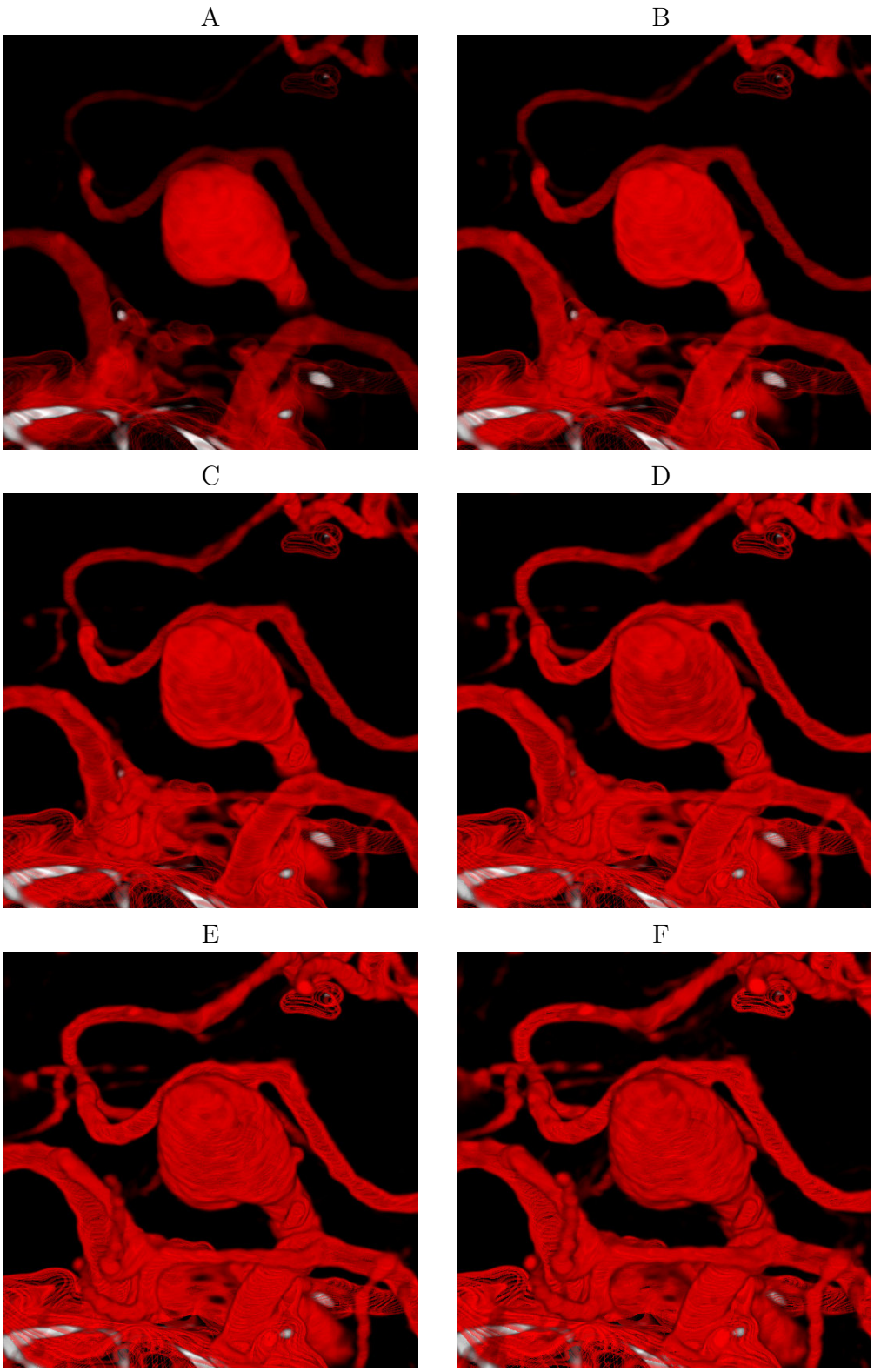

Figure 7.6: The generated transfer functions are used to render data set CTA_14.

Chapter 8

Conclusion

This chapter summarizes the major achievements of the work presented in this thesis. At first, this is the development of an easy-to use transfer function editor. Second, an application-driven technique for the design of transfer functions is introduced. Conclusions follow and the chapter closes with an outline of future challenges which are proposed to be done to improve the process of transfer function setup.

8.1 Summary

At the beginning of this work there is a motivation and the goal is formulated. It is shown that the physicians can greatly benefit from 3D visualization if some presumptions are realized. Especially the reproducibility of the whole visualization process, the availability of intuitive applications and the speed factor are very important for them.

Following, the focus is on computed tomography and algorithms for direct volume visualization. Computed tomography is introduced because it is a very popular recording technique for the acquisition of data sets for medical examinations. Algorithms for direct volume visualization such as ray casting or the shear-warp factorization are used for the generation of images for the data sets.

In general, transfer functions are either one-dimensional or multidimensional. The theory of both classes is introduced and the benefit of multidimensional transfer functions is described. It is illustrated how the consideration of the gradient magnitude leads to a clear identification of the boundaries in the data. For the design of transfer functions, tools based on interactive adjustment are widespread in scientific and commercial applications. Various approaches for the automatic or semi-automatic design of transfer functions have been developed in the last few years. These are either image-driven or data-driven techniques. Image-driven techniques are based on the image generation with different parameter settings. These approaches mainly differ in the way how the parameters are chosen. Data-driven techniques analyze the data instead of generated images. Thus, they are independent of image-related parameters such as pixel resolution or viewing position.

The developed 1D transfer function editor provides various functionalities for the

manual setup of transfer functions. Its implementation as OpenInventor nodes ensures high flexibility and portability. The used geometric primitives for the transfer function definition are trapezoids and ramps. As soon as they are added to the scene, they can be adjusted intuitively because the primitive elements (edgepoints and lines) are implemented as OpenInventor draggers.

Two different ways for the color-coding of the different tissues are implemented. At first, it is possible to define several colors for each primitives. These colors are bound to the primitives. Second, the primitive can adopt colors which are defined in a global colorbar. The final color which is used for the rendering process is displayed in a separate colorbar. All relevant information about the transfer function are stored in a parametric description. Each node which has a connection to this parametrization is able to extract the relevant information.

To be able to relate to the steps of Principal Component Analysis some mathematical foundations are necessary. In the range of statistics the mean and the variance of a data set are important basics. For a multivariate data set, the covariances can be calculated for the composition of a covariance matrix. Furthermore, some matrix algebra is required. Especially the calculation of the eigenvalues and the corresponding eigenvectors of the covariance matrix is important.

The Principal Component Analysis is implemented as an OpenInventor node, too. An intuitive user interface allows the acquisition of the input data via connections to the parametric descriptions of manually defined transfer functions. The input data are the parameterizations of the geometric primitives of of the transfer functions for several data sets. These data sets are recorded for a specific clinical examination. The collected parameters represent a point cloud in a high-dimensional space. Principal Component Analysis is used for the calculation of the principal axes of this point cloud. Because the input transfer functions contain a lot of redundancy it is possible to reduce the degrees of freedom in a high-level model for the transfer function definition significantly.

To demonstrate the benefit of the presented method, 14 different CTA data sets are used. The clinical scenario is the visualization of aneurysms. For 12 of these data sets transfer functions are manually defined and stored for the PCA calculations. The result is, that one slider is enough to produces output transfer functions which lead to meaningful visualizations. Images of two independent data sets are presented for several output transfer functions. All these images provide a good visualization of the aneurysms with slight but noticeable differences from one slider position to the next.

8.2 Conclusions

The developed transfer function editor facilitates the manual adjustment of the input transfer functions. Because of its intuitive set of functionalities this can be done very fast by experts for different clinical scenarios. The used primitives combined with the different possibilities of the color assignment build an easy-to-use but very powerful tool for the transfer function setup. Furthermore, the transfer function editor

is highly extendable and very portable because it is implemented as platform- and window system-independent OpenInventor nodes.

The node which performs the PCA calculations can be connected to the parametric description of the transfer function. With a clear user interface it is possible to save any number of transfer functions. The only presumption is, that all transfer functions have the same number of parameters. A combobox allows the selection of any previous saved transfer function. This can facilitate the transfer function setup for a new data set immensely if there is already a good amount of saved transfer functions. It is possible to apply a previous defined transfer function to the new data set before making final adjustments to produce an optimal image. The PCA node is independent of a certain parametrization. It needs only an access to the parameters of the transfer functions to work with any other transfer function editor.

The presented image-driven and data-driven techniques for the automatic or semi-automatic transfer function design can help to find a good transfer function. Image-driven techniques especially are useful for unexperienced users. With these approaches, the generation of good images is possible for them. The drawback of image-driven techniques is that most of them are not really fast and the process is hardly reproducible. A problem of the data-driven techniques is, that the provided user interfaces often are very complex and the demands on the user are too high. Both classes of approaches do not integrate the knowledge about what anatomical or functional structures are interesting for the user. Therefore, an application-driven technique is presented in this thesis. An expert can set up good transfer functions for several data sets for a specific clinical scenario.

With this approach, the physician who asks for clear and fast solutions can generate various transfer functions with the easy movement of a very limited number of sliders. In the presented example of the visualization of aneurysms, one single slider was sufficient to produce very meaningful images. The slider interaction makes transfer function setup as simple and intuitive as the greyvalue windowing for slice images.

8.3 Future Challenges

There are still some aspects that are worth a further investigation but it was not possible to address all of them in this work. First of all, it is interesting to evaluate a larger collection of data sets and to investigate other clinical scenarios than the visualization of aneurysms.

To avoid the necessity of the manual setup of the input transfer functions, the presented algorithm could be combined with automatic or semi-automatic methods. For example, image-driven techniques can be integrated to allow non-experts the definition of input transfer functions.

As described, multidimensional transfer functions can be used for the generation of images of a higher quality. Especially the clear isolation of neighbored tissues is achieved by the consideration of the gradient magnitude. If the reading and writing access to the parameters of a multidimensional transfer function is realized, the

presented algorithm will be able to handle multidimensional transfer functions, too.

Finally, semantic models for transfer functions can improve the transfer function setup for the non-expert user. These models can be used for a really goal-oriented definition of the transfer function. For a specific, well-defined application scenario an intuitive semantic in natural language can be provided by the user interface. The user choose the tissue he wants to visualize from a list and gives some orders for the way this tissue should be displayed. These orders can include formulations like *try to sharpen the vessels* or *make the brain tissue more transparent*.

Part II

German Part

Erklärung

Ich versichere, dass ich die Arbeit selbständig verfasst habe. Es wurden keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Zitate sind als solche gekennzeichnet.

Siegen, 26. Januar 2006

Kurzfassung

Diese Arbeit untersucht die Anwendung der Principal Component Analysis auf Transferfunktionen. Hierfür wird initial eine Anzahl von Transferfunktionen für Datensätze aus einem ausgewählten Bereich und für ein bestimmtes Anwendungsgebiet manuell eingestellt. Der Prozess des Transferfunktionsdesigns ist losgelöst von einer speziellen Kenntnis der Domänen einer Transferfunktion (Intensität, Gradientenbetrag usw.). Aufgrund eines hohen Freiheitsgrades und des Fehlens von zielorientierten Prozessen ist das Design von Transferfunktionen schwierig.

Existierende Ansätze ermöglichen automatisches und teilautomatisches Design von Transferfunktionen. Diese können in bildbasierte und datenbasierte Techniken unterteilt werden. Um sich jedoch auf die anatomischen und funktionalen Strukturen zu konzentrieren, die für den Benutzer interessant sind, ist es nötig, anwendungsbasierte Methoden einzuführen. Für ein genau definiertes Anwendungsszenario ist es möglich, die Komplexität der Transferfunktionsgenerierung zu reduzieren, indem der Klassifizierungsprozess auf die interessanten Strukturen für eine bestimmte Untersuchung eingeschränkt wird.

Hierfür werden zunächst Transferfunktionen für eine initiale Sammlung von Volumendatensätzen, welche zu einem bestimmten klinischen Zweck aufgenommen wurden, manuell eingestellt. Eine einzelne Transferfunktion wird durch eine Anzahl von Parametern geometrischer Primitive (Rampen oder Trapeze) repräsentiert. Jede dieser individuell eingestellten Transferfunktionen kann als Punkte-Sample im (vieldimensionalen) Parameterraum des Transferfunktionsmodells betrachtet werden. Aus der Gruppe von Punkte-Samples im Parameterraum wird ein statistisches Shape Model erstellt, auf welches die Principal Component Analysis angewendet wird. Dadurch wird ein Transferfunktionsmodell einer höheren Ordnung aufgebaut, welches nur noch eine sehr eingeschränkte Anzahl an Parametern benötigt. Der Prozess des Einstellens einer Transferfunktion wird dadurch sehr einfach und intuitiv.

Zusammenfassung

Zu Beginn wird die Arbeit motiviert, und das Ziel wird formuliert. Es wird gezeigt, dass Ärzte sehr von 3D-Visualisierung profitieren können, falls bestimmte Voraussetzungen erfüllt sind. Die Reproduzierbarkeit des gesamten Visualisierungsprozesses, das Vorhandensein intuitiver Applikationen und der Geschwindigkeitsfaktor sind hierbei von besonderer Bedeutung.

Im Folgenden werden Computertomographie und Algorithmen zur direkten Volumenvisualisierung eingeführt. Computertomographie wird beschrieben, da sie eine sehr verbreitete Aufnahmetechnik zum Erfassen von Daten für medizinische Untersuchungen ist. Algorithmen für die direkte Volumenvisualierung wie Raycasting oder Shear-warp Faktorisierung werden verwendet, um aus den Datensätzen Bilder zu generieren.

Transferfunktionen sind entweder eindimensional oder mehrdimensional. Die Theorie für beide Klassen wird eingeführt, und die Vorteile mehrdimensionaler Transferfunktionen werden beschrieben. Es wird aufgezeigt, wie die Berücksichtigung des Gradientenbetrags zu einer klaren Identifizierung der Grenzen im Datensatz führt. Zum Design von Transferfunktionen sind Anwendungen, welche auf dem interaktiven Einstellen von Transferfunktionen basieren, sowohl im wissenschaftlichen als auch im kommerziellen Bereich am weitesten verbreitet. Verschiedene Ansätze zum automatischen und teilautomatischen Design von Transferfunktionen wurden in den letzten Jahren entwickelt. Dies sind entweder bildbasierte oder datenbasierte Techniken. Bildbasierte Techniken basieren auf einer Generierung von Bildern mit einer unterschiedlichen Einstellung von Parametern. Sie unterscheiden sich in der Art und Weise, wie die Parameter gewählt werden. Im Gegensatz dazu analysieren datenbasierte Techniken die Daten. Dadurch sind diese Verfahren unhabhängig von bildbezogenen Parametern wie zum Beispiel der Pixel-Auflösung oder der Blickrichtung.

Der entwickelte 1D-Transferfunktionseditor bietet eine Reihe von Funktionen für das manuelle Einstellen der Tranferfunktion. Die Implementierung als OpenInventor Knoten stellt ein hohe Flexibilität und Portabilität sicher. Trapeze oder Rampen sind die geometrischen Primitive, die für die Definition einer Transferfunktion verwendet werden können. Sobald diese hinzugefügt sind, können sie auf sehr intuitive Weise manuell positioniert und eingestellt werden, da die Elemente der Primitive (Eckpunkte, Linien) als OpenInventor Dragger implementiert sind.

Zur Farbgebung für die verschiedenen Gewebe sind zwei verschiedene Möglichkeiten implementiert. Zum einen besteht die Möglichkeit, einem Primitiv verschiedene Farben zuzuweisen. Diese Farben sind dann an das jeweilige Primitiv

gebunden. Darüber hinaus können Farben verwendet werden, welche in einer globalen Colorbar definiert sind. Die endgütigen Farben, die für die Bildberechnung benutzt werden, werden in einer separaten Colorbar angezeigt. Alle relevanten Informationen einer Transferfunktion sind in einer parametrischen Beschreibung gespeichert. Jeder Knoten, der eine Verbindung zu dieser Beschreibung hat, kann relevante Informationen extrahieren.

Um die einzelnen Schritte der Principal Component Analysis nachvollziehen zu können, sind einige mathematischen Grundlagen erforderlich. Im Rahmen der Statistik sind Mittelwerte und Varianzen von Daten wichtige Voraussetzungen. Für multivariate Datensätze können Kovarianzen berechnet werden, die zu einer Kovarianzmatrix zusammengefügt werden können. Zusätzlich ist eine gewisse Kenntnis der Matrixalgebra erforderlich. Insbesondere die Berechnung von Eigenwerten und den dazugehörigen Eigenvektoren einer Kovarianzmatrix ist von Bedeutung.

Die Principal Component Analysis ist ebenfalls in einem OpenInventor Knoten implementiert. Eine intuitive Benutzeroberfläche erlaubt das Sammeln von Eingangsdaten durch eine Verbindung zur parametrischen Beschreibung der manuell definierten Transferfunktionen. Diese Eingangsdaten sind die Parametrisierungen der geometrischen Primitive der Transferfunktionen für verschiedene Datensätze. Diese Datensätze sind für eine bestimmte klinische Untersuchung aufgenommen. Die gesammelten Parameter repräsentieren eine Punktewolke in einem vieldimensionalen Raum. Principal Component Analysis wird dazu verwendet, die wichtigsten Achsen dieser Punktewolke zu bestimmen. Da die Eingangsdaten eine hohe Redundanz aufweisen, ist es möglich, die Freiheitsgrade durch ein Transferfunktionsmodell einer höheren Ordnung deutlich zu reduzieren.

Um die Vorzüge der präsentierten Herangehensweise zu demonstrieren, werden 14 CTA Datensätze verwendet. Das klinische Szenario ist die Visualisierung von Aneurysmen. Für zwölf dieser Datensätze werden die Transferfunktionen manuell eingestellt und für die PCA Berechnungen gespeichert. Das Ergebnis zeigt, dass ein Slider ausreicht, um Ausgangstransferfunktionen zu generieren, welche zu sehr aussagekräftigen Visualisierungen führen. Für zwei unabhängige Datensätze werden Bilder zu verschiedenen Ausgangstransferfunktionen präsentiert. All diese Bilder liefern eine gute Visualisierung der Aneurysmen mit leichten aber sichtbaren Unterschieden zwischen den Sliderpositionen.

Bibliography

[1] H. Anton. *Elementary Linear Algebra*. Wiley, New York, 8th edition, 2000.

[2] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. *ACM Symp. on Vol. Vis.*, 1994.

[3] R.B. Cattell. The Scree Test for the Number of Factors. *Multivariate Behavioral Research*, 1:245–276, 1966.

[4] R. Gordon, R. Bender, and G.T. Herman. Algebraic Reconstruction Techniques (ART) for Three-Dimensional Electron Microscopy and X-ray Photography. *J. Theor. Biol.*, 29:471–481, 1970.

[5] W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer-Verlag, Berlin Heidelberg, 2003.

[6] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of Transfer Functions with Stochastic Search Techniques. In *Proc. IEEE Visualization*, 1996.

[7] G.T. Herman. *Image Reconstruction from Projections*. Academic Press Inc., London, 1980.

[8] F. Vega Higuera, P. Hastreiter, B. Tomandl, C. Nimsky, and G. Greiner. 3D Visualization of Intracranial Aneurysms with Multidimensional Transfer Functions. *Bild Verarbeitung für die Medizin (BVM)*, 2003.

[9] F. Vega Higuera, N. Sauber, B. Tomandl, C. Nimsky, G. Greiner, and P. Hastreiter. Automatic Adjustment of Bidimensional Transfer Functions for Direct Volume Visualization of Intracranial Aneurysms. In *Proc. SPIE Medical Imaging*, 2004.

[10] G.N. Hounsfield. Computerized Traverse Axial Scanning (Tomography), Part I. Description of System. *Br. J. Radiol.*, 46:1016–1022, 1973.

[11] Barco Medical Imaging. available online at http://www.barco.com/medical/images/voxar/hr/MIP-meningioma.jpeg.

[12] Imaginis. Spiral CT and Helical CT. available online at http://imaginis.com/ct-scan/spiral.asp.

[13] S. Jackson and R. Thomas. *Cross-Sectional Imaging Made Easy.* Churchill Livingstone, 2004.

[14] I.T. Jolliffe. *Principal Components Analysis.* Springer-Verlag, New York, 2nd edition, 2002.

[15] L. Joskowicz and R.H. Taylor. Computers in Imaging and Guided Surgery. *Computers in Science and Engineering*, 3(5):65–72, 2001.

[16] H.F. Kaiser. The Application of Electronic Computers to Factor Analysis. *Educational and Psychological Measurement*, 20:141–151, 1960.

[17] J. Kajiya and B. Von Herzen. Ray Tracing Volume Densities. In *Proc. SIGGRAPH*, 1984.

[18] W.A. Kalender. *Computed Tomography.* Publicis MCD Verlag, Munich, 2000.

[19] R.A. Ketcham and W.D. Carlson. Acquisition, Optimization and Interpretation of X-ray Computed Tomographic Imagery: Applications to the Geosciences. *Computers & Geosciences*, 27:381–400, 2001.

[20] G. Kindlmann. Transfer Functions in Direct Volume Rendering: Design, Interface, Interaction. *SIGGRAPH 2002 Course Notes*, 2002.

[21] G. Kindlmann and J. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. *IEEE Symposium on Volume Visualization*, pages 79–86, 1998.

[22] G. Kindlmann and T. Möller R. Whitaker, T. Tasdizen. Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications. In *Proc. IEEE Visualization*, pages 513–520, 2003.

[23] J. Kniss, G. Kindlmann, and C. Hansen. Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets. *IEEE Visualization*, pages 255–262, 2001.

[24] J. Kniss, G. Kindlmann, and C. Hansen. Multi-Dimensional Transfer Functions for Interactive Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):270–285, 2002.

[25] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. *Comp. Graphics*, 28(4), 1994.

[26] D.C. Lay. *Linear Algebra and Its Applications.* Addison-Wesley, New York, 2000.

[27] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics & Applications*, 8(5):29–37, 1988.

[28] W. Lorenson and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Comp. Graphics*, 21(4):163–169, 1987.

[29] J. Marks, B. Andalman, P. Beardsley, and H. Pfister. Design Galleries: A General Approach for Setting Parameters for Computer Graphics and Animation. In *Proc. SIGGRAPH*, 1997.

[30] T. Möller and J. Gein. RadBuilder - Getting Started Guide Rev. 1.0, 2005.

[31] Newmat. The Newmat C++ Matrix Library Website. available online at http://www.robertnz.net/nm_intro.htm.

[32] National Library of Medicine. The Visible Human Project. available online at http://www.nlm.nih.gov/research/visible/visible_human.html.

[33] OpenGL. The Official OpenGL Website. available online at http://www.opengl.org.

[34] H. Pfister. *The Visualization Handbook*, chapter Hardware-Accelerated Volume Rendering, pages 229–258. Elsevier, 2005.

[35] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L.S. Avila, K. Martin, R. Machiraju, and J. Lee. The Transfer Function Bake-Off. *IEEE Computer Graphics & Applications*, 21(3):16–22, 2001.

[36] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition, 2002.

[37] C. Rezk-Salama. *Volume Rendering Techniques for General Purpose Graphics Hardware*. Phd thesis, University of Erlangen-Nuremberg, 2002.

[38] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2000.

[39] C. Rezk-Salama and M. Scheuering. Multitexturbasierte Volumenvisualisierung in der Medizin. *Bildverarbeitung in der Medizin: Algorithmen, Systeme, Anwendungen*, 2001.

[40] J. Shlens. A Tutorial on Principal Component Analysis, Version 2. available online at www.snl.salk.edu/˜shlens/pub/notes/pca.pdf, 2005.

[41] L.I. Smith. A Tutorial on Principal Components Analysis. available online at http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002.

[42] S.Roettger, S.Guthe, D. Weiskopf, T. Ertl, and W.Strasser. Smart Hardware-Accelerated Volume Rendering. *Eurographics/IEEE TCVG Symp. on Visualization*, 2003.

[43] Dalhousie University. Physics of Biological and Medical Technology. available online at www.physics.dal.ca/files/15_-_Computed_Tomography_I.ppt.

[44] J. Wernecke. *The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Addison-Wesley, 1994.

[45] L. Westover. Interactive Volume Rendering. *Chapel Hill Volume Visualization Workshop*, 28(4), 1989.

[46] Computergraphics TU Wien. Direct Volume Renderer. available online at http://www.cg.tuwien.ac.at/courses/Visualisierung/2004-2005/Beispiel1/BauchingerMaquil/.

[47] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA Volume Splatting. In *Proc. IEEE Visualization*, 2001.