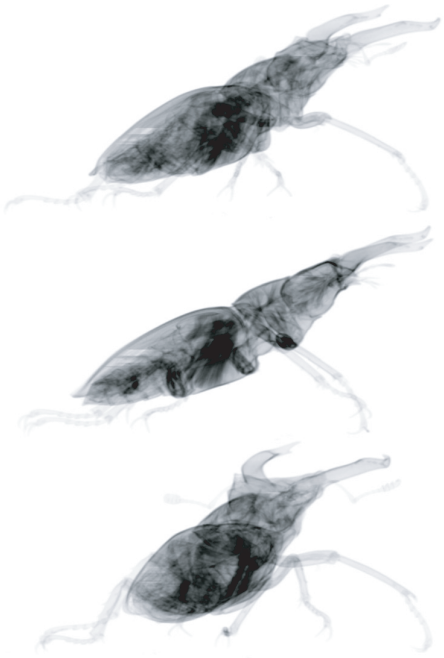
Peter Rautek

D^2VR
High-Quality Volume Rendering of Projection-based Volumetric Data

Master's Thesis



supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Dipl.-Inform. Dr.techn. Sören Grimm

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Abstract

Volume rendering techniques are conventionally classified into two categories represented by direct and indirect methods. Indirect methods require to transform the initial volumetric model into an intermediate geometrical model in order to efficiently visualize it. In contrast, direct volume-rendering (DVR) methods can directly process the volumetric data. Modern 3D scanning technologies, like CT or MRI, usually provide data as a set of samples on rectilinear grid points, which are computed from the measured projections by discrete tomographic reconstruction. Therefore the set of these reconstructed samples can already be considered as an intermediate volume representation. In this diploma thesis a new paradigm for *direct* direct volume rendering (D^2VR) is introduced, which does not even require a rectilinear grid, since it is based on an immediate processing of the measured projections. Arbitrary samples for ray casting are reconstructed from the projections by using the Filtered Back-Projection algorithm. The method presented in this thesis removes an unnecessary and lossy resampling step from the classical volume rendering pipeline. Thus, it provides much higher accuracy than traditional grid-based resampling techniques do. Furthermore a novel high-quality gradient estimation scheme, which is also based on the Filtered Back-Projection algorithm is presented. Finally we introduce a hierarchical space partitioning approach for projection-based volumetric data, which is used to accelerate D^2VR.

Kurzfassung

Volumenvisualisierungs-Techniken werden normalerweise in direkte und indirekte Methoden unterteilt. Indirekte Methoden verwenden zur effizienten Visualisierung eine geometrisch definierte Zwischenrepräsentation der volumetrischen Funktion. Im Gegensatz dazu, werden die volumetrischen Daten von direkten Volumenvisualisierungs-Methoden (*Direct Volume Rendering - DVR*) unmittelbar verarbeitet. Moderne 3D Scanner Verfahren, wie zum Beispiel CT oder MRI, liefern eine Menge von Daten, die normalerweise auf einem rektilinearen Gitter angeordnet sind. Diese Daten werden durch diskrete tomographische Rekonstruktion aus den ursprünglich gemessenen Daten berechnet. Die Daten auf rektilinearen Gittern sind demzufolge bereits eine Zwischenrepräsentation der volumetrischen Funktion. In dieser Diplomarbeit wird ein neues Paradigma für *direkte* direkte Volumenvisualisierung vorgestellt (*Direct Direct Volume Rendering - $D^2\,VR$*). D^2VR basiert auf der direkten Visualisierung der projektionsbasierten volumetrischen Funktion. Zur Visualisierung der gemessenen Daten wird auf die Erstellung jeglicher Zwischenrepräsentation der volumetrischen Funktion verzichtet. Die Rekonstruktion an beliebigen Positionen erfolgt mittels des Algorithmus der gefilterten Rückprojektion (*Filtered Back-Projection*). Die in dieser Diplomarbeit präsentierte Methode vermeidet einen unnötigen Abtastschritt der volumetrischen Funktion, der in der traditionellen Volumenvisualisierungs-Pipeline vorhanden ist. Deshalb wird mit der präsentierten Methode eine höhere Genauigkeit als bei gitterbasierten Methoden erzielt. Außerdem wird ein neues hoch qualitatives Gradienten-Schätzverfahren präsentiert, welches ebenfalls auf der gefilterten Rückprojektion basiert. Schließlich wird ein hierarchischer Raumunterteilungs-Ansatz für projektionsbasierte volumetrische Daten vorgestellt, der zur Beschleunigung von D^2VR verwendet wird.

Contents

Chapter 1

Introduction

Everything should be made as simple as possible, but not simpler.

Albert Einstein

Modern 3D scanning technologies, like Computed Tomography (CT) or Magnetic Resonance Imaging (MRI), usually provide data values on rectilinear grid points. These data values are computed from measured projections by discrete tomographic reconstruction [19, 6]. The set of the reconstructed data values (or samples) can be interpreted as a discrete representation of the underlying continuous phenomenon. In order to authentically visualize the original continuous signal, it has to be accurately reconstructed from the discrete samples. In the traditional approach such a signal reconstruction is differentiated from discrete tomographic reconstruction. From a signal-processing point of view, the original signal can be perfectly reconstructed from discrete samples if it is band-limited and the sampling frequency is above the Nyquist limit [16]. Theoretically the perfect continuous reconstruction is obtained by convolving the discrete volume representation by the *sinc* function. The *sinc* function is considered to be the best reconstruction kernel, since it represents an ideal low-pass filter. In practice, however, it is difficult to convolve a discrete signal with the *sinc* kernel, because of its infinite support. Therefore,

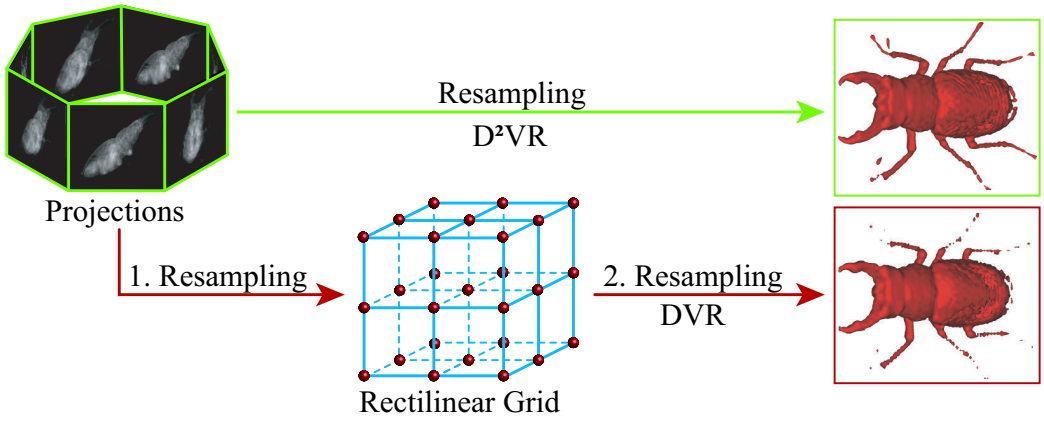


Figure 1.1: Data processing work flow of projection- and grid-based volume rendering. The red line corresponds to the traditional volume rendering pipeline. It requires two resampling steps in order to visualize the data. First an intermediate grid is resampled and then this grid is resampled again for rendering. The green line corresponds to the projection-based volume rendering pipeline; the lossy first resampling step is avoided.

practical reconstruction filters either approximate it or truncate it with an appropriate windowing function [11, 20]. Moreover, real-world signals can hardly be considered band-limited. As a consequence, practical resampling results in a loss of information. Figure 1.1 shows the signal-processing approach of the traditional volume rendering pipeline (follow the red line). The first step of the traditional approach is the discrete tomographic reconstruction of a rectilinear volume representation from the measured projections. Although there exist different algorithms for tomographic reconstruction, one of the most popular techniques is the Filtered Back-Projection algorithm. It first performs high-pass filtering on the measured projections and afterwards the samples at rectilinear grid points are computed by back-projecting the filtered signals. As the projections are acquired by measuring accumulated attenuation by a limited number of sensors, they are actually available as discrete representations of continuous projection functions. Therefore, high-pass filtering is performed in discrete frequency domain, so the result is also a discrete function. In the back-projection phase, however, the rectilinear grid points are not necessarily projected exactly onto the discrete samples

of the filtered projections. Therefore for back-projection resampling is necessary, which results in the *first* loss of information in the pipeline. The obtained rectilinear volume can be visualized by different rendering techniques. Using indirect methods, like the classical Marching Cubes algorithm [9], an intermediate geometrical model of an iso-surface is constructed from the volumetric model. This geometrical model is then interactively rendered by, for example, conventional graphics hardware. In contrast, Direct Volume Rendering (DVR) approaches, like ray casting [7] or splatting [23, 24] directly render the volumetric model without any intermediate representation. In both cases an interpolation technique is applied to define data values between the rectilinear grid points. In other words, a resampling of the discrete volume representation is required. This resampling results in the *second* loss of information in the traditional pipeline.

In order to minimize the loss of information we propose to modify the traditional volume-rendering pipeline by simply removing an unnecessary resampling step (follow the green line in Figure 1.1). To render the underlying continuous phenomenon, data samples at arbitrary sample points need to be defined, and for shading computation the corresponding gradients need to be determined. As it will be shown, both tasks can be solved using directly the filtered projections. This eventually leads to an alternative projection-based volume representation. Thus, there is no need to compute samples at regular grid points by discrete tomographic reconstruction, and as a consequence one resampling step (see Figure 1.1) is unnecessary. Traditional direct volume-rendering methods rely on such an intermediate grid representation, so in this sense they are in fact indirect. In contrast to that, DVR *directly* from the measured raw data is presented in this thesis. To distinguish from the common DVR the novel approach is referred to as D^2VR (pronunciation: [diskwerd vi ar]). In Chapter 2 previous work related to discrete tomographic reconstruction and volume resampling is reviewed. In Chapter 3 the mathematical background of Computed Tomography is given. In Section 3.1 the Fourier slice theorem is described. Filtered Back-Projection is based on the Fourier slice theorem and derived afterwards in Section 3.2. In Section 3.2.2 a brief overview on filters is given. Finally, in Section 3.2.3 a novel gradient

estimation scheme for projection-based volumetric data based on the Filtered Back-Projection is described. Chapter 4 introduces D^2VR, a volume rendering approach for projection-based volumetric data. It is based on raycasting which is described in detail in Section 4.1. Furthermore, in Section 4.2.2, an acceleration data structure, based on the principle of hierarchical space partitioning, for projection-based volumetric data is presented. In Chapter 5 details on the implementation are given. In Section 5.1 the implemented Computed Tomography scanner simulation is described and in Section 5.2 the implementation of D^2VR is summarized. Chapter 6 reports the results of D^2VR. An error analysis as well as a visual comparison of grid-based and projection-based volume rendering is given. Finally, in Chapter 7 the contents of this diploma thesis are summarized.

Chapter 2

Related Work

Advice is what we ask for
when we already know the
answer but wish we didn't.

Erica Jong

In most of the practical volume-rendering applications, especially in 3D medical imaging, the input data is usually generated from measured projections by using tomographic reconstruction [19, 6, 15]. The set of projections is referred to as the Radon transform of the original signal. Therefore the tomographic reconstruction is, in fact, the inversion of the Radon transform. The inversion can be performed by using the classical Filtered Back-Projection [3] algorithm, which is based on the Fourier projection-slice theorem [6, 10]. Although there exist alternative tomographic reconstruction techniques like algebraic or statistical ones, Filtered Back-Projection is still the most popular method used in commercial CT scanners.

The output of tomographic reconstruction is a discrete (or sampled) representation of the underlying continuous phenomenon. The samples are conventionally generated on rectilinear grid points. The rectilinear grid has several advantages. For example, the sampled signal can be represented by 3D arrays, implicitly storing the locations of the samples. Furthermore, the neighborhood of a certain sample can be efficiently addressed, which is im-

portant for many volume-processing or volume-rendering algorithms.

Nevertheless, in order to render the underlying continuous 3D function, data values need to be defined also between the rectilinear grid points. The *sinc* kernel as ideal reconstruction filter is impractical because of its infinite extent. In practice it is approximated by filters of finite support [11, 20]. Generally, the wider the support of the reconstruction filter, the higher the quality of the reconstruction. On the other hand, the wider the support of the filter, the higher the computational cost of a spatial-domain convolution. Therefore several researchers analyzed different reconstruction filters, both in terms of accuracy and computational cost [12, 11, 13, 14]. As the practical filters only approximate the ideal low-pass filter they result in either aliasing or smoothing [11], which can be interpreted as a loss of information. For frequent resampling tasks, like rotation, or upsampling, frequency-domain techniques can be alternatively applied [8, 1, 4, 5, 21, 22]. In frequency domain, it is exploited that a computationally expensive spatial-domain convolution is replaced by a simple multiplication. Although the frequency-domain resampling methods generally provide higher accuracy than spatial-domain methods do, they assume that the new samples to be computed are also located at regular grid points.

In order to avoid a lossy resampling step in the traditional volume-rendering pipeline, we directly use the tomographic inversion in order to reconstruct the underlying function at arbitrary sample positions. Therefore we do not generate an intermediate rectilinear volume representation, but we directly process the filtered projections as an alternative volume representation. Using this gridless or projection-based volume-rendering approach as a new paradigm, the same accuracy can be ensured at all the sample positions. In contrast, using the traditional grid-based approach, accurate samples are available only at the grid points, while the accuracy of intermediate samples depends on the quality of the applied imperfect reconstruction filter.

Chapter 3

Mathematical Preliminaries

If I had eight hours to chop down a tree, I'd spend six hours sharpening my ax.

Abraham Lincoln

In this chapter the mathematical background, in order to understand the principles of a Computed Tomography scanner, is described. In Chapter 4 all formulas which are needed to implement D^2VR are briefly reviewed again and references to the mathematical derivation in this chapter are given.

In Section 3.1 the Fourier slice theorem is explained. It is the basis of the Filtered Back-Projection which is described afterwards in section 3.2. The Filtered Back-Projection is derived in its continuous version as well as in its discretized version. It can be used to reconstruct the density function, see Section 3.2.1, as well as to estimate the derivatives of the density function, see Section 3.2.3. As the filtering is inherent in the Filtered Back-Projection the accuracy of the discrete Filtered Back-Projection mainly depends on the use of appropriate filters. Therefore in Section 3.2.2 a brief survey of filters is given in order to improve the accuracy of the discrete Filtered Back-Projection.

3.1 Fourier Slice Theorem

The Fourier slice theorem was introduced by Charles Fourier (1772-1837). Originally it has nothing to do with a Computed Tomography scanner, as the first Computed Tomography scanner was presented in the year 1972. However, it is very illustrative to explain the Fourier slice theorem by means of a Computed Tomography scanner.

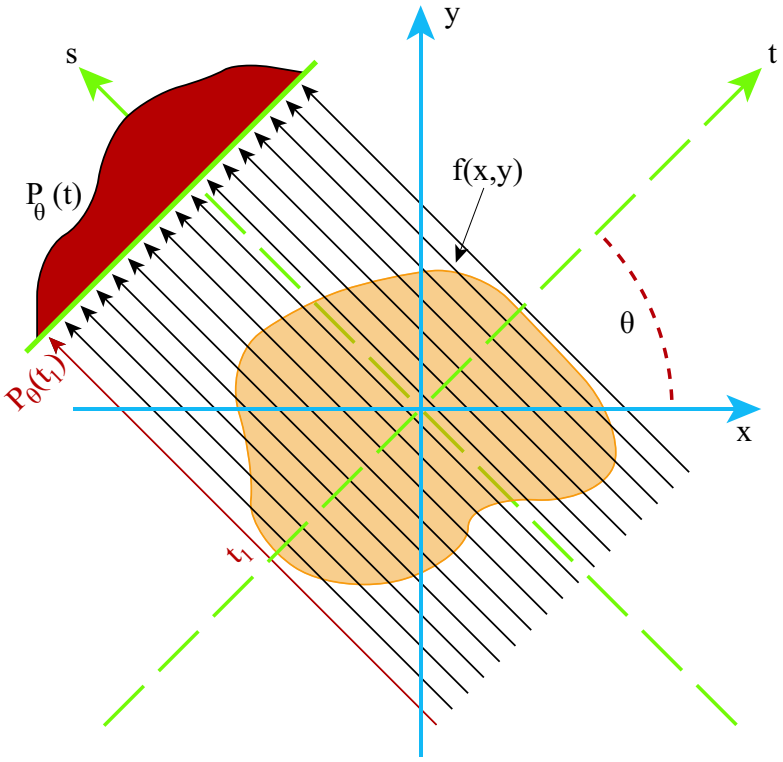


Figure 3.1: Parallel projection for a specific angle θ. All values on the line with constant $t = t_1$ project onto the same point $P_\theta(t_1)$. $P_\theta(t)$ is the line integral along a line with direction $(-\sin\theta, \cos\theta)$ and with a normal distance t to the origin.

Consider a Computed Tomography scanner measuring the absorption of x-rays cast through an object. Let us assume, that the measurements are taken while the x-ray source and the detector are moving on parallel lines on opposite sides of the object. These measurements are combined in a

projection $P_\theta(t)$ where θ is the angle at which the measurements were taken. This process is known as parallel projection and is shown in Figure 3.1. One measurement $P_\theta(t_1)$ for a specific t_1 corresponds to the line integral of the absorption of the x-rays along a line with direction $(-\sin(\theta), \cos(\theta))$ and a normal-distance t_1 to the origin. Let the object's density function be denoted $f(x, y)$, then, a parallel projection can be written as

$$P_\theta(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x\cos\theta + y\sin\theta - t)\ dx\ dy \tag{3.1}$$

where δ denotes the delta function. The relations between the (x, y)- and the (t, s)- coordinate system, see Figure 3.1, are given by:

$$\begin{aligned} x &= t\cos\theta - s\sin\theta \\ y &= t\sin\theta + s\cos\theta \end{aligned} \tag{3.2}$$

This leads to an alternative representation of $P_\theta(t)$:

$$P_\theta(t) = \int_{-\infty}^{\infty} f(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)ds \tag{3.3}$$

A parallel projection is also known as the Radon transform of a density function.

The Fourier slice theorem states that:

> The one-dimensional Fourier transform of the Radon transform of a density function $f(x, y)$ taken at angle θ is equal to a line of the two-dimensional Fourier transform, $F(u, v)$, subtending an angle θ with the u-axis.

In other words, the one-dimensional Fourier transform of P_θ gives the values of $F(u, v)$ along line L_θ, see Figure 3.2.
The one-dimensional Fourier transform of P_θ is given by:

$$S_\theta(w) = \int_{-\infty}^{\infty} P_\theta(t)e^{-j2\pi wt}\ dt \tag{3.4}$$

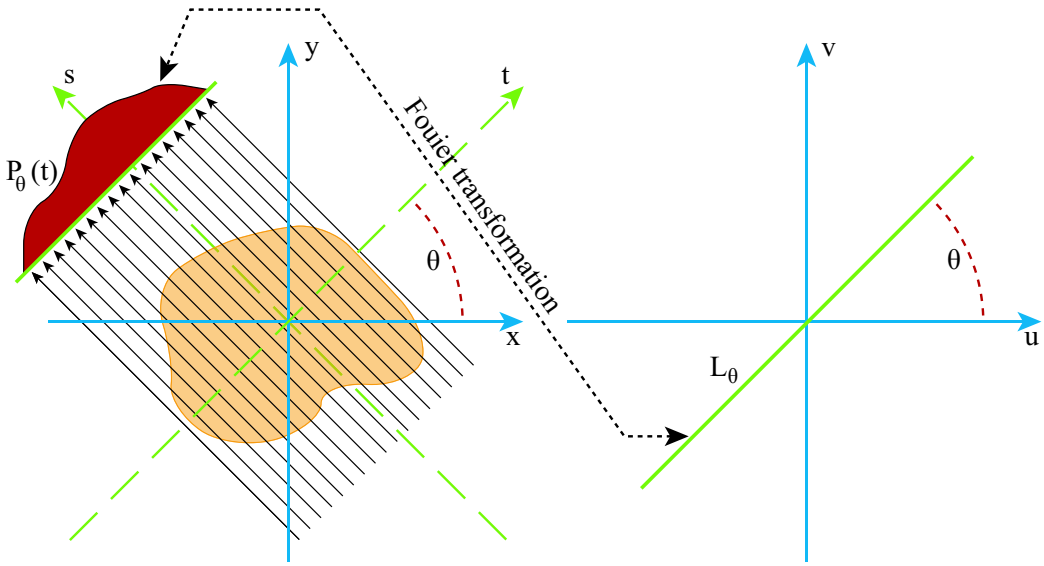


Figure 3.2: Fourier slice theorem

Substituting Equation 3.3 into Equation 3.4 leads to

$$S_\theta(w) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\ ds \right] e^{-j2\pi wt}\ dt \qquad (3.5)$$

In order to proof the Fourier slice theorem we have to change the coordinate system. Changing variables of a double integral can be done by using the following relation:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\ dx\ dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x(t,s),y(t,s))J(t,s)\ dt\ ds \qquad (3.6)$$

where $J(t,s)$ denotes the determinant of the Jacobian matrix (i.e. the Jacobian determinant) of the transformation, given by

$$J(t,s) = \begin{vmatrix} \frac{\partial x}{\partial t} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial t} & \frac{\partial y}{\partial s} \end{vmatrix} \qquad (3.7)$$

According to the relations in Equation 3.2 the Jacobian determinant in our

case, is given by

$$J(t,s) = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix} = \cos^2\theta + \sin^2\theta = 1 \tag{3.8}$$

We can therefore rewrite Equation 3.5 using Cartesian coordinates:

$$S_\theta(w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi w(x\cos\theta + y\sin\theta)}\,dx\,dy \tag{3.9}$$

The right-hand side of Equation 3.9 is the definition of the two-dimensional Fourier transform of $f(x,y)$ at a spatial frequency of $(u = w\cos\theta, v = w\sin\theta)$. It can be seen that:

$$S_\theta(w) = F(w\cos\theta, w\sin\theta) = F(u,v) \tag{3.10}$$

This Equation is the proof of the Fourier slice theorem and is therefore the basis for Computed Tomography.

3.2 Filtered Back-Projection

For simplicity we illustrate the Filtered Back-Projection in 2D. Once, we are able to reconstruct the two-dimensional density function $f(x,y)$, and therefore whole slices of the density function, it is straightforward to extend the Filtered Back-Projection to three dimensions. In Section 3.2.1 and Section 3.2.2 it is shown how to achieve accurate resampling using the Filtered Back-Projection. Commonly the Filtered Back-Projection is used to resample the density function on a rectilinear grid. Derivatives of the underlying density function are then approximated from the rectilinear grid. However, the derivatives can also be computed directly from the filtered projections using the Filtered Back-Projection algorithm, see Section 3.2.3.

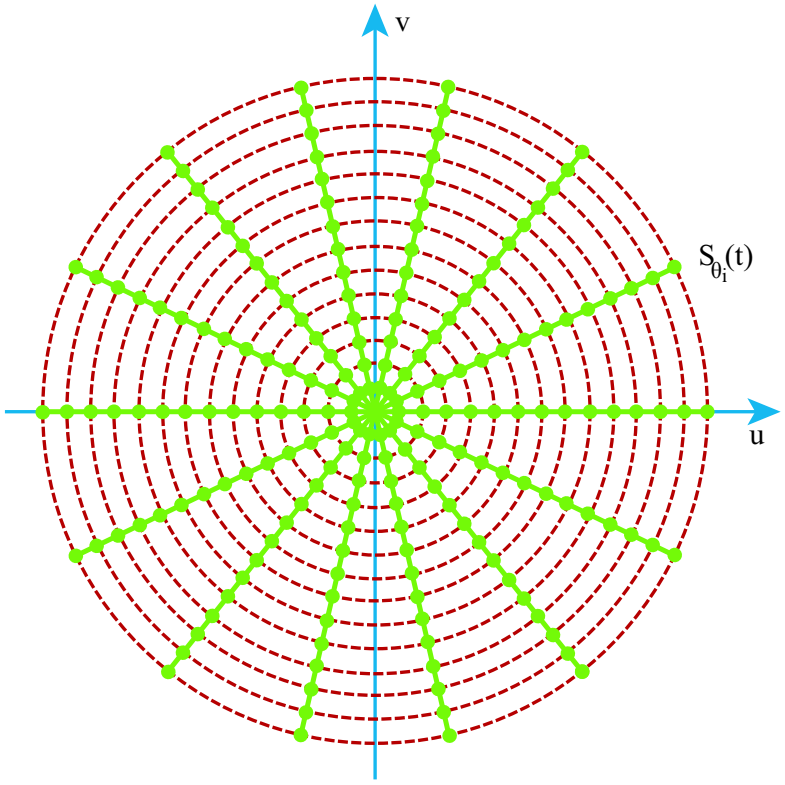


Figure 3.3: Alignment of one-dimensional Fourier transformed parallel projections in the two dimensional frequency domain.

3.2.1 Data Reconstruction

Let us again think of our Computed Tomography scanner: We want to reconstruct the object's density function $f(x, y)$. We take parallel projections at different angles. According to the Fourier slice theorem the one dimensional Fourier transform of these projections is equal to radial lines of the two-dimensional Fourier transformation of the underlying density function. In Figure 3.3 the one-dimensional Fourier transformed projections can be seen. They are arranged as radial lines in the two-dimensional frequency domain. Instead of the Cartesian coordinate representation of the two-dimensional Fourier Transformation we use in the following the polar coordinate repre-

sentation which is defined as:

$$f(x,y) = \int_0^{2\pi} \int_0^{\infty} F(w,\theta) e^{j2\pi w(x\cos\theta + y\sin\theta)} w \; dw \; d\theta \tag{3.11}$$

If we had an infinite number of continuous projections, taken around the object, we would also have an infinite number of radial lines in frequency domain, see Figure 3.3. Therefore we could perfectly reconstruct the underlying density function by simply applying the two-dimensional inverse Fourier transform. In practice the number of projections is limited. Therefore we have to find an appropriate approximation for Equation 3.11. If we consider θ from 0 to π, the integral can be split as follows:

$$\begin{aligned} f(x,y) = \int_0^{\pi} \int_0^{\infty} F(w,\theta) e^{j2\pi w(x\cos\theta + y\sin\theta)} w \; dw \; d\theta + \\ + \int_0^{\pi} \int_0^{\infty} F(w,\theta+\pi) e^{j2\pi w(x\cos(\theta+\pi) + y\sin(\theta+\pi))} w \; dw \; d\theta \end{aligned} \tag{3.12}$$

Since $F(w,\theta+\pi) = F(-w,\theta)$, the above expression can be written as:

$$f(x,y) = \int_0^{\pi} \left[\int_{-\infty}^{\infty} F(w,\theta) \, |w| \, e^{j2\pi wt} \; dw \right] \; d\theta \tag{3.13}$$

where $t = x\cos\theta + y\sin\theta$. By substituting $S_\theta(w)$ for the two-dimensional Fourier transform $F(w,\theta)$ the above integral can be expressed as:

$$f(x,y) = \int_0^{\pi} \int_{-\infty}^{\infty} S_\theta(w) \, |w| \, e^{j2\pi wt} \; dw \; d\theta \tag{3.14}$$

According to the Fourier slice theorem $S_\theta(w)$ is the Fourier transform of $P_\theta(t)$. Let us define:

$$Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(w) \, |w| \, e^{j2\pi wt} \; dw \tag{3.15}$$

which is the inverse Fourier transform of $S_\theta(w) \cdot |w|$. As multiplication in frequency domain corresponds to a convolution in spatial domain, according to Equation 3.15, $Q_\theta(t)$ is obtained by high-pass filtering the measured projection $P_\theta(t)$.

By substituting Equation 3.15 into Equation 3.14, we can derive

$$f(x,y) = \int_0^{\pi} Q_\theta(x\cos\theta + y\sin\theta)\ d\theta \tag{3.16}$$

In practice, as mentioned above the number of projections is limited and the integral in Equation 3.16 is approximated by

$$f(x,y) \approx \tilde{f}(x,y) = \frac{\pi}{K}\sum_{i=1}^{K} Q_{\theta_i}(x\cos\theta_i + y\sin\theta_i) \tag{3.17}$$

where Q_{θ_i} is the filtered representation of the projection P_{θ_i}. The P_{θ_i} are taken at the K angles θ_i $i = 1 \ldots K$. Thus, according to Equation 3.17 the density function can be reconstructed from a fixed number of projections. Equation 3.17 is the continuous version of the Filtered Back-Projection

The filtered projections Q_{θ_i} as well as the projections P_{θ_i} have been assumed to be continuous so far. However, in practice they are not. According to the sampling theorem the continuous function P_θ can be perfectly reconstructed if it is sampled above the Nyquist rate. If w_c is a frequency higher than the highest frequency component in each projection the projections are bandlimited and therefore can be sampled at intervals of

$$T = \frac{1}{2w_c} \tag{3.18}$$

without introducing any error. On the other hand if the projections are sampled with sampling intervals of T they cannot contain any frequency higher than

$$w_c = \frac{1}{2T} \tag{3.19}$$

In other words the discrete projections are already bandlimited. In order to get the filtered discrete projections $\hat{Q}_{\theta_i}$ we can apply the Discrete Fourier Transform (DFT) to the discrete projections $\hat{P}_{\theta_i}$ The DFT of $\hat{P}_{\theta_i}$ is obtained by approximation of the integral in the Fourier transform. It is given by:

$$\hat{S}_{\theta_i}(m\frac{2w_c}{N}) = T\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{P}_{\theta_i}(kT)e^{-j2\pi km\frac{2w_c}{N}} \tag{3.20}$$

The bandlimited and filtered projections Q_{θ_i} are according to Equation 3.15 given by

$$Q_{\theta_i}(t) = \int_{-w_c}^{w_c} S_{\theta_i}(w)\,|w|\,e^{j2\pi wt}\,dw \tag{3.21}$$

The integral in Equation 3.21 is approximated by

$$\hat{Q}_{\theta_i}(t) \approx \frac{2w_c}{N} \sum_{m=-\frac{N}{2}}^{\frac{N}{2}} \hat{S}_{\theta_i}(m\frac{2w_c}{N}) \left| m\frac{2w_c}{N} \right| e^{j2\pi m\frac{2w_c}{N}t} \tag{3.22}$$

Substituting the continuous filtered projections in Equation 3.17 by the discrete filtered projections given in Equation 3.22 leads to the discrete version of the Filtered Back-Projection:

$$f(x,y) \approx \tilde{f}(x,y) = \frac{\pi}{K} \sum_{i=1}^{K} \hat{Q}_{\theta_i}(x\cos\theta_i + y\sin\theta_i) \tag{3.23}$$

Note, that the filtering of the projections is performed in the frequency domain. The filtered projections Q_θ are derived in Equation 3.15 by multiplying the Fourier transformed projections S_θ with $|w|$. In order to derive an alternative representation of Q_θ we start with explicitly denoting $|w|$ as a filter:

$$Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(w)H(w)e^{j2\pi wt}\,dw \tag{3.24}$$

where $H(w)$ is a filter and simply the bandlimited version of $|w|$:

$$H(w) = \begin{cases} |w| & |w| < w_c \\ 0 & \text{otherwise} \end{cases}$$

This filter is shown in Figure 3.4. The impulse response of this filter is given by the inverse Fourier transform of $H(w)$:

$$h(t) = \int_{-\infty}^{\infty} H(w)e^{j2\pi wt}\,dw \tag{3.25}$$

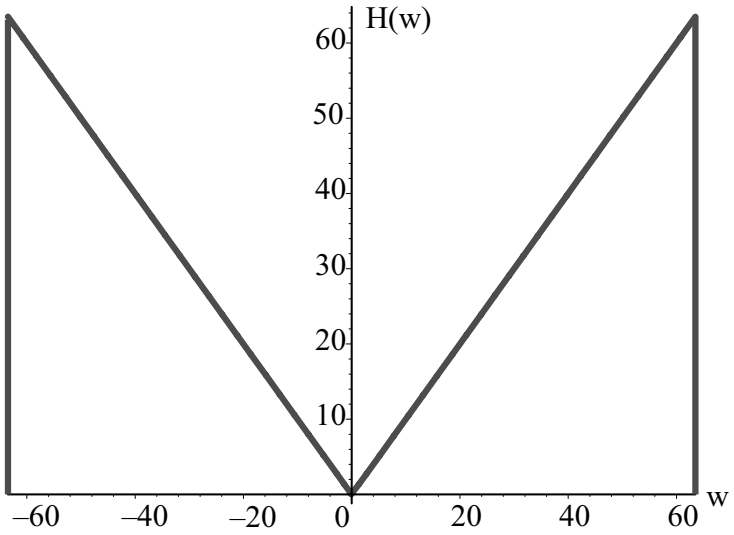


Figure 3.4: A ramp filter for a bandlimited function with a cutoff frequency $w_c = 64$.

The above integral can be evaluated analytically using Equation 3.18:

$$h(t) = \frac{1}{2T^2}\frac{\sin 2\pi\frac{t}{2T}}{2\pi\frac{t}{2T}} - \frac{1}{4T^2}\left(\frac{\sin \pi\frac{t}{2T}}{\pi\frac{t}{2T}}\right)^2 \tag{3.26}$$

Since the projections are measured with a sampling interval of T, $h(t)$ needs only to be known at the same sampling intervals and therefore simplifies to:

$$\hat{h}(nT) = \begin{cases} \frac{1}{4T^2} & n = 0 \\ 0 & n \text{ even} \\ -\frac{1}{n^2\pi^2T^2} & n \text{ odd} \end{cases}$$

Using the convolution theorem, Equation 3.24 can be written as

$$Q_\theta(t) = \int_{-\infty}^{\infty} P_\theta(\bar{t})h(t-\bar{t})\, d\bar{t} \tag{3.27}$$

Since the filtering is only performed at discrete samples of $Q_\theta(t)$ and due to

the finite extend, we can derive:

$$\hat{Q}_\theta(nT) = T \sum_{k=0}^{N-1} \hat{P}_\theta(kT)\hat{h}(nT - kT) \tag{3.28}$$

where $n = 0 \dots N-1$ and $\hat{h}$ is the discretized filter. It can be seen that the argument of $\hat{h}$, $nT - kT$, is in the range $-(N-1)T \dots (N-1)T$ and therefore the filter should be defined in this range. Due to the lack of any approximation in Equation 3.28 this leads us to a different, and more accurate result than Equation 3.22. While it is more accurate to use Equation 3.28 the computational cost is approximately the same, since both Equations can be implemented as a multiplication in frequency domain. For the frequency domain implementation one has to keep in mind, that the convolution in Equation 3.28 is aperiodic and therefore it leads to inter-period artifacts. However, these artifacts can be avoided by zero-padding the projections with a sufficient number of zeros. It can be shown, that for an implementation based on an FFT algorithm the projections need to be zero-padded so that they are $2N-1$ elements long, for further details see [2]. For the implementation based on the generally faster *base 2 FFT algorithm*, the projections need to be padded, so that each is $(2N-1)_2$ elements long, where $(i)_2$ denotes the smallest integer that is a power of 2 and is greater than i.

3.2.2 Filters

In practice the measured data is noisy. Therefore superior results can be achieved by smoothing the projections. This can for example be achieved by multiplying the filter $H(w)$ by a Hamming window. A Hamming window deemphasizes higher frequencies and therefore the noise is filtered. The formula for a generalized Hamming window is as follows:

$$W_\alpha(k) = \alpha + (1-\alpha)\cos(2\pi\frac{k}{N-1}) \tag{3.29}$$

where the parameter α controlls the deemphasizing of the higher frequencies. While a multiplication of $H(k)$ with a Hamming window $W_1(k)$ is equal

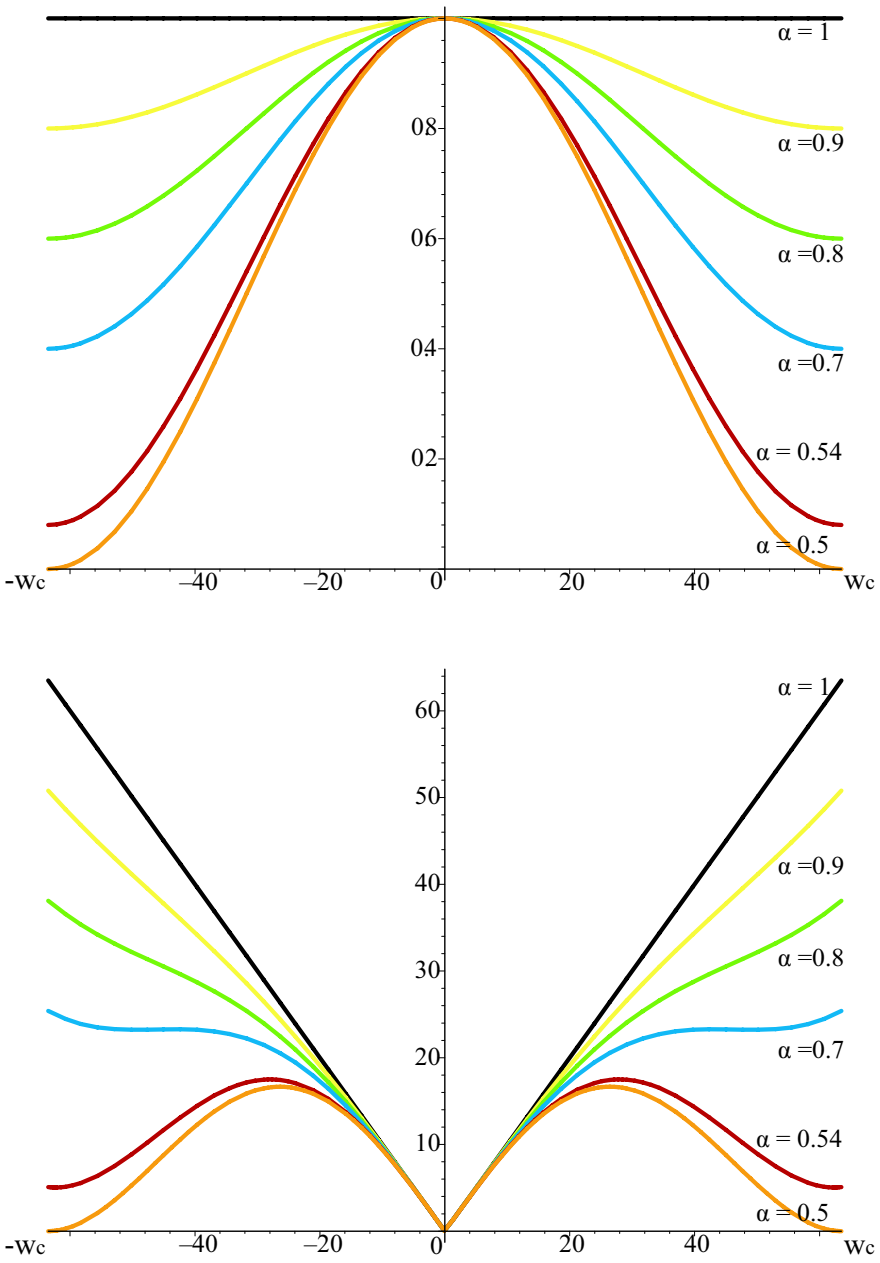


Figure 3.5: Generalized Hamming windows. Top: Hamming windows with different settings for the parameter α. The orange line ($\alpha = 0.5$) is the so called Hanning window; The red line ($\alpha = 0.54$) is the original Hamming window. Bottom: Multiplication of the Hamming windows with the filter $H(k) = |k|$ shown in Figure 3.4. All examples are for a filter in the range $-63 \ldots 63$.

to $H(k)$, a setting of $\alpha = 0.5$ leads to a deemphasizing such that $H(N - 1) \cdot W_{0.5}(N - 1) = 0$. Therefore, with a setting of $\alpha = 0.5$, the highest frequency if forced to zero. A generalized Hamming window with $\alpha = 0.5$ is called Hanning window. Figure 3.5 shows the multiplication of the filter $H(k) = |k|$ with different Hamming windows.

3.2.3 Derivative Estimation

Filtered Back-Projection is a practical approach to reconstruct the measured density function. For many volume processing algorithms the density as well as the derivatives of the underlying three-dimensional function need to be known. The traditional approach resamples the density function onto a rectilinear grid. This grid is used to estimate gradients. Commonly derivative estimation in rectilinear volumetric representations is done using a close neighborhood of samples. In contrast to that, using a projection based volumetric representation, derivatives can be estimated directly from the filtered projections using the Filtered Back-Projection.

For example the partial derivative $\tilde{f}_x$ according to variable x can be expressed by using Newton's difference quotient:

$$\begin{aligned} \tilde{f}_x = & \quad \tfrac{\partial \tilde{f}(x,y)}{\partial x} \quad = \\ & = \lim_{\Delta x \to 0} \tfrac{1}{\Delta x} (\tfrac{\pi}{K} (\sum_{i=1}^{K} Q_{\theta_i}(x \cos\theta_i + y \sin\theta_i) - \\ & \quad - \sum_{i=1}^{K} Q_{\theta_i}((x + \Delta x) \cos\theta_i + y \sin\theta_i))) \end{aligned} \tag{3.30}$$

Substituting $t_i := x \cos\theta_i + y \sin\theta_i$ we obtain:

$$\begin{aligned} \tilde{f}_x = & \lim_{\Delta x \to 0} \tfrac{1}{\Delta x} (\tfrac{\pi}{K} (\sum_{i=1}^{K} Q_{\theta_i}(t_i) - \sum_{i=1}^{K} Q_{\theta_i}(\Delta x \cos\theta_i + t_i))) = \\ & = \lim_{\Delta x \to 0} \tfrac{1}{\Delta x} (\tfrac{\pi}{K} \sum_{i=1}^{K} Q_{\theta_i}(t_i) - Q_{\theta_i}(\Delta x \cos\theta_i + t_i)) = \\ & = \tfrac{\pi}{K} \sum_{i=1}^{K} \lim_{\Delta x \to 0} \tfrac{1}{\Delta x} (Q_{\theta_i}(t_i) - Q_{\theta_i}(\Delta x \cos\theta_i + t_i)) \end{aligned} \tag{3.31}$$

The term

$$\lim_{\Delta x \to 0} \frac{1}{\Delta x} (Q_{\theta_i}(t_i) - Q_{\theta_i}(\Delta x \cos\theta_i + t_i)) \tag{3.32}$$

is the partial derivative of the projections Q_{θ_i}, but scaled with $\cos\theta_i$. We can therefore calculate the partial derivative $\tilde{f_x}$ directly as sum of scaled derivatives of the projection data. Analogously, taking the difference quotient with respect to y we obtain:

$$\tilde{f_y} = \frac{\partial \tilde{f}(x,y)}{\partial y} = \frac{\pi}{K} \sum_{i=1}^{K} \lim_{\Delta y \to 0} \frac{1}{\Delta y} (Q_{\theta_i}(t_i) - Q_{\theta_i}(\Delta y \sin\theta_i + t_i)) \tag{3.33}$$

It can be seen that applying Newton's difference quotient directly on the filtered projections is equivalent to applying Newton's difference quotient to the 2D density function $f(x, y)$. Moreover, any higher order derivative can be obtained by applying Newton's difference quotient multiple times.

Using Filtered Back-Projection for gradient estimation we expect higher accuracy than using the traditional gradient estimation schemes on the rectilinear grid. Consider central differences on the continuous reconstruction from a rectilinear representation. In order to calculate the gradient at an arbitrary sampling point six additional samples have to be interpolated. As interpolation usually causes loss of information, the introduced errors are accumulated in the estimated gradients. In contrast, using Filtered Back-Projection, the density values at the additional sample points are as accurate as the density values at the grid points. Therefore, no interpolation error is introduced.

3.3 Summary

In Section 3.1 we discussed the Fourier slice theorem. We defined a parallel projection as a combination of line integrals of parallel lines. According to the Fourier slice theorem a two-dimensional function can be perfectly reconstructed from an infinite number of parallel projections. In order to reconstruct the density function from a finite number of projections we explained the Filtered Back-Projection algorithm in Section 3.2.1. To achieve high accuracy for reconstruction, we briefly discussed the filtering of the discrete projections in Section 3.2.2. Finally, the estimation of derivatives

directly from the projections was shown, in Section 3.2.3. Taking all the parts together, we introduced a projection based volumetric data representation, which will be exploited for Volume Visualization in the following chapters.

Chapter 4

D^2VR

Science... never solves a problem without creating ten more.

George Bernard Shaw

In this chapter *direct* direct volume rendering (D^2VR) based on an image order rendering approach is presented. The image is computed by casting one ray for each pixel of the image plane through the scene (*Raycasting*). Along the ray resampling is performed at consecutive steps. The underlying 3D volumetric function needs to be reconstructed at these resampling locations. In case the data is given on a rectilinear grid the reconstructed function value is computed from a close neighborhood of samples as shown in Figure 4.1a. In contrast to that, for raycasting directly performed on the filtered projections the reconstructed function value is computed from the filtered projections at the corresponding positions, see Figure 4.1b. Furthermore, gradients at these resample locations need to be determined in order to perform shading. Each step of the raycasting pipeline is discussed in Section 4.1. Furthermore in Section 4.2 a hierarchical space partitioning data structure is presented in order to accelerate the volume rendering process.

4.1 Raycasting

Raycasting is an approach to approximate the volume rendering integral for every viewing ray. Raycasting can be split into four successive steps which are described in the following:

4.1.1 Reconstruction

The Filtered Back-Projection algorithm, as it is presented in Section 3.2 is conventionally used for discrete tomographic reconstruction in order to obtain a rectilinear representation of the original density function. This intermediate representation is then usually resampled by many volume visualization algorithms. However, it is not necessary to generate an intermediate rectilinear grid representation for this purpose. In fact the additional resampling step should be avoided, because each resampling step usually causes a loss of information.

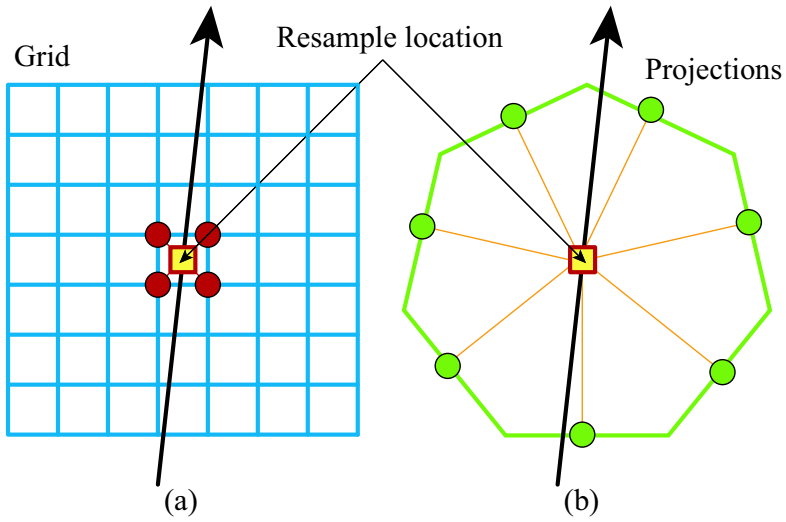


Figure 4.1: (a) Resampling along a ray on rectilinear volumetric data. (b) Resampling along a ray directly from the filtered projections

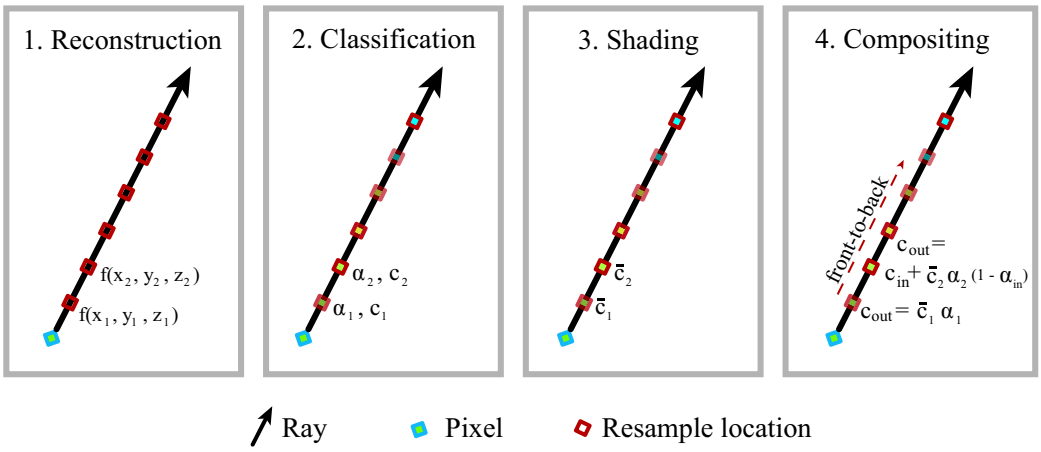


Figure 4.2: The raycasting pipeline: 1. The data reconstruction step evaluates the density function, 2. Classification leads to an opacity value α_i and to a color c_i, 3. The outcome of shading is a shaded color $\overline{c}_i$, 4. Compositing accumulates the values of all samples along a ray.

Projection-Based Reconstruction

The first step in the raycasting pipeline is the reconstruction of the density function, see Figure 4.2. The density function has to be evaluated along each ray at intervals of Δ_s (the object sample distance). As shown in Section 3.2.1 the density function can be reconstructed using the Filtered Back-Projection. The formula in Equation 3.23 can be considered as a resampling scheme to estimate the density value at an arbitrary sample point. In Section 3.2 we explained the Filtered Back-Projection for a single slice. Consider this slice as one out of many slices parallel to the x, y-plane. The projections $P_\theta(t, r)$ of these parallel slices are taken at discrete intervals $n\Delta_r$ where $n = 1 \ldots N$ and N is the number of parallel slices. In the following we use two-dimensional filtered discrete projections $\hat{Q}_{\theta_i}(t, r)$. They are obtained by filtering each line of the two-dimensional projections $P_\theta(t, r)$ separately. We can extend the Filtered Back-Projection to three-dimensions by interpolating between these parallel slices. The formula for resampling at an arbitrary position is given by:

$$\tilde{f}(x, y, z) = \frac{\pi}{K} \sum_{i=1}^{K} \hat{Q}_{\theta_i}(x \cos\theta_i + y \sin\theta_i, z) \tag{4.1}$$

where $\hat{Q}_{\theta_i}$ are the discrete filtered two-dimensional projections and $x\cos\theta_i + y\sin\theta_i$ is the projection of the x and y coordinates onto the filtered projections. $\hat{Q}_{\theta_i}$ is given as a set of discrete samples. The projection of the resample location onto $\hat{Q}_{\theta_i}$ does not necessarily correspond to a position where $\hat{Q}_{\theta_i}$ is given. Therefore $\hat{Q}_{\theta_i}$ needs to be interpolated. As mentioned in [6] bilinear interpolation is adequate, and due to it's simplicity the method of choice. The derivation of $\hat{Q}_{\theta_i}$ can be reviewed in Section 3. Equation 4.1 is the three-dimensional extension of Equation 3.23.

4.1.2 Classification

The second step in the raycasting pipeline is classification, see Figure 4.2. Classification is the assigning of material properties to the reconstructed density value. The function which maps the density onto a material property is known as transfer function. We use two different transfer functions. The first maps the density value onto the opacity of the given sample. Samples with zero-opacity need not to be considered for any following computation, since they do not contribute to the final appearance of the image. The second transfer function assigns a color to samples with nonzero-opacity. After the classification step the opacity α_{s_i} and the color c_{s_i} of a given sample s_i are available.

4.1.3 Shading

The third step in the raycasting pipeline is shading, see Figure 4.2. Shading is the process to determine a sample's light intensity. In our approach the Phong illumination model is used for shading. The final light intensity of a sample depends on the sum of the ambient term $I_{ambient}$, the diffuse term $I_{diffuse}$, and the specular term $I_{specular}$. The shaded color $\overline{c}_{s_i}$ is obtained by multiplying the sample color by the light intensity:

$$\overline{c}_{s_i} = c_{s_i}\left(I_{ambient} + I_{diffuse} + I_{specular}\right) \tag{4.2}$$

The ambient term is determined only by the ambient coefficient:

$$I_{ambient} = C_{ambient} \tag{4.3}$$

Therefore the ambient term is constantly assigned to each sample. The ambient coefficient can be predefined in order to determine the amount of ambient light.

The diffuse term is used in order to simulate a diffuse reflection. According to Lambert's law, diffuse reflections only depend on the angle between the light vector L and the surface normal N. The light vector is the normalized vector pointing from the sample position in the direction of the light source, see Figure 4.3. Lambert's law states that the portion of the diffuse reflected light is equal to the cosine of the angle between N and L. Therefore, the diffuse term depends on the diffuse coefficient $C_{diffuse}$ and the dot product of N and L:

$$I_{diffuse} = C_{diffuse} N \cdot L \tag{4.4}$$

Note, that the dot product of two normalized vectors is equal to the cosine of the angle between these two vectors. The diffuse coefficient is chosen in order to control the intensity of diffuse reflections.

The specular term is used to simulate a specular reflection on the surface. It depends on the specular coefficient $C_{specular}$ and the specular exponent n. Both can be chosen to influence the appearance of specular reflections. Furthermore, it depends on the angle between the surface normal N and the halfway vector H. The halfway vector is the vector half way between the light vector and the surface normal, see Figure 4.3. The specular term is given by:

$$I_{specular} = C_{specular}(N \cdot H)^n \tag{4.5}$$

Higher values of n lead to smaller highlights while lower values of n lead to wider and softer highlights. The specular coefficient determines the amount of light to be reflected.

All the terms are by definition positive. Negative values of the dot product are therefore clamped to zero. It can easily be seen, that the diffuse

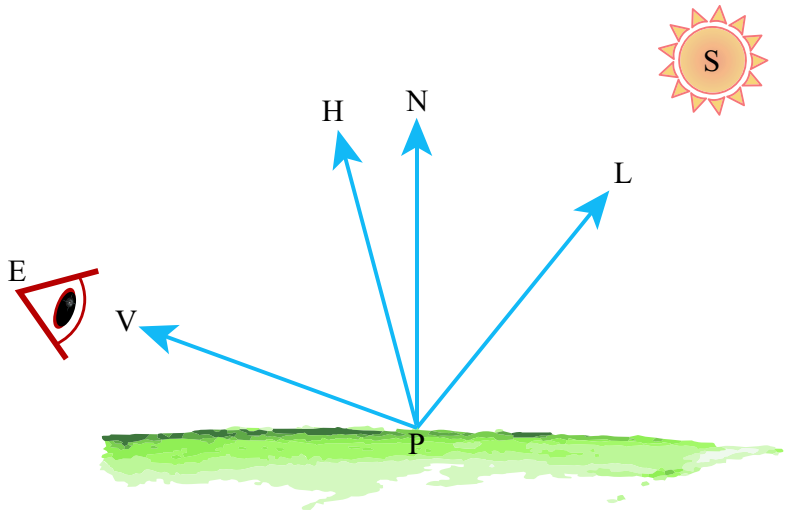


Figure 4.3: Vectors used in the Phong illumination model: N is the surface normal at point P. V is the vector pointing to the eye point E. L is the light vector pointing to the light source S. H is the halfway vector, half way between V and L.

term as well as the specular term depend on the surface normal. Due to the lack of surface normals in volumetric data they need to be approximated by a gradient estimation scheme. In case of a grid based representation the straightforward way is to estimate the gradient from a certain voxel neighborhood. To determine the gradient, common methods, such as intermediate difference gradient, central difference gradient, or higher order gradient estimation schemes are applied. In case of projection based volumetric data, computing the derivatives from a certain 3D neighborhood of samples requires to perform a large number of back projections. Especially for higher order gradient estimation schemes, which need a large neighborhood of samples, the computational costs would be significantly high. As presented in Section 3.2.3 the Filtered Back-Projection reconstruction scheme can also be exploited to compute derivatives directly. The partial derivative with respect

to x is given by

$$\begin{aligned} \tilde{f}_x = & \frac{\partial \tilde{f}(x,y,z)}{\partial x} = \\ = & \frac{\pi}{K} \sum_{i=1}^{K} \lim_{\Delta x \to 0} \frac{1}{\Delta x} (\hat{Q}_{\theta_i}(t_i, z) - \hat{Q}_{\theta_i}(\Delta x \cos\theta_i + t_i, z)) \end{aligned} \tag{4.6}$$

where $\hat{Q}_{\theta_i}(t, r)$ are the two-dimensional filtered projections and $t_i = x\cos\theta_i + y\sin\theta_i$. Analogously, the partial derivative with respect to y is given by:

$$\begin{aligned} \tilde{f}_y = & \frac{\partial \tilde{f}(x,y,z)}{\partial y} = \\ = & \frac{\pi}{K} \sum_{i=1}^{K} \lim_{\Delta y \to 0} \frac{1}{\Delta y} (\hat{Q}_{\theta_i}(t_i, z) - \hat{Q}_{\theta_i}(\Delta y \sin\theta_i + t_i, z)) \end{aligned} \tag{4.7}$$

And finally the partial derivative with respect to z is given by:

$$\begin{aligned} \tilde{f}_z = & \frac{\partial \tilde{f}(x,y,z)}{\partial z} = \\ = & \frac{\pi}{K} \sum_{i=1}^{K} \lim_{\Delta z \to 0} \frac{1}{\Delta z} (\hat{Q}_{\theta_i}(t_i, z) - \hat{Q}_{\theta_i}(t_i, z + \Delta z)) \end{aligned} \tag{4.8}$$

After the shading step the shaded color $\overline{c}_{s_i}$ is available. Again, the projection of the sample position onto the discrete filtered projections $\hat{Q}_{\theta_i}$, calls for interpolation. As already described for data reconstruction, bilinear interpolation is an adequate method.

4.1.4 Compositing

Compositing is the last step in the raycasting pipeline, see Figure 4.2. While the first three steps of the raycasting pipeline determine the appearance of a single sample, the compositing step accumulates the results of all evaluated samples of a ray. Raycasting can either be performed in back-to-front order or in front-to-back order. Back-to-front order starts the evaluation of samples at the background and continues in the direction of the eye-point. It describes the way of the light from back to front. The result of front-to-back order is equivalent to the result of back-to-front order. It starts the evaluation of samples at the image plane and continuous through the volume along the ray. In order to get the composited color of all the samples along a ray an

iterative front-to-back approach is used. Let c_{in} be the current composited color of the ray (initialized with black) and let further $\overline{c}_{s_i}$ be the color of the current sample, then the composited color c_{out} is given by:

$$c_{out} = c_{in} + \overline{c}_{s_i}\alpha_{s_i}(1 - \alpha_{in}) \tag{4.9}$$

where α_{in} is the accumulated opacity (initialized with zero). The composited opacity α_{out} of the ray is given by:

$$\alpha_{out} = \alpha_{in} + \alpha_{s_i}(1 - \alpha_{in}) \tag{4.10}$$

After the compositing step c_{in} is set to c_{out} and the next sample of the ray has to be evaluated until the ray reaches the end of the volume. If the whole volume is processed and α_{out} is still less than one, the background color $c_{background}$ with full opacity is applied to c_{out}:

$$c_{out} = c_{in} + c_{background}(1 - \alpha_{in}) \tag{4.11}$$

After the termination of the ray the composited color for one image pixel is available.

4.2 Accelerating D²VR

In order to speed up volume rendering, computations without any impact on the resulting image, needs to be avoided. In order to accelerate D²VR two different techniques are described in this Section.

4.2.1 Early ray termination

If raycasting is performed in front-to-back order the evaluation of samples along the ray can be stopped if the ray has nearly full opacity. If the opacity of the ray α_{out} reaches a certain threshold α_{accum} the ray is terminated. The threshold α_{accum} is less than but close to one. The evaluation along a ray is continued as long as $\alpha_{out} < \alpha_{accum}$. If $\alpha_{out} \geq \alpha_{accum}$ the contribution

of the samples behind the current sampling position can be neglected and therefore the ray can be terminated. Due to early ray termination, occluded parts of the volume are not processed and the volume rendering is accelerated considerably.

4.2.2 Hierarchical Space Partitioning

For almost all volumetric processing approaches a hierarchical space partitioning data structure is essential for efficient processing. The performance gain which can be achieved with such a data structure mainly depends on its granularity. An octree is one of the most widely used data structures for organizing three-dimensional space. An octree is based on the principle of hierarchical space partitioning. Each node of the octree represents a

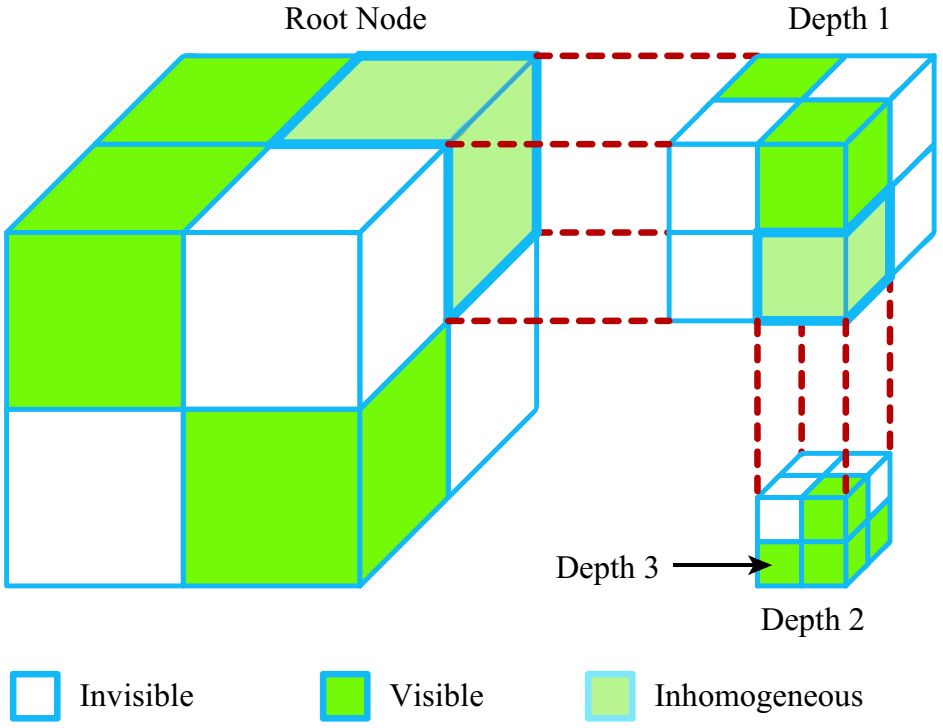


Figure 4.4: Octree nodes are classified as visible, invisible or inhomogeneous. Inhomogeneous nodes at depth level i are subdivided into 8 subnodes of depth level $i + 1$.

cuboid cell. In a min-max octree for volumetric data each node contains the minimum and maximum of the enclosed data. Due to this information an octree node is classified as *visible*, *invisible* or *inhomogeneous*, see Figure 4.4. The octree is created starting with a root cell which encloses the whole volume. The content of the root cell is classified as visible, invisible or inhomogeneous. If it is inhomogeneous the cell is subdivided into eight sub cells, see Figure 4.4. This subdivision is repeated recursively as long as a certain depth of the octree is reached. The complete octree consists of visible cells and invisible cells. Inhomogeneous leaf cells are treated as visible.

The first step during rendering is to find cells which contain visible data. It is important to note, that the visibility of cells only changes when the transfer function is modified. In a second step the visible cells are rendered into the z-buffer using OpenGL. The values in the z-buffer are then used to determine the position where the evaluation of the rays needs to be started. The octree can be used to efficiently skip data which is classified as transparent. Therefore the sampling process starts near the data of interest. As the evaluation of samples is a computational expensive task the use of a hierarchical space partitioning data structure for empty space skipping, significantly increases the performance of volume rendering. It is straightforward to find the minimum and maximum value of an octree cell in case of grid-based volumetric data as long as trilinear interpolation is assumed as reconstruction method. It is important to note that the minimum and maximum values enclosed by a cuboid cell of the octree depend on the used data reconstruction method. The convex nature of trilinear interpolation ensures that all function values within a cuboid are bounded by the minimal and maximal values at the grid positions.

Projection-Based Octree

The convexity condition is not valid for reconstruction based on the Filtered Back-Projection. However, it is still possible to generate a min-max octree for projection-based volumetric data.

Consider an octree cell C projected onto all filtered projections P_{θ_i}, see

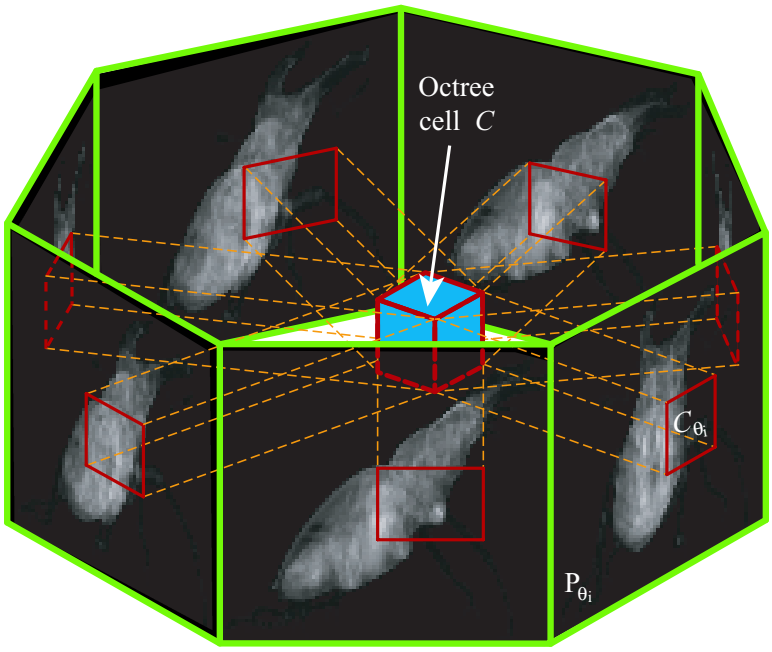


Figure 4.5: C_{θ_i} is the octree cell C projected onto projection P_{θ_i}.

Figure 4.5. The resulting projections of C are referred to as C_{θ_i}. According to Equation 3.17 an upper bound of the maximum value contained in the octree cell C can be determined by:

$$\max C_{\tilde{f}} \leq \frac{\pi}{K} \sum_{i=1}^{K} \max C_{\theta_{i\,\tilde{f}}} \tag{4.12}$$

where

$$C_{\tilde{f}} = \left\{ v = \tilde{f}(\vec{x}) | \vec{x} \in C \subseteq \Re^3 \right\}$$

are all possible function values within the cuboid cell C and

$$C_{\theta_{i\,\tilde{f}}} = \left\{ v = Q_{\theta_i}(\vec{t}) | \vec{t} \in C_{\theta_i} \subseteq \Re^2 \right\}$$

are all projection values within the enclosing rectangle of the cell C projected onto the projection P_{θ_i}. Analogously a lower bound of the minimum of the

octree cell C can be determined by:

$$\min C_{\tilde{f}} \geq \frac{\pi}{K} \sum_{i=1}^{K} \min C_{\theta_{i\,\tilde{f}}} \tag{4.13}$$

Note, that the projections of cell C onto the filtered projections are rectangles. As mentioned above the two-dimensional reconstruction scheme used on the filtered projections is bilinear interpolation. Due to the convex nature of bilinear interpolation the minimal and maximal value within a projected cell $C_{\theta_{i\,\tilde{f}}}$ are bounded by the minimal and maximal value at the grid points enclosed by the projected cell C. The algorithm to generate an octree for

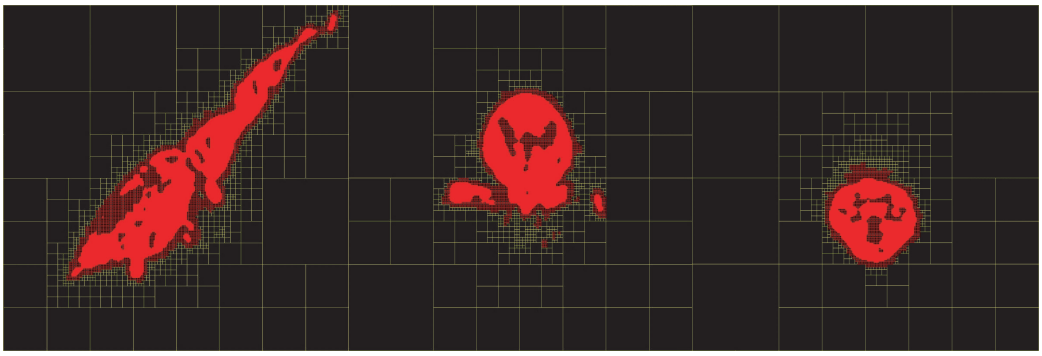


Figure 4.6: Octree up to depth 8 of the stag beetle data. Red octree cells are classified as visible.

projection-based volumetric data, is using a bottom-up approach and is as follows: Starting with a root cell, which encloses the entire volumetric function defined by the projections, we recursively, subdivide the octree cells. Once we reach a desired octree depth we compute the maximum respectively the minimum using Equation 4.12 and 4.13. These values are then propagated to the higher levels. Instead of propagating these minimums and maximums to the higher levels, it would have been also possible to compute the minimum and maximum directly for each of these higher levels using Equation 4.12 and 4.13. However, this would lead to a much less efficient approximation. The algorithm to generate a projection-based octree is given in pseudo code in Section 5.2. Although this space partitioning is a conservative

approximation, it works very well in practice, as shown in Figure 4.6.

4.3 Summary

In this Chapter D^2VR based on an image order approach was presented. In Section 4.1 the raycasting pipeline was described. All the steps of the pipeline are carried out using projection-based volumetric data. In order to accelerate D^2VR two different techniques were described in Section 4.2. In Section 4.2.1 early ray termination was briefly reviewed and in Section 4.2.2 an acceleration data structure based on the principle of hierarchical space partitioning was presented. In Chapter 5 details about the implementation of the concepts presented in this Chapter, are given.

Chapter 5

Implementation

A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it.

Albert Einstein

In Chapter 4 we presented all the necessary components for D^2VR. First, in Section 4.1 we presented an approach for data reconstruction as well as the computation of gradients directly from filtered projections at arbitrary sample positions. In Section 4.2 we presented acceleration techniques for D^2VR. We described early ray termination and presented a hierarchical space partitioning data structure for projection-based volumetric data, which is used in the following as acceleration data structure for D^2VR. All these approaches do not employ an intermediate grid representation.

5.1 Computed Tomography Scanner Simulation

For a proof of concept implementation of all the presented approaches, a projection based representation of the volumetric function needs to be given. Unfortunately, we did not have access to the projection data of Computed Tomography scanners. Therefore a Computed Tomography scanner simulation was implemented. In order to estimate the accuracy of the data reconstruction scheme and the proposed gradient estimation method different types of density functions had to be implemented. The goal was to estimate the accuracy as well as the visual quality of our approach. We tested D^2VR using artificially generated projections. A simulation of a Computed Tomography scanner was implemented using C++. The CT scanner simulation is able to cope with analytically defined continuous density functions. Furthermore, it is able to handle so called phantoms. Finally, high resolution rectilinear grids in combination with trilinear interpolation are used as input for the CT scanner simulation. In Figure 5.1 the work flow of the CT scanner simulation can be seen. First of all a three-dimensional density function is defined. This function is then used as input for the CT scanner simulation. The scanner simulation generates parallel projections of the three-dimensional input function. In order to get a projection P_{θ_i} the density function $f(x, y, z)$ needs to be integrated along parallel lines. Each of the possible 3D functions is implemented as a separate class derived from the class *3DFunction*. The class *3DFunction* provides an interface *LineIntegration* which is implemented differently for each of the 3D functions. This interface is used by the class *CTScanner* in order to compute the line integral. The line integral is denoted $I_\theta(t)$ in the following. If possible the line integration is implemented analytically. However, if no analytical line integral is given the density function's integral is numerically approximated. For the numerical approximation of the line integral an algorithm based on the trapezoidal rule is used. The line integral of a line with a given normal distance to the origin t_0 in the interval

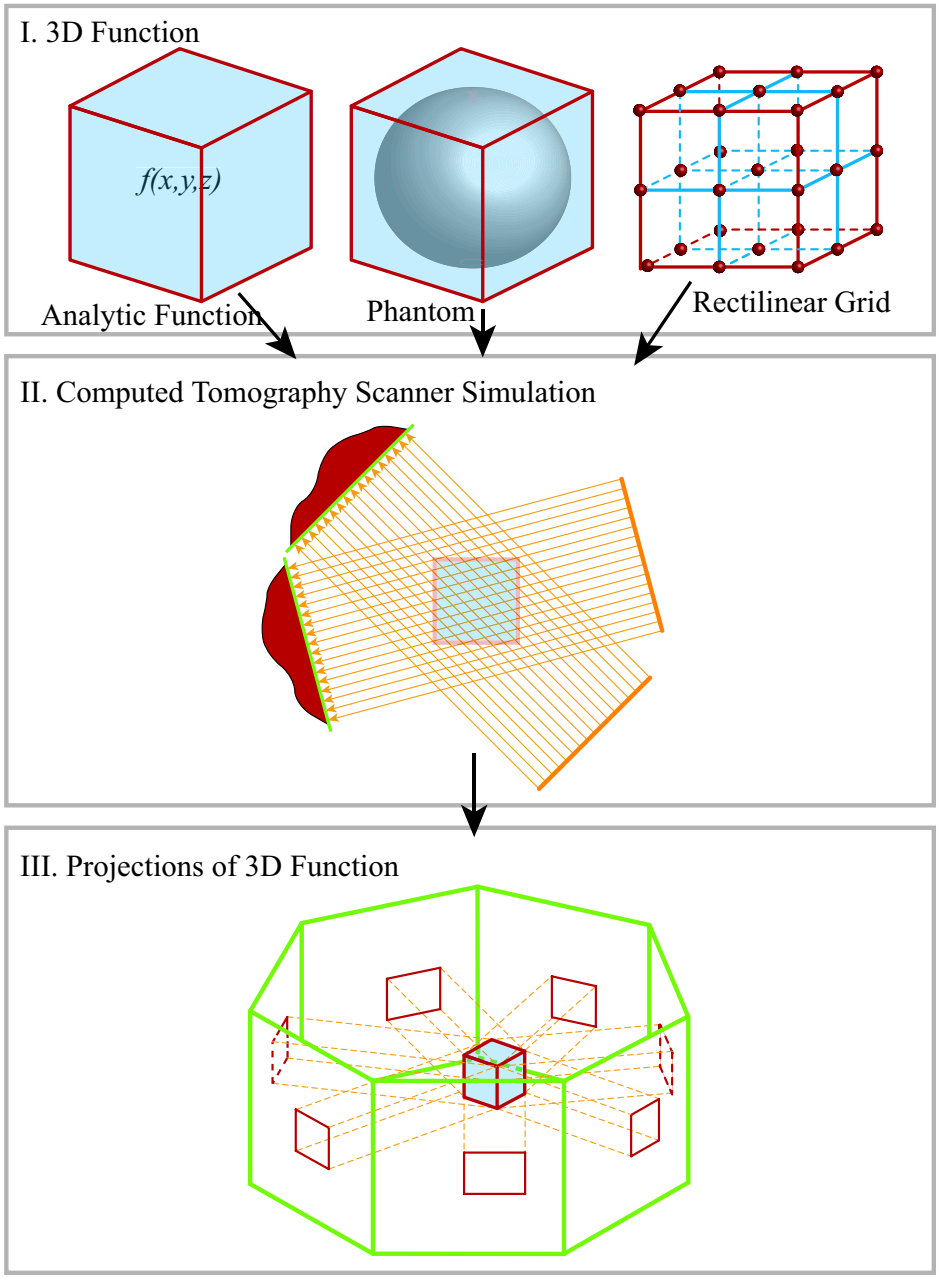


Figure 5.1: Work flow (part one). I: Three different types of density functions are defined. II: The Computed Tomography scanning process is simulated, it takes a three-dimensional density function as input and generates projection images as output. III: Projections are saved to disk.

s_0 to s_1 can be numerically approximated with:

$$I_\theta(t) = \int_{s_1}^{s_0} f(s_\theta, t_0)\ ds_\theta \approx \sum_{i=0}^{N-1} \frac{\varepsilon_s}{2} f(s_0 + i\varepsilon_s, t_0) \tag{5.1}$$

where N is the number of samples and ε_s is the sampling distance given by:

$$\varepsilon_s = \frac{s_1 - s_0}{N - 1} \tag{5.2}$$

The accuracy of the trapezoidal rule integration mainly depends on the number of samples. The greater the number of samples the higher the accuracy but also the computation time. In order to simulate the Computed Tomography scanning process a large number of line integrals needs to be computed. Therefore the faster Romberg integration is used. Romberg integration is based on the idea of performing the numerical integration for various values of ε_s, and then extrapolating the result to the continuum limit $\varepsilon_s = 0$. As described in [18] Romberg integration is one or even two magnitudes faster than brute force numerical integration and achieves the same accuracy.

In the following the three different types of three-dimensional functions are described:

Marschner & Lobb Function As a continuous analytically defined test function the well known Marschner & Lobb function is used. In Figure 5.2 an iso-surface rendering of the Marschner & Lobb function can be seen. The density function is given by:

$$f(x, y, z) = \frac{1 - \sin(1/2\,\pi\, z) + \alpha\,\left(1 + \cos\left(2\,\pi\, f_M\, \cos\left(1/2\,\pi\,\sqrt{x^2 + y^2}\right)\right)\right)}{2 + 2\,\alpha} \tag{5.3}$$

where f_M and α are parameters which influence the appearence of the function. The original setting [11] is $f_M = 6$ and $\alpha = 0.25$. All following results were attained with these settings. In Figure 5.2 an iso-surface rendering of the Marschner & Lobb function is shown. The Marschner & Lobb function is integrated using Equation 5.1. Since the partial derivatives of the Marschner & Lobb function can be analytically computed, it is used to estimate the ac-

Figure 5.2: Analytical iso-surface rendering of the Marschner & Lobb function with the isovalue $f(x, y, z) = 0.5$

curacy of the gradient estimation scheme. The partial derivative with respect to x is given by

$$f_x(x,y,z) = \frac{\alpha\ \sin\left(2\,\pi\,f_M\ \cos\left(1/2\,\pi\,\sqrt{x^2+y^2}\right)\right)\pi^2 f_M\ \sin\left(1/2\,\pi\,\sqrt{x^2+y^2}\right)x}{\sqrt{x^2+y^2}\,(2+2\,\alpha)} \tag{5.4}$$

Analogously the partial derivative with respect to y is given by

$$f_y(x,y,z) = \frac{\alpha\ \sin\left(2\,\pi\,f_M\ \cos\left(1/2\,\pi\,\sqrt{x^2+y^2}\right)\right)\pi^2 f_M\ \sin\left(1/2\,\pi\,\sqrt{x^2+y^2}\right)y}{\sqrt{x^2+y^2}\,(2+2\,\alpha)} \tag{5.5}$$

Finally, the partial derivative of the Marschner & Lobb function with respect to z is given by

$$f_z(x,y,z) = -1/2\,\frac{\cos\left(1/2\,\pi\,z\right)\pi}{2+2\,\alpha} \tag{5.6}$$

The results of the comparison of different gradient estimation schemes are presented in Chapter 6.

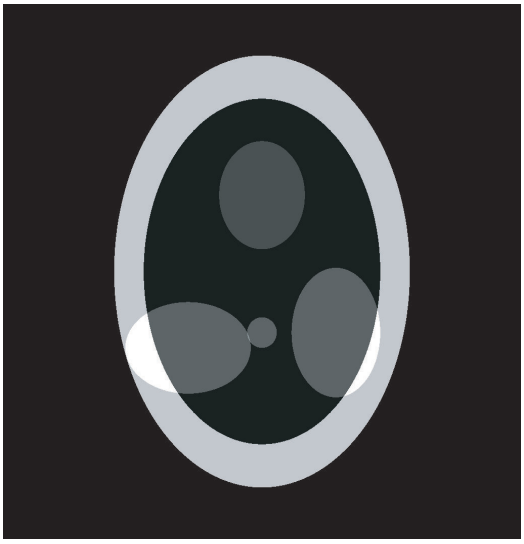


Figure 5.3: Slice of a 3D Phantom. Phantoms are defined as a collection of primitives each with a given density. The density of the Phantom is defined as the sum of densities of the primitives.

Phantoms The Phantoms are analytically defined density functions. In contrast to the Marschner & Lobb function, they are not continuous and therefore not differentiable. They are defined as a set of geometrical primitives such as cylinders, spheres, ellipsoids, boxes, etc. In Figure 5.3 an example of a phantom is shown. The geometrical primitive p_i has a defined density d_i inside. The density function of a geometrically defined primitive is given by:

$$f_{p_i}(x,y,z) = \begin{cases} d_i & \text{if } (x,y,z) \text{ is inside } p_i \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

The density function of the phantom at a certain sample position is defined as the sum of the density values of the geometrical primitives containing this sample position.

$$f(x,y,z) = \begin{cases} \tilde{f}(x,y,z) & \text{if } \tilde{f}(x,y,z) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

where

$$\tilde{f}(x,y,z) = \sum_{i=1}^{K} f_{p_i}(x,y,z) \tag{5.9}$$

and K is the number of geometrically defined primitives. While the density function of the phantom is by definition always positive, the density of single geometrical primitives may also be negative. Due to the simplicity of the primitives the line integral of the phantom can be analytically evaluated. An intersection of the line and each geometrically defined primitive has to be computed. The length of the intersecting line segment of a line and the primitive p_i is denoted as l_i. The line integral of the phantom is given by:

$$I_\theta(t) = \begin{cases} \tilde{I}_\theta(t) & \tilde{I}_\theta(t) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\tilde{I}_\theta(t)$ is

$$\tilde{I}_\theta(t) = \sum_{i=1}^{K} l_i d_i \tag{5.10}$$

The phantoms are used to estimate the accuracy of the data reconstruction scheme. The results are summed up in Chapter 6.

Figure 5.4: Example of a dataset given on a rectilinear grid.

Rectilinear Volumetric Data As a proof of concept the visual quality of D^{2}VR is tested on high resolution rectilinear grids. In Figure 5.4 an example of a dataset given on a rectilinear grid can be seen. It has to be

taken into account, that the density function of the rectilinear grid is only given at the sample positions. In order to evaluate line integrals, the density function $f(x, y, z)$ has to be defined using a certain reconstruction scheme. We define $f(x, y, z)$ by using trilinear interpolation of the rectilinear grid. Let $f_{000} \dots f_{111}$ be the eight samples of the cubic cell enclosing the sample position $(x, y, z)^T$ and let further $(x_s, y_s, z_s)^T$ be the sample location within the cubic cell. According to trilinear interpolation $f(x, y, z)$ is then:

$$\begin{aligned} f(x,y,z) = \quad & f_{000}(1-x_s)(1-y_s)(1-z_s) + \\ & f_{100}x_s(1-y_s)(1-z_s) + \\ & f_{010}(1-x_s)y_s(1-z_s) + \\ & f_{001}(1-x_s)(1-y_s)z_s + \\ & f_{011}(1-x_s)y_sz_s + \\ & f_{101}x_s(1-y_s)z_s + \\ & f_{110}x_sy_s(1-z_s) + \\ & f_{111}x_sy_sz_s \end{aligned} \tag{5.11}$$

The line integrals are computed with the numerical integration method described above.

During the scanning process the results of the line integration are stored in projection images. The result of the scanning process are the projections of the density function. They are visualized as two dimensional textures. All intermediate as well as all final results are visualized using OpenGL. In Figure 5.5, 64 projections of a human head can be seen. Each row corresponds to the projections taken within an angle range of 45 degrees. All values of the projections are scaled for the visualization to the range $0 \dots 1$. White corresponds to the highest value of the line integrals and black corresponds to zero. While the projections in Figure 5.5 are shown side by side they can also be arranged in their 3D spatial context, see Figure 5.6.

In order to compare the accuracy of different filters, all the filters described in Section 3.2.2 are implemented. The filtering is implemented in frequency domain as well as in spatial domain. The filtering in frequency domain is much faster and the results are, as one would expect, equivalent.

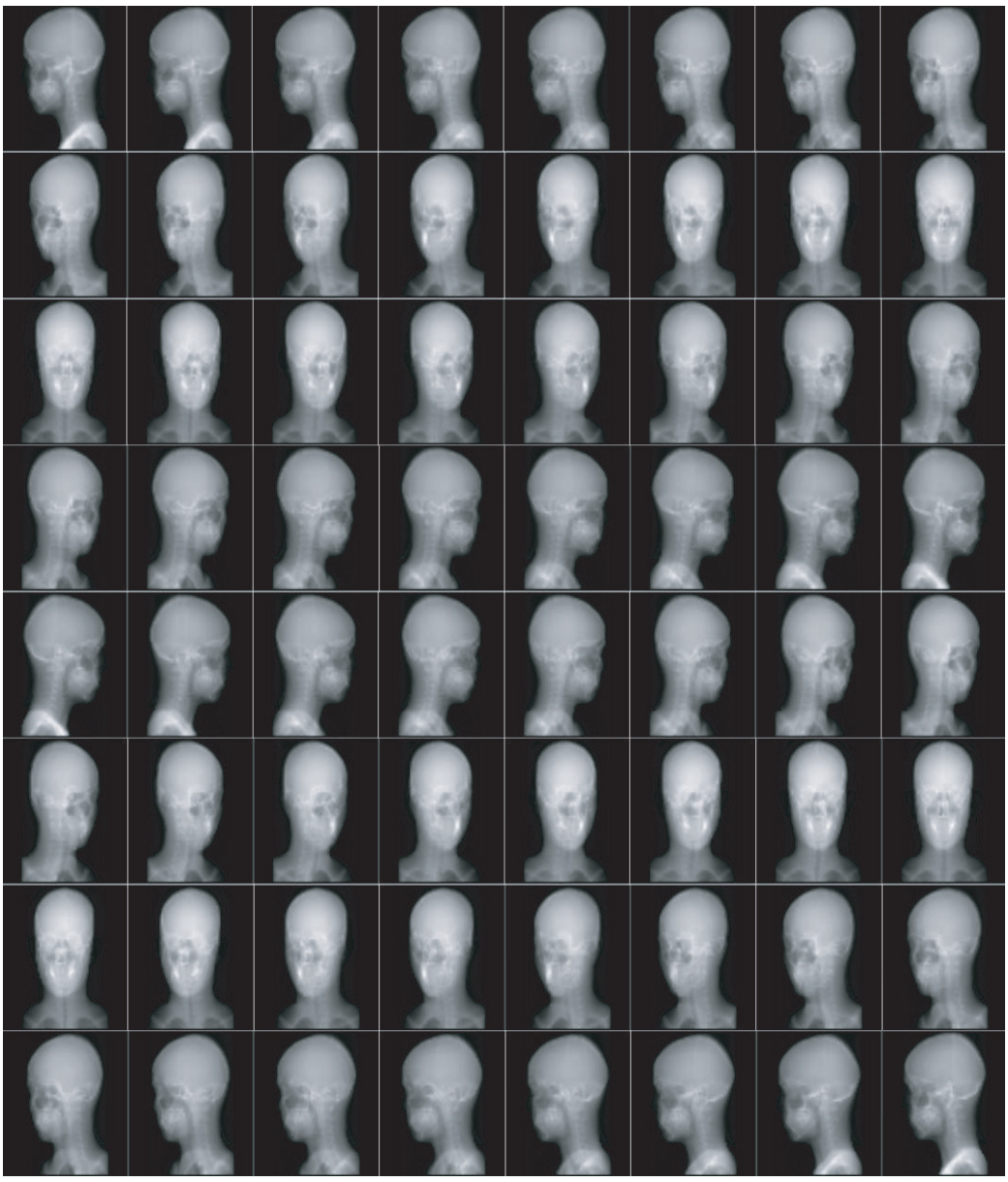


Figure 5.5: 64 projections of a rectilinear grid. The values of the projections are scaled for the visualization.

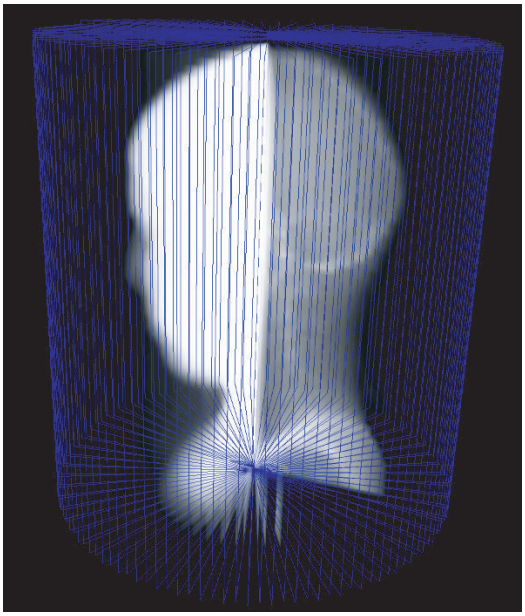


Figure 5.6: 3D arrangement of 64 projections.

5.2 D^{2}VR Implementation

While the first part of the implementation deals with the generation of the filtered projections, the second part of the implementation deals with the visualization of the projection based volumetric data. The work flow of the second part of the implementation can be seen in Figure 5.7. The projection based volumetric data is either visualized directly from the filtered projections using D^{2}VR or resampled on an intermediate rectilinear grid and visualized using DVR. In order to compare the visualization results the original density function (which was used as input for the scanning process) can also be visualized using raycasting.

We implemented a CPU-based as well as a GPU-based prototype for orthographic and perspective projection. The CPU implementation is based on the raycasting approach. For each pixel of the image plane, rays are cast through the volumetric space enclosed by the filtered projections. At each resample location the underlying 3D density function is reconstructed according to Equation 3.17 and gradient estimation is done using Equation 3.32.

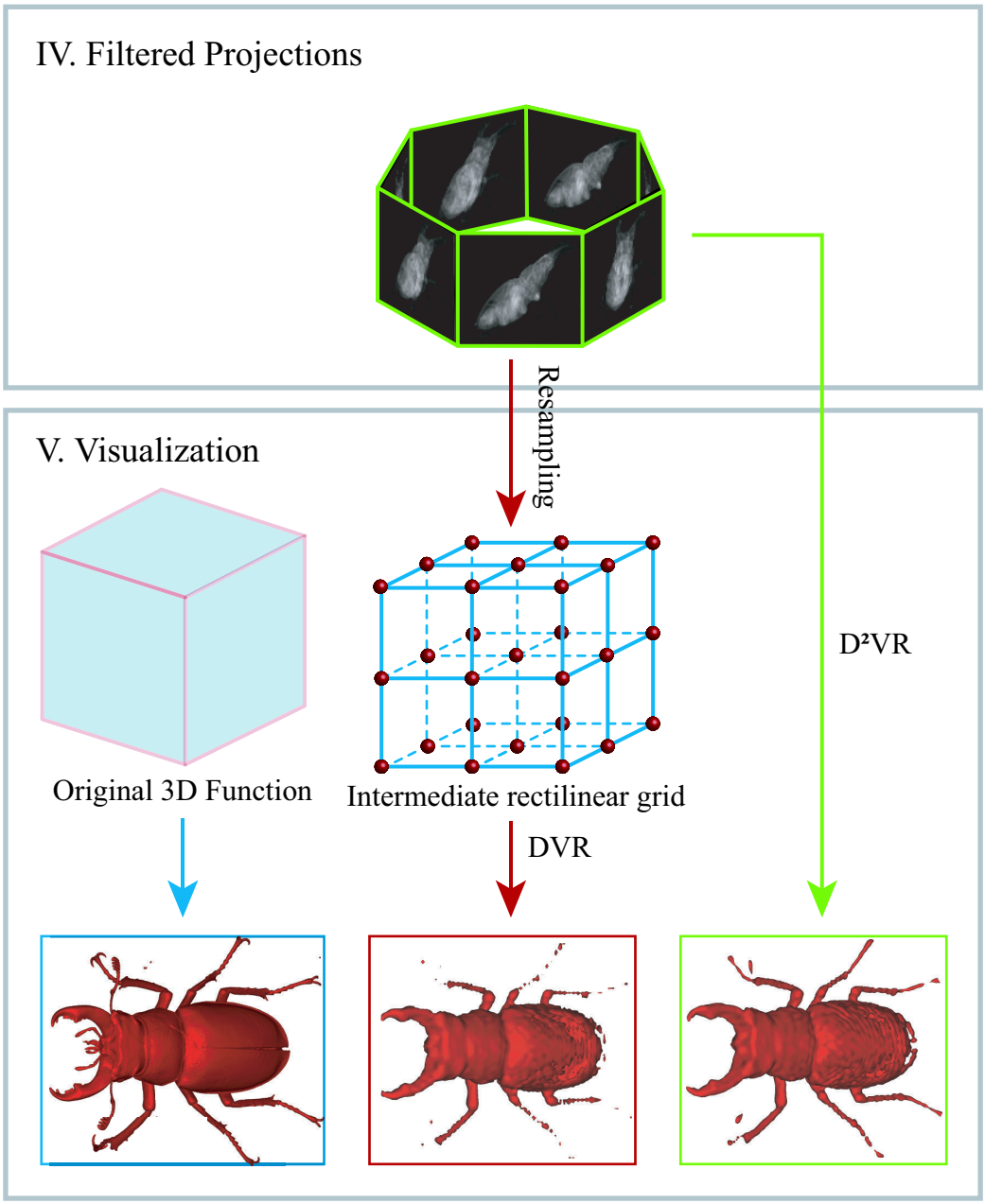


Figure 5.7: Work flow (part two). IV: The projections are loaded from disk and filtered. V: Three representations of the density function are visualized: Left: The analytically defined function is visualized; Middle: A rectiliner grid is resampled from the projections and visualized using DVR (this corresponds to the traditional volume rendering pipeline); Right: The projection-based volumetric function is visualized using D^2VR.

The final color and opacity of the image pixel is determined by the over-operator [17] in front-to-back order. The GPU version is implemented in C++ and Cg using OpenGL. We are aware that the current Cg compiler does not always produce optimal code. We did not manually optimize the code by using assembly as this is a proof of concept implementation. We implemented D^2VR on the GPU (NVidia GForce 6800 GT, 256 MB) utilizing texture-based volume rendering with view-aligned slices in combination with filtered projection textures. All the filtered projections are linearly stored in a 3D texture. Basically we compute texture slices parallel to the viewing plane. As the newest NVidia hardware supports dynamic branching, the slices are directly computed in a *for*-loop using Equation 3.17 from all the filtered projections. The same is done for the gradient computation using Equation 3.32. The volumetric space enclosed by the filtered projections is thereby cut by polygons parallel to the viewing plane. The polygons are then projected by the graphics hardware onto the image plane and composited using alpha blending. In order to accelerate the rendering process, we utilized the octree described in Section 4.2.2. We project all visible octree cells onto the image-plane. The resulting z-buffer image is then used for early z-culling, a capability of modern hardware, implementing empty space skipping. On the average, for example, we measured for 128 projections (128^2 sized) using a 256^2 view-port four seconds per frame for an iso-surface rendering. Further optimization, such as early ray termination, are not employed due to the lack of graphics hardware support. In the future we will develop alternative acceleration approaches in order to provide fully interactive projection-based volume rendering. In order to get an impression of the quality of the conservative approximation of the min-max octree the visible and invisible cells were rendered in a 3D as well as in a 2D wire-frame mode using OpenGL. In Figure 5.8 an example of an octree is shown in 3D. The creation of such an octree is achieved by recursive calls to the function *CreateOctreeNode*. The implementation of the function *CreateOctreeNode* is given in pseudo-code in Algorithm 1. Note that the argument *cuboidCell* is an instance of a class *Cell*, that holds a geometric description of the bounding cuboid of the cell. The class *Cell* implements a method *createSubCell(i)* which returns an instance

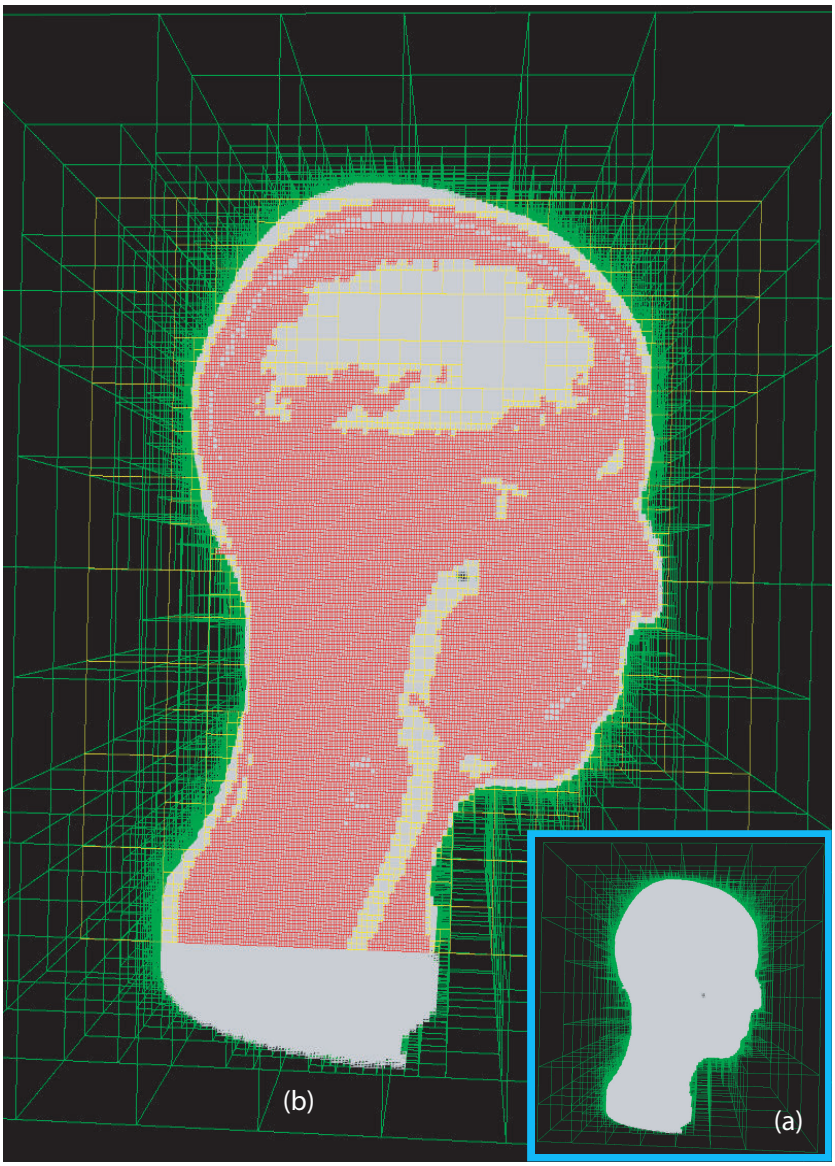


Figure 5.8: Wire-frame mode rendering of visible and invisible octree cells. (a) Gray cells are classified as visible, green cells are classified as invisible. (b) Same rendering as (a) overlaid with a slice through the octree. The red cells are visible 2D cells on the slice, the yellow cells are invisible cells on the slice.

Algorithm 1 Recursive octree creation

```
CreateOctreeNode(currentDepth, maxDepth, cuboidCell)
if (currentDepth < maxDepth) then
  child[1].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(1))
  child[2].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(2))
  child[3].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(3))
  child[4].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(4))
  child[5].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(5))
  child[6].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(6))
  child[7].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(7))
  child[8].Create(currentDepth+1, maxDepth, cuboidCell.CreateSubCell(8))
else
  GetMinMaxValueWithin(cuboidCell)
end if
if (HasChildren) then
  minValue = child[1].minValue
  maxValue = child[1].maxValue
  for i = 2...8 do
    if (minValue > child[i].minValue) then
      minValue = child[i].minValue
    end if
    if maxValue < child[i].maxValue then
      maxValue = child[i].maxValue
    end if
  end for
end if
```

of the class *Cell* with a geometric description of the sub-cell i.

In order to create a min-max octree a root node has to be initialized and its method *CreateOctreeNode* has to be called.

Chapter 6

Results

Once you replace negative thoughts with positive ones, you'll start having positive results.

Willie Nelson

In order to show the differences between projection-based and grid-based data reconstruction and gradient estimation we simulated a Computed Tomography scanner. We scanned several different density functions, such as the Marschner & Lobb function, a phantom, a stag beetle, a human head and a carp. The Marschner & Lobb function is analytically defined. For the other data sets, an analytical representation is not given, therefore we took high resolution grids in combination with trilinear interpolation as ground truth. The Marschner & Lobb function is scanned taking 64 projections, each projection with a resolution of 64^2. From this projections a grid is reconstructed, using the same amount of samples (64^3). Additionally we also reconstructed a grid with eight times more samples (128^3). Furthermore we computed an iso-surface directly from the analytical Marschner & Lobb function. Figure 6.1a shows the differences between the analytical value and the reconstructed value using trilinear interpolation on the grid (64^3) and Figure 6.1b shows the differences between the analytical value and the

reconstructed value using Filtered Back-Projection. To encode the data reconstruction error on the iso-surface a color coding is applied. Green encodes low error, on the other hand red encodes higher errors.

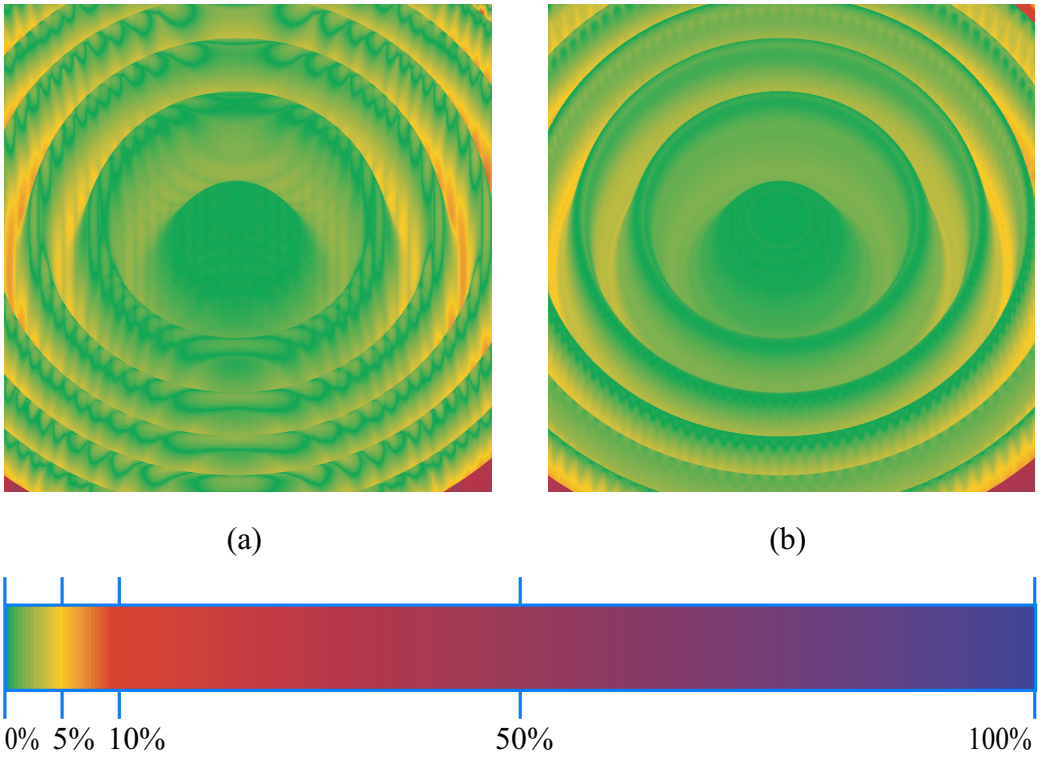


Figure 6.1: Color encoded differences between analytical value and (a) the reconstructed value using trilinear interpolation on the grid (64^3), (b) the reconstructed value using Filtered Back-Projection.

Figure 6.2a shows the orientation differences in degrees between the analytically computed gradients and the estimated gradients using central difference gradient estimation. Figure 6.2b shows the orientation differences in degrees between the analytically computed gradients and the estimated gradients using our new projection-based gradient estimation method.

Figure 6.3 shows a comparison of iso-surface renderings of the Marschner & Lobb function: Figure 6.3a shows an analytical rendering. Figure 6.3b shows DVR of the 64^3 grid. Figure 6.3c shows DVR of the eight times bigger grid. And finally in Figure 6.3d our D^2VR from 64 filtered projections, each

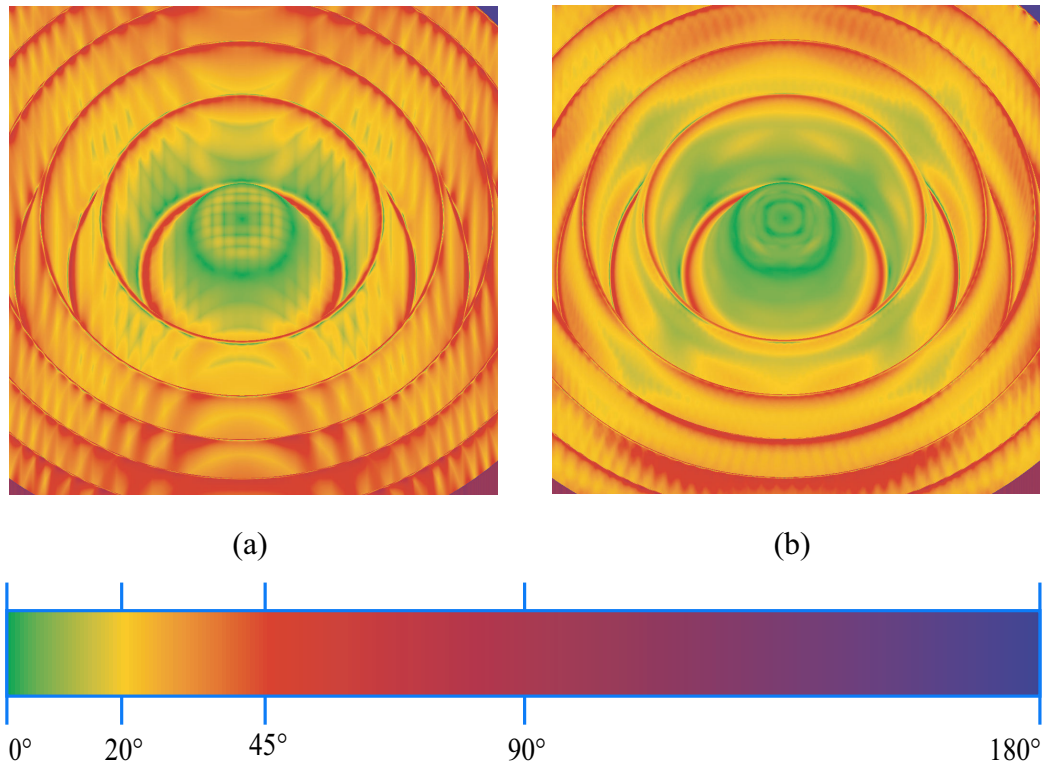


Figure 6.2: Color encoded orientation differences in degrees between analytically computed gradients and (a) the estimated gradients using central difference gradient estimation on the grid, (b) the estimated gradients using our new projection-based gradient estimation method.

projection with a resolution of 64^2 is shown.

Furthermore, we rendered a carp, a human head, and a stag beetle. In Figure 6.4, Figure 6.5, and Figure 6.6 the differences between rendering from projection-based and grid-based volumetric data can be seen. In order to visualize the influence of the filters on the data reconstruction, a slice of a phantom was reconstructed. In Figure 6.7 the influence of the different filters can be seen. The slice of the phantom was reconstructed from 256 projections, each with a resolution of 256^2. In Figure 6.7a the filter $H(w) = w$ was used for reconstruction. In Figure 6.7b the filter from Equation 3.2.1 in Section 3.2.2 was used for reconstruction. In Figure 6.7c the same filter as in Figure 6.7b was used but multiplied by a Hanning window. In order to

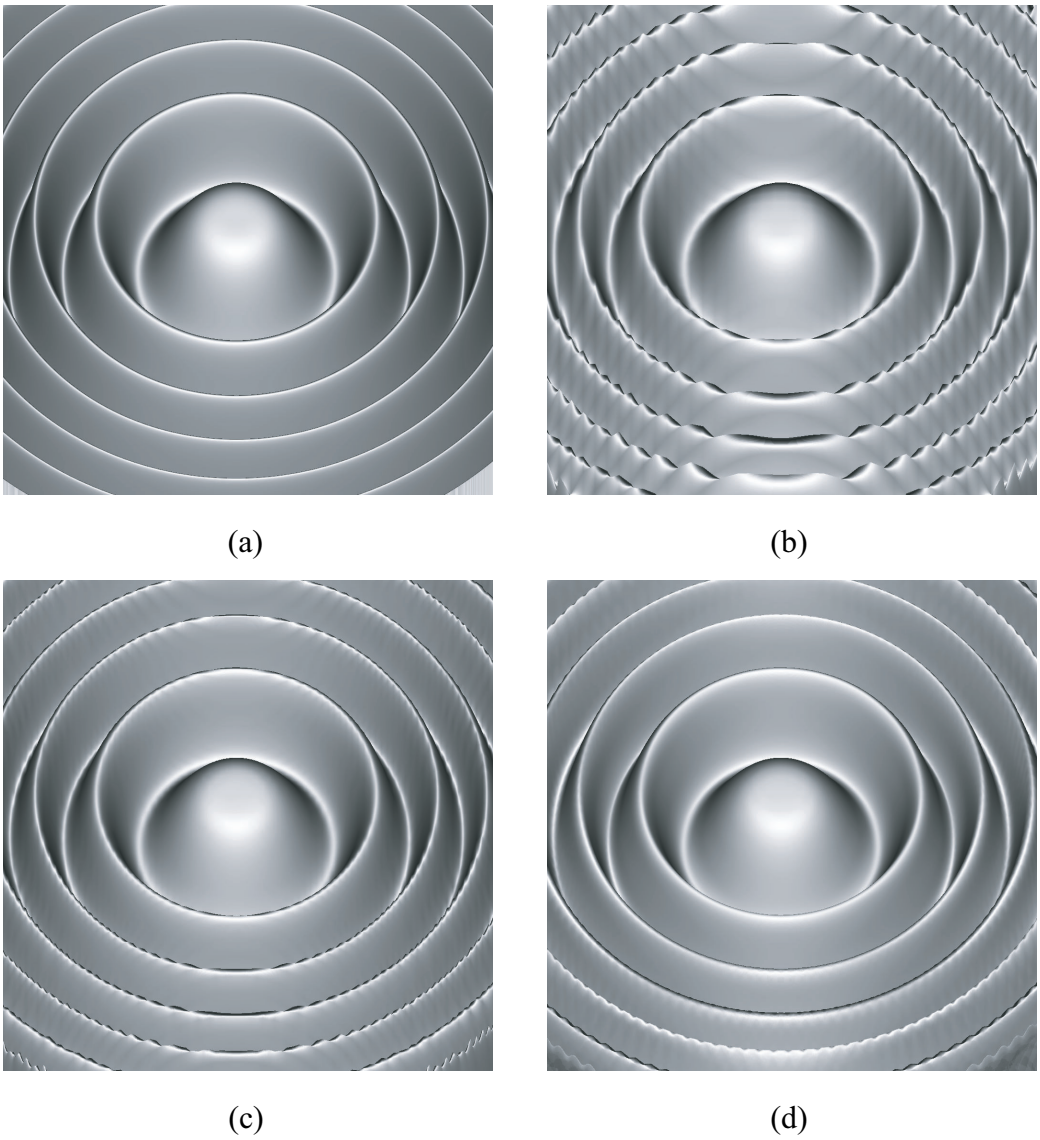


Figure 6.3: Comparison of an iso-surface of the Marschner & Lobb function: (a) Analytically computed. (b) DVR rendering of a 64^3 grid. (c) DVR rendering of an eight times bigger grid (128^3). (d) D^2VR from projection-based volumetric data (64 projections, each projection with a resolution of 64^2). Grids are reconstructed from 64 filtered projections, each projection with a resolution of 64^2.

compare the different results, difference images were rendered. In Figure 6.7d the difference image between the analytically defined phantom and the re-

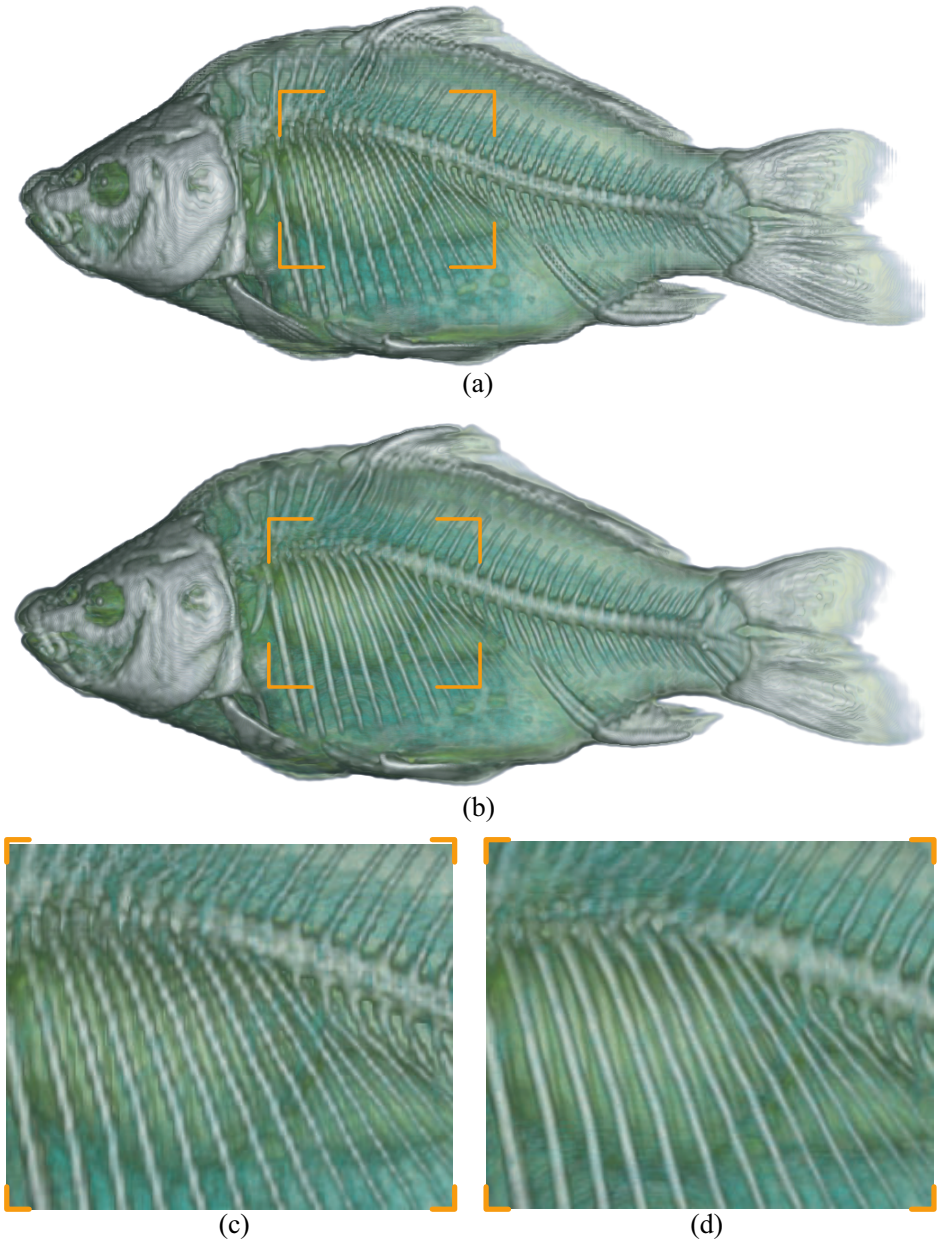
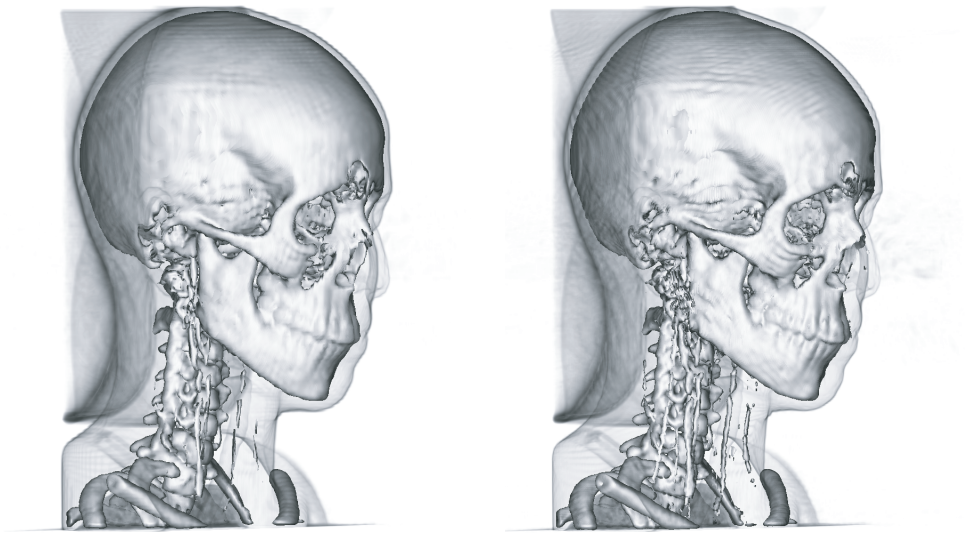


Figure 6.4: CT scan of a Carp (256x256x512): (a) DVR of a 128x128x256 grid, reconstructed from 128 filtered projections, each projection with a resolution of 128x256. (b) D^2VR of projection-based volumetric data (128 projections, each projection with a resolution of 128x256). (c) Zoom in of DVR. (c) Zoom in of D^2VR.



(a) (b)

Figure 6.5: CT scan of a human head (512x512x333): (a) DVR of a 128x128x128 grid, reconstructed from 128 filtered projections, each projection with a resolution of 128x128. (b) D^2VR of projection-based volumetric data (128 projections, each projection with a resolution of 128x128). Note, that due to the second resampling step of the traditional volume rendering pipeline many details are lost in (a).

constructed slice in Figure 6.7a can be seen. In Figure 6.7e the difference image between the analytically defined phantom and the reconstructed slice in Figure 6.7b can be seen; and finally in Figure 6.7f the difference image between the analytically defined phantom and the reconstructed slice in Figure 6.7c can be seen. All difference images are contrast enhanced in order to make the differences between them visible.

In order to estimate the average data reconstruction error we reconstructed a 128^3 grid from the filtered projections for all the shown data sets. From this grid as well as from the filtered projections we also reconstructed a rotated grid. In Table 6.1 the root mean squared data reconstruction error with respect to the analytical function respectively to the corresponding high resolution grid in combination with trilinear interpolation is shown.

We estimated the average computation time for a 256^2 image rendered

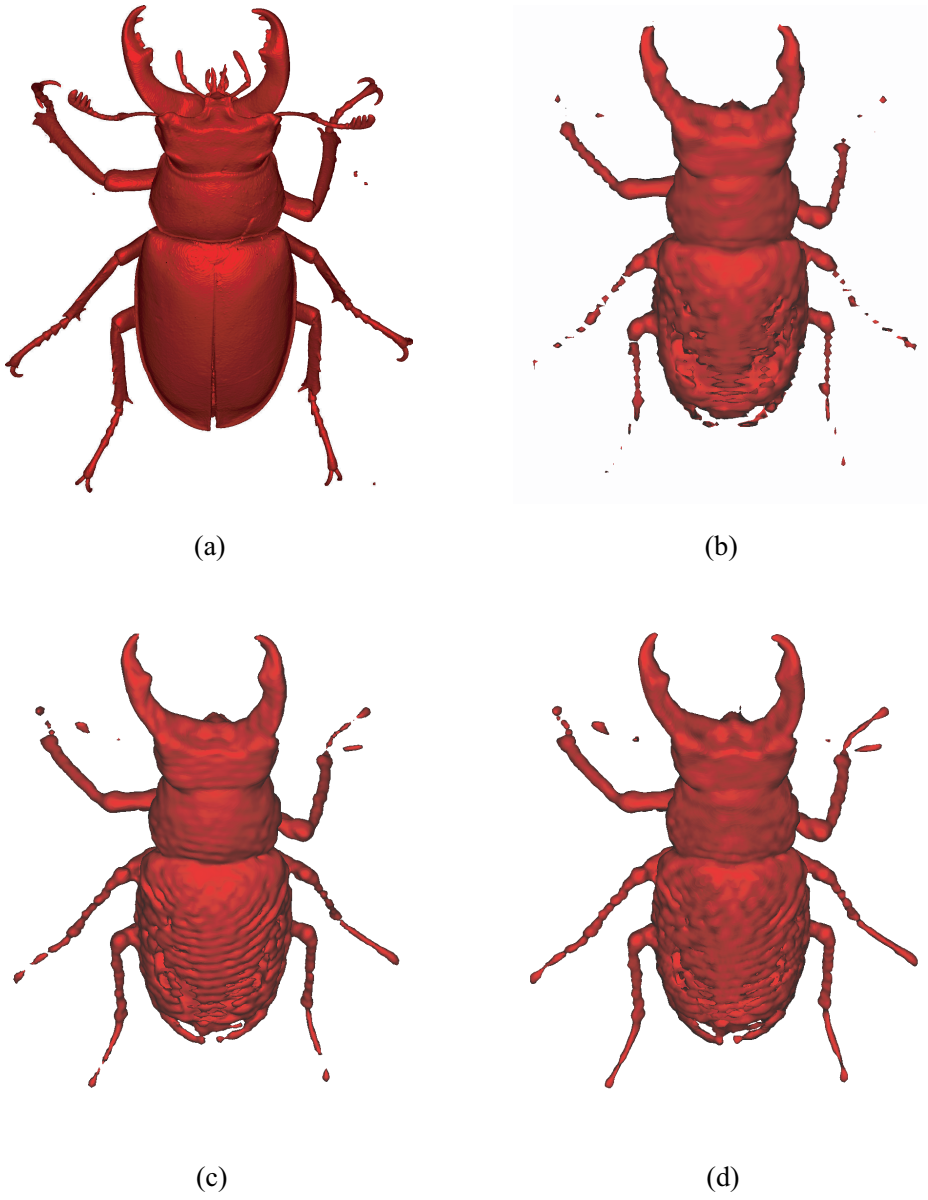


Figure 6.6: CT scan of stag beetle (832x832x494): (a) DVR of original grid. (b) DVR of a 64^3 grid. (c) DVR of a 128^3 grid. (d) D^2VR of projection-based volumetric data (64 projections, each projection with a resolution of 64^2). Grids are reconstructed from 64 filtered projections, each projection with a resolution of 64^2.

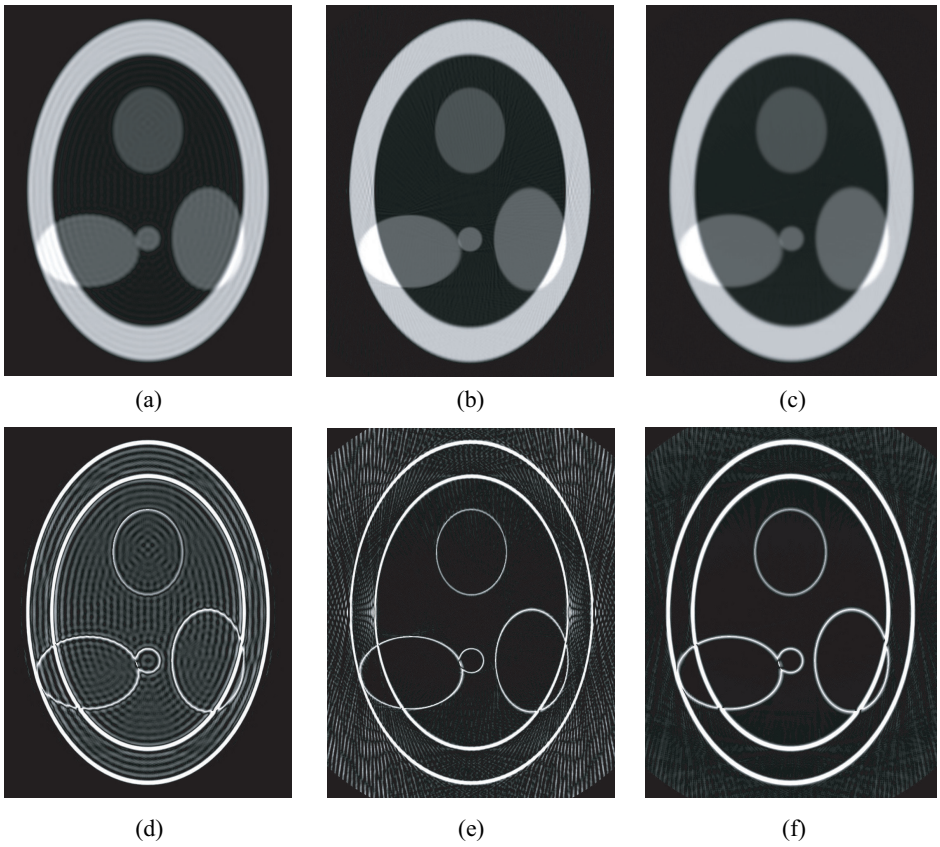


Figure 6.7: Top: Slices of a phantom, reconstructed with different filters (a) the filter $H(w) = |w|$ was used, (b) the filter of Equation 3.2.1 was used, (c) the filter of Equation 3.2.1 multiplied by a Hamming window was used. Bottom: Difference images: (d) difference of analytically defined phantom and (a), (e) difference of analytically defined phantom and (b), (f) difference of analytically defined phantom and (c). The used slice of the analytically defined phantom can be seen in Figure 5.3. All difference images are contrast enhanced.

Data	RMS of grid	RMS of filtered projections	Ratio
Marschner & Lobb	0.0624	0.0537	1.1620
Phantom	0.0720	0.0641	1.1232
Fish	37.5486	26.9216	1.3947
Human head	89.8823	73.3274	1.2258
Stag beetle	80.2559	77.8378	1.0311

Table 6.1: Root Mean Squared (RMS) data reconstruction error with respect to the analytical function respectively to the corresponding high resolution grid in combination with trilinear interpolation. Data values for the Marschner & Lobb data set are between zero and one, and for the other data sets between zero and 4095. In column three the ratio of column one and two is shown.

from 128 projections each of size 128^2, as well as for a 512^2 and a 1024^2 image for a brute force CPU implementation of raycasting. The results can be seen in Table 6.2.

Image size	256^2	512^2	1024^2
Computation time in Seconds	1453	5982	24037

Table 6.2: Computation time for images of different size. All computations were performed on 128 projections each of size 128^2.

Chapter 7

Summary

It is done.

Jesus Christ

7.1 Introduction

Medical visualization of volumetric datasets is a powerful tool to understand a patient's pathological conditions, to improve surgical planning, and to support medical education. The most widely used techniques for three-dimensional medical visualization are direct volume rendering methods. All direct volume rendering methods are based on rectilinear grid representations of the volumetric function. These rectilinear grids are usually provided by modern 3D scanning technologies, like Computed Tomography (CT) or Magnetic Resonance Imaging (MRI). These data values are computed from measured projections by discrete tomographic reconstruction. The set of the reconstructed data values (or samples) can be interpreted as a discrete representation of the underlying continuous phenomenon. In order to authentically visualize the original continuous signal, it has to be accurately reconstructed from the discrete samples (note that such a signal reconstruction is differentiated from discrete tomographic reconstruction). From a signal-processing point of view, the original signal can be perfectly reconstructed

from discrete samples if it is band-limited and the sampling frequency is above the Nyquist limit. Theoretically the perfect continuous reconstruction is obtained by convolving the discrete volume representation by the *sinc* function. The *sinc* function is considered to be the best reconstruction kernel, since it represents an ideal low-pass filter. In practice, however, it is difficult to convolve a discrete signal with the *sinc* kernel, because of its infinite support. Therefore practical reconstruction filters either approximate it or truncate it with an appropriate windowing function. Moreover, real-world signals can hardly be considered to be band-limited. As a consequence, practical resampling results in a loss of information. The first step in the traditional volume visualization pipeline is the discrete tomographic reconstruction of a rectilinear volume representation from the measured projections. Although there exist different algorithms for tomographic reconstruction, one of the most popular techniques is the Filtered Back-Projection algorithm. It first performs high-pass filtering on the measured projections and afterwards the samples at rectilinear grid points are computed by back-projecting the filtered signals. As the projections are acquired by measuring accumulated attenuation by a limited number of sensors, they are actually available as discrete representations of continuous projection functions. Therefore high-pass filtering is performed in discrete frequency domain, so the result is also a discrete function. In the back-projection phase, however, the rectilinear grid points are not necessarily projected exactly onto the discrete samples of the filtered projections. Therefore for back-projection resampling is necessary, which results in the *first* loss of information in the pipeline. The obtained rectilinear volume can be visualized by different rendering techniques. Using indirect methods, like the classical Marching Cubes algorithm [9], an intermediate geometrical model of an iso-surface is constructed from the volumetric model. This geometrical model is then interactively rendered by, for example, conventional graphics hardware. In contrast, Direct Volume Rendering (DVR) approaches, like ray casting or splatting directly render the volumetric model without any intermediate representation. In both cases an interpolation technique is applied to define data values between the rectilinear grid points. In other words, a resampling of the discrete volume

representation is required. This resampling results in the *second* loss of information in the traditional pipeline. In order to minimize the loss of information we propose to modify the traditional volume-rendering pipeline by simply removing an unnecessary resampling step. To render the underlying continuous phenomenon, data samples at arbitrary sample points need to be defined, and for shading computation the corresponding gradients need to be determined. As shown, both tasks can be solved using directly the filtered projections. This eventually leads to an alternative projection-based volume representation. Thus, there is no need to compute samples at regular grid points by discrete tomographic reconstruction, and as a consequence one resampling step is unnecessary. Traditional direct volume-rendering methods rely on such an intermediate grid representation, so in this sense they are in fact indirect. In contrast to that we presented *Direct Direct Volume Rendering*, a high quality volume rendering of projection-based volumetric data. In most of the practical volume-rendering applications, especially in 3D medical imaging, the input data is usually generated from measured projections by using tomographic reconstruction. The output of tomographic reconstruction is a discrete (or sampled) representation of the underlying continuous phenomenon. The samples are conventionally generated on rectilinear grid points. The rectilinear grid has several advantages. For example, the sampled signal can be represented by 3D arrays, implicitly storing the locations of the samples. Furthermore, the neighborhood of a certain sample can be efficiently addressed, which is important for many volume-processing or volume-rendering algorithms.

Nevertheless, in order to render the underlying continuous 3D function, data values need to be defined also between the rectilinear grid points. The *sinc* kernel as ideal reconstruction filter is impractical because of its infinite extent. In practice it is approximated by filters of finite support. Generally, the wider the support of the reconstruction filter, the higher the quality of the reconstruction. On the other hand, the wider the support of the filter, the higher the computational cost of a spatial-domain convolution. As the practical filters only approximate the ideal low-pass filter they result in either aliasing or smoothing, which can be interpreted as a loss of information.

For frequent resampling tasks, like rotation, or upsampling, frequency-domain techniques can be alternatively applied. In frequency domain, it is exploited that a computationally expensive spatial-domain convolution is replaced by a simple multiplication. Although the frequency-domain resampling methods generally provide higher accuracy than spatial-domain methods do, they assume that the new samples to be computed are also located at regular grid points.

7.2 Mathematical Preliminaries

The set of projections is referred to as the Radon transform of the original signal. Therefore the tomographic reconstruction is, in fact, the inversion of the Radon transform. The inversion can be performed by using the classical Filtered Back-Projection algorithm, which is based on the Fourier projection-slice theorem. Although there exist alternative tomographic reconstruction techniques like algebraic or statistical ones, Filtered Back-Projection is still the most popular method used in commercial CT scanners. In order to avoid a lossy resampling step in the traditional volume-rendering pipeline, we directly use the tomographic inversion in order to reconstruct the underlying function at arbitrary sample positions. Therefore we do not generate an intermediate rectilinear volume representation, but we directly process the filtered projections as an alternative volume representation. Using this gridless or projection-based volume-rendering approach as a new paradigm, the same accuracy can be ensured at all the sample positions. In contrast, using the traditional grid-based approach, accurate samples are available only at the grid points, while the accuracy of intermediate samples depends on the quality of the applied imperfect reconstruction filter.

7.3 D^2VR

We presented D^2VR based on a raycasting approach. In order to perform raycasting the underlying 3D volumetric function needs to be reconstructed

at arbitrary resampling locations. In case the data is given on a rectilinear grid the reconstructed function value is computed from a close neighborhood of samples. In contrast to that, for raycasting directly performed on the filtered projections the reconstructed function value is computed from the filtered projections at the corresponding positions. Furthermore, gradients at these resample locations need to be determined in order to perform shading.

7.3.1 Data reconstruction

Data reconstruction from projection-based volumetric data requires Filtered Back-Projection. For simplicity we illustrated the Filtered Back-Projection in 2D based on a computed tomography scanning process using orthographic projection, and later extended it to three dimensions.. Parallel projections are taken by measuring a set of parallel rays for a number of different angles. A projection is formed by combining a set of line integrals. The whole projection is a collection of parallel ray integrals as is given by $P_\theta(t)$ for a constant angle θ. The line integrals are measured by moving a x-ray source and detector along parallel lines on opposite sides of the object. In order to reconstruct the underlying density function we proposed to use the discrete Filtered Back-Projection as a resampling scheme. According to discrete Filtered Back-Projection the density function can be reconstructed from a fixed number of projections by the following formula:

$$f(x,y) \approx \tilde{f}(x,y) = \frac{\pi}{K} \sum_{i=1}^{K} Q_{\theta_i}(x \cos\theta_i + y \sin\theta_i) \tag{7.1}$$

The Filtered Back-Projection algorithm is conventionally used for discrete tomographic reconstruction in order to obtain a rectilinear representation of the original density function. This intermediate representation is then usually resampled by many volume visualization algorithms. As shown in this thesis it is unnecessary to generate an intermediate rectilinear representation by discrete tomographic reconstruction. In fact it should be avoided to reconstruct an intermediate rectilinear representation, as each resampling step usually causes a loss of information. By avoiding the unnecessary intermediate grid

representation, the quality of reconstruction can be improved. Previous reconstruction techniques assume that accurate samples are available at the grid points. In practice these samples are obtained by tomographic reconstruction. In order to maintain the same accuracy at any arbitrary sample location, we suggested to apply the Filtered Back-Projection to reconstruct the density value.

7.3.2 Derivative Estimation

In order to process or render volumetric data in many cases derivatives of the original density function are necessary. For example, for volume rendering the estimated gradients are used as surface normals to perform shading. In case of a grid-based representation the straightforward way is to estimate the derivatives from a certain voxel neighborhood. To determine the gradient, common methods, such as intermediate difference gradient, central difference gradient, or higher order gradient estimation schemes are applied. In our case, computing the derivatives from a certain 3D neighborhood of samples requires to perform a large number of back projections. Especially for higher order gradient estimation schemes, which need a large neighborhood of samples, the computational costs would be significantly high. However, the Filtered Back-Projection reconstruction scheme can also be exploited to compute derivatives. We first presented derivative estimation in two dimensions and extended the gradient estimation scheme to three dimensions later. It was shown, that the partial derivatives of the density function are equal to the sum of scaled partial derivatives of the projections. Therefore applying Newton's difference quotient directly on the filtered projections is equivalent to applying Newton's difference quotient on the 2D density function $f(x, y)$. Moreover, any higher order derivative can be obtained by applying Newton's difference quotient multiple times.

We showed that using Filtered Back-Projection for gradient estimation leads to higher accuracy than using the traditional central differences gradient estimation scheme. In order to calculate the central difference gradient at an arbitrary sampling point six additional samples have to be interpolated.

As interpolation usually causes loss of information, the introduced errors are accumulated in the estimated gradients. In contrast, using Filtered Back-Projection, the density values at the additional sample points are as accurate as at the grid points. Therefore, no interpolation error is introduced.

7.3.3 Hierarchical Space Partitioning

For almost all volumetric processing approaches a hierarchical space partitioning data structure is essential for efficient processing. The performance gain which can be achieved with such a data structure mainly depends on its granularity. An octree is one of the most widely used data structures for organizing three-dimensional space. An octree is based on the principle of hierarchical space partitioning. Each node of the octree represents a cuboid cell. In a min-max octree for volumetric data each node contains the minimum and maximum of the enclosed data. The minimum and maximum value of an octree cell, in case of grid-based volumetric data, depends on the used data reconstruction method. The most widely used data reconstruction method is trilinear interpolation. The convex nature of trilinear interpolation ensures that all function values within a cuboid are bounded by the maximal and minimal values at the grid positions.

This convexity condition does not hold for reconstruction based on the Filtered Back-Projection. However, it is still possible to generate an min-max octree for projection-based volumetric data. Consider an octree cell C projected onto all filtered projections P_{θ_i}, see Figure 4.5. The resulting projections of C are referred to as C_{θ_i}. According to Equation 3.17 an upper bound of the maximum value contained in the octree cell C can be determined by:

$$\max C_{\tilde{f}} \leq \frac{\pi}{K} \sum_{i=1}^{K} \max C_{\theta_i \tilde{f}} \tag{7.2}$$

where

$$C_{\tilde{f}} = \left\{ v = \tilde{f}(\vec{x}) | \vec{x} \in C \subseteq \Re^3 \right\}$$

are all possible function values within the enclosing cuboid of cell C and

$$C_{\theta_i \tilde{f}} = \left\{ v = Q_{\theta_i}(\vec{t}) | \vec{t} \in C_{\theta_i} \subseteq \Re^2 \right\}$$

are all projection values within the enclosing rectangle of the projected cell C onto projection P_{θ_i}. Analogously a lower bound of the minimum of the octree cell C can be determined by:

$$\min C_{\tilde{f}} \geq \frac{\pi}{K} \sum_{i=1}^{K} \min C_{\theta_i \tilde{f}} \tag{7.3}$$

The algorithm to generate an octree, using a bottom-up approach, for projection-based volumetric data is as follows:
Starting with a root cell, which encloses the entire volumetric function defined by the projections, we recursively subdivide the octree cells. Once we reach a desired octree depth we compute the maximum respectively the minimum using Equation 7.2 and 7.3. These values are then propagated to the higher levels. Instead of propagating these minimums and maximums to the higher levels, it would have been also possible to compute the minimum and maximum directly for each of these higher levels using Equation 7.2 and 7.3. However, this would lead to a much less efficient approximation. Although this space partitioning is a conservative approximation, it was shown that it works very well in practice.

7.4 Implementation

The implementation was split into two major parts. First the implenentation of a Computed Tomography scanner simulation was described. The simulation is able to handle three different types of density functions. Analytically defined continuous density functions, so called phantoms, and finally high resolution rectilinear grids were used. Secondly, an implementation of D^2VR based on a raycasting approach was described.

We implemented a CPU-based as well as a GPU-based prototype for orthographic and perspective projection. The CPU implementation is based

on a raycasting approach. For each pixel of the image plane, rays are cast through the volumetric space enclosed by the filtered projections. At each resample location the underlying 3D density function is reconstructed according to Equation 3.17 and gradient estimation is done using Equation 3.32. The final color and opacity of the image pixel is determined by the over-operator [17] in front-to-back order. The GPU version is implemented in C++ and Cg using OpenGL. We are aware that the current Cg compiler does not always produce optimal code. We did not manually optimize the code by using assembly as this is a proof of concept implementation. We implemented D^2VR on the GPU (NVidia GForce 6800 GT, 256 MB) utilizing texture-based volume rendering with view-aligned slices in combination with filtered projection textures. All the filtered projections are linearly stored in a 3D texture. Basically we compute texture slices parallel to the viewing plane. As the newest NVidia hardware supports dynamic branching, the slices are directly computed in a *for*-loop using Equation 3.17 from all the filtered projections. The same is done for the gradient computation using Equation 3.32. The volumetric space enclosed by the filtered projections is thereby cut by polygons parallel to the viewing plane. The polygons are then projected by the graphics hardware onto the image plane and composited using alpha blending. In order to accelerate the rendering process, we utilized the octree described in Section 4.2.2. We project all visible octree cells onto the image-plane. The resulting z-buffer image is then used for early z-culling, a capability of modern hardware, implementing empty space skipping. In average, for example, we measured for 128 projections (128^2 sized) using a 256^2 view-port four seconds per frame for an iso-surface rendering. Further optimizations, such as early ray termination, are not employed due to the lack of graphics hardware support.

7.5 Results

In order to show the differences between projection-based and grid-based data reconstruction and gradient estimation we simulated a Computed Tomography scanner. We scanned several different density functions, such as

the Marschner & Lobb function, a phantom, a carp, a human head, and a stag beetle. The Marschner & Lobb function is analytically defined. For the other data sets, an analytical representation is not given, therefore we took high resolution grids in combination with trilinear interpolation as ground truth. The Marschner & Lobb function is scanned taking 64 projections, each projection with a resolution of 64^2. From these projections a grid is reconstructed, using the same amount of samples (64^3). Additionally we also reconstructed a grid with eight times more samples (128^3). Furthermore we computed an iso-surface directly from the analytical Marschner & Lobb function. The differences between the analytical value and the reconstructed value using trilinear interpolation on the grid (64^3) and the differences between the analytical value and the reconstructed value using Filtered Back-Projection were shown. To encode the data reconstruction error on the iso-surface a color coding was applied. Furthermore, the orientation differences in degrees between the analytically computed gradients and the estimated gradients using central difference gradient estimation and the orientation differences in degrees between the analytically computed gradients and the estimated gradients using our new projection-based gradient estimation method were shown. Finally a comparison of iso-surface renderings of the Marschner & Lobb function was shown. Furthermore, we rendered a stag beetle a human head and a carp The differences between rendering from projection-based and grid-based volumetric data were presented. Additionally for all the shown data sets we reconstructed a 128^3 grid from the filtered projections. From this grid as well as from the filtered projections we also reconstructed a rotated grid and showed the root mean squared data reconstruction error with respect to the analytical function respectively to the corresponding high resolution grid in combination with trilinear interpolation.

7.6 Conclusion

In this thesis a new direct volume-rendering paradigm has been introduced. It has been shown that volumetric raw data measured as a set of projections can

be directly rendered without generating an intermediate grid-based volume representation by using tomographic reconstruction. As our method avoids an unnecessary and lossy resampling step, it provides much higher image quality than traditional direct volume-rendering techniques do. Furthermore, our novel projection-based gradient estimation scheme avoids the accumulation of interpolation errors. Traditional methods ensure accurate samples at the grid points, while the accuracy of intermediate samples strongly depends on the quality of the applied interpolation method. In contrast, our approach provides an accurate data value for an arbitrary sample position.

In order to accelerate D^2VR a hierarchical data structure for empty space skipping was presented. Furthermore, as the filtered projections can be interpreted as 2D textures, conventional graphics cards can be exploited to efficiently accumulate the contributions of the filtered projections to view-aligned sampling slices. Inspite of these optimizations D^2VR is still slower than previous presented direct volume rendering methods. On the other hand, in the last two decades, a huge research effort was spent to accelerate traditional direct volume rendering, which was far from interactive in the beginning. As the new introduced approach tries to open a completely new research direction, it would not be fair to compare the current performance of the presented technique to that of a well developed technology.

Acknowledgements

Cheers!

Sören Grimm

My first thanks, go to my supervisor Sören Grimm, for his support, help, and constant confidence in my work. Furthermore, I want to thank Eduard Gröller, our beloved master, as well as Stefan Bruckner, for the valuable and inspiring discussions we had, about the topic and beyond. Last but not least, I want to thank Balázs Csébfalvi, for his important suggestions and for being a perfect tourist guide.

Bibliography

[1] M. Artner, T. Möller, Ivan Viola, and Eduard Gröller. High-quality volume rendering with resampling in the frequency domain. In *Proceedings of joint IEEE VGTC Eurographics Symposium on Visualization (EuroVis)*, 2005. to appear.

[2] Jr. C. V. Jakowatz and A. C. Kak. Computerized tomography using x-rays and ultrasound, 1976.

[3] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 91–98, 1994.

[4] Q. Chen, R. Crownover, and M.Weinhous. Subunity coordinate translation with Fourier transform to achieve efficient and quality three-dimensional medical image interpolation. *Med. Phys.*, 26(9):1776–1782, 1999.

[5] R. Cox and R. Tong. Two- and three-dimensional image rotation using the FFT. *IEEE Transactions on Image Processing*, 8(9):1297–1299, 1999.

[6] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[7] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications, Vol.8, No.3*, pages 29–37, 1988.

[8] A. Li and K. Mueller. Methods for efficient, high quality volume resampling in the frequency domain. In *Proceedings of IEEE Visualization*, pages 3–10, 2004.

[9] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH '87)*, pages 163–169, 1987.

[10] T. Malzbender. Fourier volume rendering. *ACM Transactions on Graphics, Vol.12, No.3*, pages 233–250, 1993.

[11] S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of IEEE Visualization*, pages 100–107, 1994.

[12] D. Mitchell and A. Netravali. Reconstruction filters in computer graphics. In *Proceedings of SIGGRAPH*, pages 221–228, 1988.

[13] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A comparison of normal estimation schemes. In *Proceedings of IEEE Visualization*, pages 19–26, 1997.

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

[15] L. T. Niklason, B. T. Christian, L. E. Niklason, D. B. Kopans, D. E. Castleberry, B. H. Opsahl-Ong, P. J. Slanetz C. E. Landberg, A. A. Giardino, R. Moore, D. Albagli, M. C. DeJule, P. F. Fitzgerald, D. F. Fobare, B. W. Giambattista, R. F. Kwasnick, J. Liu, S. J. Lubowski, G. E. Possin, J. F. Richotte, C. Y. Wei, and R. F. Wirth. Digital tomosynthesis in breast imaging. *Radiology*, 205:399–406, 1997.

[16] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall Inc., Englewood Cliffs, 2nd edition, 1989.

[17] T. Porter and T. Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, 1984.

[18] 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, 1992.

[19] J. Russ. *The Image Processing Handbook*. CRC Press, 1992.

[20] T. Theußl and E. Gröller. Mastering windows: Improving reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 101–108, 2000.

[21] R. Tong and R. Cox. Rotation of NMR images using the 2D chirp-z transform. *Magn. Reson. Med.*, 41:253–256, 1999.

[22] M. Unser, P. Thevenaz, and L. Yaroslavsky. Convolution-based interpolation for fast, high-quality rotation of images. *IEEE Transactions on Image Processing*, 4(10), 1995.

[23] L. Westover. Footprint evaluation for volume rendering. *Computer Graphics (Proceedings of SIGGRAPH '90)*, pages 144–153, 1990.

[24] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA volume splatting. In *Proceedings of IEEE Visualization 2001*, pages 29–36, 2001.

List of Figures

List of Tables