TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

D I P L O M A R B E I T

High-Quality Volume Rendering with Resampling in the Frequency Domain

ausgeführt am

Institut für Computergraphik und Algorithmen
der Technischen Universität Wien

unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Institut 186 für Computergraphik und Algorithmen,
und
Dipl.-Ing. Ivan Viola
als verantwortlichen mitwirkenden Universitätsassistenten,
sowie unter der Mitbetreuung von
Dr. Torsten Möller, Assistant Professor,
School of Computing Science,
Simon Fraser University, Vancouver, Canada,

erstellt von

Martin Artner
Mühlbach 26
3972 Bad Großpertholz

7. Dezember 2004 _____
Datum

_____
Unterschrift

Martin Artner

High-Quality Volume Rendering with Resampling in the Frequency Domain

Master's Thesis

under the supervision of

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Dipl.-Ing. Ivan Viola
Institute of Computer Graphics and Algorithms
Vienna University of Technology

Dr. Torsten Möller, Assistant Professor
School of Computing Science
Simon Fraser University, Vancouver, Canada

To my parents Maria and Franz

Abstract

This thesis introduces a volume rendering technique that is conceptually based on the shear-warp factorization. The novelty in our approach is that we perform the shear transformation entirely in the frequency domain. A compensation for the direction dependent sampling distance along the viewing rays, which is present in the standard shear-warp approach, is developed. This compensation is also carried out in the frequency domain and is capable of producing freely selectable sampling distances. The accurate scaling of the volume slices is achieved by using the zero padding interpolation property. Finally, a high quality gradient estimation scheme is presented which uses the derivative theorem of the Fourier transform. Experimental trials have shown that the presented method can outperform established algorithms in the quality of the resulting images. Especially in the case when the dataset was acquired according to the sampling theorem, the presented method is capable of a perfect reconstruction of the original function.

Kurzfassung

In dieser Diplomarbeit wird eine Volumen-Rendering Methode, die konzeptionell auf der Shear-Warp Faktorisierung basiert, vorgestellt. Die Innovation in unserem Ansatz besteht darin, dass die Shear-Transformation vollständig im Fourier-Raum durchgeführt wird. Eine Kompensation für die richtungsabhängige Abtastrate auf den Blickstrahlen wird vorgestellt. Diese Kompensation wird ebenfalls im Fourier-Raum durchgeführt und kann beliebige Abtastraten produzieren. Hochgenaues Skalieren von Volumen-Schichten erreichen wir durch die Verwendung der Zero-Padding-Interpolationseigenschaft. Zuletzt wird ein hoch qualitatives Gradienten-Berechnungsschema eingeführt, welches das Ableitungs-Theorem der Fourier Transformation benutzt. Experimentelle Versuche haben gezeigt, dass die hier präsentierte Methode bereits etablierte Algorithmen in der Qualität überbieten kann. Speziell im Falle, dass der verwendete Datensatz unter Einhaltung des Sampling-Theorems aquiriert wurde, ist die hier präsentierte Methode fähig, die originale Function perfekt zu rekonstruieren.

Acknowledgments

I would like to thank:

- My family for their endless encouragement and support throughout my studies.
- My supervisor, Torsten Möller, for giving me the opportunity to work with him in Vancouver for the last 12 months, for beeing an infinite source of new ideas, and for pushing me where I wouldn't have gone by myself.
- My supervisors at my home university, Meister Eduard Gröller, Ivan Viola and Thomas Theußl, for their support with my work, their trust in me and for making my stay in Vancouver possible.
- The entire GrUVi lab, especially Steven Bergner, Andrew Clements, Eric Dagenais, Ramsay Dyer, Alireza Entezari, Zhe Fang, David Fayegh, Mayu Ishida, Rong Liu, Matt Olson, Haris Widjaya and Hamid Younesy for beeing very nice, open for fun and through that making my stay in Vancouver very pleasant and enjoyable.
- Helene and Manfred Artner for their generous financial support and encouragement.
- All of my friends and fellow students at the universities in Vienna, Madrid and Vancouver for making my studies not only a great academic experience but also a very entertaining one.

Contents

Chapter 1

Introduction

There is no shame in not knowing;
the shame lies in not finding out.

Russian proverb

Volume rendering has become a very fruitful area of research over the last decades. Its applications reach from medical imaging to scientific visualization of data. Fast and, especially, accurate display of the data can be crucial for decisions made by surgeons or researchers. The intention of this thesis is to introduce a rendering technique that provides results in superior quality compared to those obtained from existing methods.

1.1 Motivation

The basic challenge in volume rendering is to display data that is given on a three-dimensional grid from arbitrary directions. The volume rendering algorithms can be classified into two categories, image order and object order based algorithms. In this work we focus on the image order algorithms that cast for each pixel of the final image one ray through the scene. Since rays coming from an arbitrary direction usually don't hit the sample points exactly, some form of interpolation is needed, in order to calculate data along the rays. Although a lot of research has been done in this area, the interpolation filters used in standard methods are only approximations to the perfect reconstruction of the *sinc* filter kernel. Therefore visible artifacts are introduced in the final images.

The use of the Fourier transform in volume rendering was first proposed in 1990 by Dunne et al. [7] followed by Malzbender [21], Levoy [19], Totsuka and Levoy [34]. All these approaches focus on the *Projection Slice Theorem* [31] that states that slicing in the frequency domain equals a projection in the spatial domain. Therefore to composite a given volume onto a

viewing plane a two dimensional slice is extracted of the three dimensional frequency domain representation of the volume. This slice has to be in the same orientation as the chosen viewing plane. The inverse Fourier transformation of this slice leads to the resulting image. If the projected volume is of size N^3, the extraction of the slice and the following inverse Fourier transform has a complexity of $\mathcal{O}(N^2 logN)$ as compared to the projection in spatial domain with a complexity of $\mathcal{O}(N^3)$. The forward Fourier transform, $\mathcal{O}(N^3 logN)$, of the volume has to be done only once, and is considered as a preprocessing step. The gain in rendering speed comes with the drawback of a low image quality. Resulting images are similar to X-ray imaging.

In our proposed method we create regular ray casting like images [17] that feature occlusion and use transfer-function specification to map the density values to optical properties. To achieve this we had to turn away from the projection slice theorem and use other properties of the Fourier transform, that have found wide applications in 1D and 2D signal processing. In particular we exploit the *Packing Theorem*, which is also known as zero padding, to perform zooming operations, *Shifting Theorem* to offset the data along the coordinate axis and the *Derivative Theorem* to calculate derivatives of the input signals.

The Shear-Warp Factorization introduced by Lacroute et al. [14] was designed to be one of the fastest software based rendering algorithms. This method uses three copies of the volume, represented as stacks of slices, each aligned to one of the main coordinate axis. The stack of slices which is the closest to be perpendicular to the viewing direction is used in a given rendering pass. The viewing transformation of the volume from the object space into the image space is factorized into a permutation, a shear and a warp transformation. The permutation transformation is already performed by choosing one of the three stacks. The shear transformation uses 2D interpolation within the slices as compared to computationally more expensive 3D interpolation schemes [17]. Additionally no rotation is applied to the slices, just shifts parallel to the coordinate axis are done. These shears can be done in the frequency domain by applying the time shifting theorem onto both axes of the two-dimensional slices. To create a high quality zoom, the slices can be scaled by zero padding in the frequency domain. The shear transforms the volume into the sheared object space, from which the slices are projected onto the intermediate image plane. The final image is created by performing the remaining warping transformation on this 2D frame buffer.

We suggest a rendering algorithm based on the shear-warp factorization carrying out the shear transformation in the frequency domain. To further improve image quality a modification of the standard shear-warp approach is introduced to calculate intermediate slices in order to obtain a selectable and viewing direction independent sampling distance along the viewing rays. For high quality gradient estimation three copies of the volume dataset are created. Each of them is differentiated in the direction of one of the three main coordinate axis by applying the derivative theorem. These volumes are then processed through the proposed rendering pipeline

to the sheared object space, where they are combined to form a gradient vector field. The gradient vector field is used for calculating lighting effects on the data slices before they are blended into the intermediate image.

1.2 Goals

The intention of this thesis is to bring high quality signal processing methods that are based on the Fourier transform to the volume rendering pipeline. Therefore the following goals were set:

- Show how the Fourier transform and its theorems can be used to interpolate and differentiate functions provided as sampled data with very high quality.
- Combine the approach of the shear-warp factorization with Fourier transform based methods to create a volume rendering method that can perform interpolation with *sinc* filter quality.
- Introduce high quality gradient estimation by exploiting the derivative theorem of the Fourier Transform.

1.3 Structure

This document is structured in the following way:

Chapter 2 provides a review of the *Related Work* on which this thesis is based. This includes an introduction to volume rendering techniques and the role of the Fourier transform in this area. Additionally a brief overview of the theory of sampling and interpolation is given.

Chapter 3 introduces the *Fourier Transform* and its theorems which are used in this work.

Chapter 4 describes in detail the algorithm of the *Shear-Warp Factorization in the Fourier Domain*.

Chapter 5 provides *Implementation* details to discuss issues that arise when dealing with data in the Fourier domain.

Chapter 6 presents the rendering *Results* and provides qualitative analysis of the images obtained by the new technique in comparison to traditional methods.

Chapter 7 gives a *Summary* of the presented work.

Chapter 8 finally draws *Conclusions* and gives suggestions for *Future Work*.

1.4 Notation

The mathematical notation used throughout this thesis is introduced here. The symbols for the number sets are $\mathbb{N}$ for non-negative integers, $\mathbb{Z}$ for integers, $\mathbb{R}$ for real and $\mathbb{C}$ for complex numbers. For complex numbers the lower case i represents the complex unit, i.e. $i^2 = -1$, and an asterisk like $*$ indicates the complex conjugate.

Continuous signals are represented using brackets as in $x(t)$, while discrete signals use square brackets as in $x[n]$. Finite discrete signals have only a finite number of non-zero elements. Therefore the signal $x[n]$ has length N if beside a sequence of N elements all remaining elements of $x[n]$ are zero. The sequence of the N non-zero elements in a finite discrete signal is assumed to start at the index $n = 0$, that signifies that $x[n] = 0$ for $n < 0$ and $n > N - 1$.

Periodic signals are indicated with a circumflex on the top of the function name as in $\hat{x}(t)$ for continuous periodic signals and $\hat{x}[n]$ for discrete periodic signals. Unless otherwise stated, continuous periodic signals have a period of 2π. To express that a signal has a specific length, that length is included as a subscript in capital letters after the signal name. As an example $x_N[n]$ is a signal with length N and $\hat{x}_N[n]$ is a signal with period N.

Lower case letters as signal names indicate spatial domain signals, capital letters stand for the frequency domain signals. Corresponding names of signals from the spatial and the frequency domain indicate transform pairs related through the Fourier transform. Whereas the spatial domain signal represents the signal itself and the frequency domain signal captures the frequency spectrum representation of the original signal. The transform pair relation is also expressed by a two-sided arrow like $\hat{f}[n] \Longleftrightarrow \hat{F}[\mu]$.

Higher dimensional signals are indicated by a number of function arguments that correspond to the dimensionality of the signal as in $f[l, m, n]$ for a three dimensional signal. If the signal has a specific length in each dimension, that length is included as a subscripts as in $f_{LMN}[l, m, n]$.

Because of the separability property of the Fourier transform, multi-dimensional signals can possibly be transformed to the frequency domain in only a selected number of directions, where the remaining directions stay in the spatial domain. These resulting signals of these operations are indicated by a lower case function name and the index variables of the directions in the frequency domain are labeled with Greek letters. For example the signal $\hat{f}_{LMN}[l, m, n]$ is Fourier transformed in l direction into $\hat{f}_{LMN}[\lambda, m, n]$.

Unless otherwise stated, it is assumed that the function values of spatial domain signals are real and the function values of the frequency domain signals are complex. Multi-dimensional signals which have some axes in the spatial domain and some axes in the frequency domain are assumed to be complex.

Chapter 2

State of the Art in Volume Rendering

Inventions reached their limit long ago, and I see no hope for further development.

Julius Frontinus, 1st century A.D.

The intention of this thesis is to bring new aspects of the Fourier transform to the field of volume rendering. Reconstruction of discrete signals is an important part of many volume rendering algorithms, therefore a brief introduction to the mathematical basics of *Sampling and Reconstruction* is given. The next section provides an overview of *Optical Models for Direct Volume Rendering* followed by an introduction to the most common *Volume Rendering Techniques*. The last section deals with the role of the *Fourier Transform in Volume Rendering*, which is mainly an application of the projection slice theorem of the Fourier transform.

2.1 Sampling and Reconstruction

The data used in volume rendering is usually given as sample points on a particular grid. In order to be able to render the volume from an arbitrary angle, interpolation within these sample points is necessary. The basic concepts and requirements of this process are addressed in this section.

2.1.1 Special Functions

There are several functions used in sampling theory to simplify the notation of mathematical equations. The most common ones that are necessary to follow the ideas in this thesis are briefly introduced here.

Dirac delta

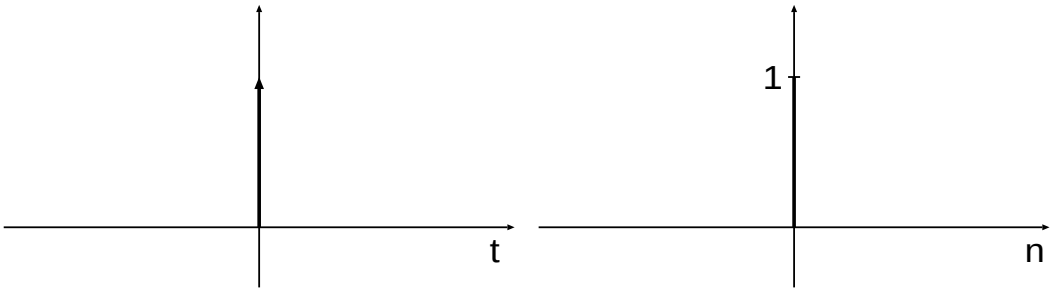
$\delta(t)$ is the unit impulse function or Dirac delta function, see Figure 2.1(a), defined as

$$\delta(t) = 0, t \neq 0 \tag{2.1}$$

$$\int_{-\infty}^{+\infty} \delta(t)dt = 1 \tag{2.2}$$

For discrete signals, also known as the unit sample sequence, see Figure 2.1(b),

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \tag{2.3}$$



(a) (b)

Figure 2.1: (a) The continuous impulse $\delta(t)$ and (b) the discrete impulse $\delta[n]$.

Impulse Train

The *shah* function $\text{Ш}(t)$ is a continuous impulse train, which is used to relate continuous and discrete signals, Figure 2.2.

$$\text{Ш}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n) \tag{2.4}$$

Unit Step

$H(t)$ is called the unit step function, see Figure 2.3(a), defined as

$$H(t) = \begin{cases} 1 & t > 0 \\ \frac{1}{2} & t = 0 \\ 0 & t < 0 \end{cases} \tag{2.5}$$

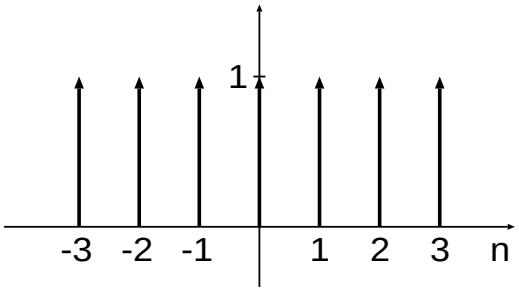
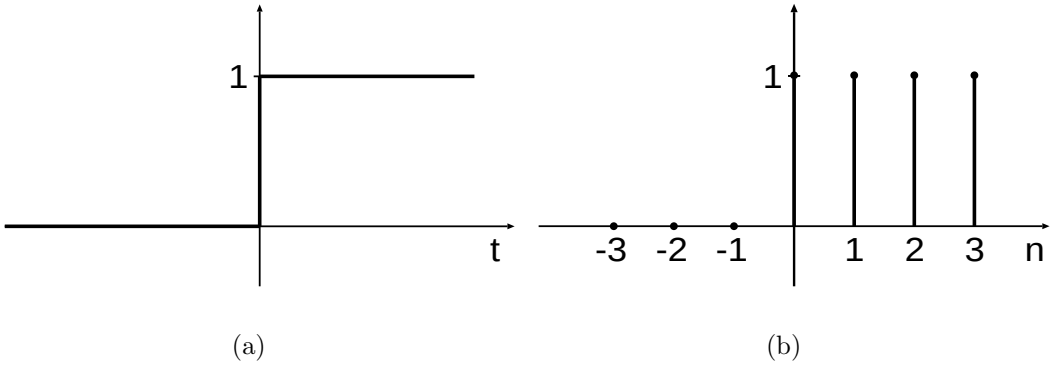


Figure 2.2: The *shah* function $\text{Ш}(t)$.

in the continuous case, also known as the Heaviside function. The unit step for the discrete case, see Figure 2.3(b),

$$H[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{2.6}$$



(a) (b)

Figure 2.3: (a) Continuous unit step $H(t)$ and (b) discrete unit step $H[n]$.

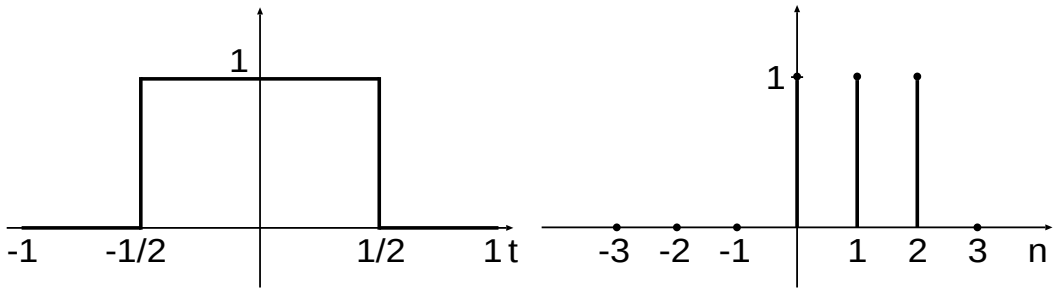
Box

The rectangle or box function is used in relating continuous and discrete signals, see Figure 2.4(a). The continuous version is defined as follows

$$\Pi(t) = \begin{cases} 1 & \mid t \mid < \frac{1}{2} \\ 0 & \mid t \mid > \frac{1}{2} \end{cases} \tag{2.7}$$

and the discrete box, see Figure 2.4(b),

$$\Pi_N(n) = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & otherwise \end{cases} \tag{2.8}$$



(a) (b)

Figure 2.4: (a) The continuous box $\Pi(t)$ and (b) the discrete box $\Pi_3[n]$.

Sinc

The *sinc* function is the Fourier transform pair to the continuous box function. It is used in relating continuous and discrete signals, see Figure 2.5. It is given by

$$sinc(t) = \frac{sin\pi t}{\pi t} \tag{2.9}$$

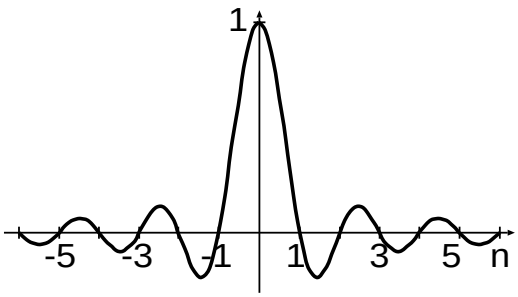


Figure 2.5: The *sinc* function $sinc(t)$.

2.1.2 Filter Theory

To make the continuous world around us accessible to a computer that is only capable of processing discrete numbers, is it inevitable to convert a given signal into numbers. The first step is to transform the continuous-time signal into a discrete-time signal. This means measuring the signal value in a periodic manner over the domain of the function and storing the obtained values in a list. How many samples are necessary to represent a signal perfectly is determined by its frequency spectrum. If a function has a very complex characteristics, then more samples are needed to cover the information. Shannon's sampling theorem, also known as Nyquist crite-

rion gives the general answer to the question of how many samples are necessary to completely represent the continuous function by its discrete samples. It states that the sampling frequency must be at least twice the maximum frequency in the sampled signal. If the sampling is done according to the Nyquist criterion, the original signal can be reconstructed by convolving with the *sinc* function. If the sampling rate is smaller, aliasing artifacts occur.

Figure 2.6 gives an overview of how sampling and reconstruction work. The left column represents the spatial domain and shows all the events that occur during the process, the right column does the same for the frequency domain. The first line introduces the band limited initial function Figure 2.6(a) and its spectrum Figure 2.6(b). In the second row the sampling *shah* function Figure 2.6(c) and its Fourier transform pair, also a *shah* function, but with reciprocal spacing as seen in Figure 2.6(d) are used to sample the input signal in spatial domain. The sampling is done in the spatial domain by multiplication of the input signal with the *shah* function which gives Figure 2.6(e) as a result. Multiplication in spatial domain is equivalent to a convolution in frequency domain. Therefore Figure 2.6(f) is the result of convolving the spectrum of the input signal with the Fourier transform pair of the sampling *shah* function.

We now have the sampled data available. In order to reconstruct the original input function, the situation of the first row in Figure 2.6 has to be restored. This can be done by multiplying with a box function in the frequency domain to cut out the central replica of the spectrum (see Figure 2.6(h)). The Fourier transform pair to the box is the *sinc* function (see Figure 2.6(g)). Multiplication with the box in frequency domain signifies convolution with the *sinc* function in spatial domain, to restore the original function.

If the signal is sampled with a lower frequency, the replicas of the frequency spectrum move closer. At the point when the sample frequency gets bellow the Nyquist frequency the replicas begin to overlap, see Figure 2.7. The overlapping areas of the replicas are summed up, which makes a reconstruction of the original function impossible. This error introduces aliasing artefacts in the reconstruction process.

An example for signal processing is the human voice, which has frequency components up to 20kHz. In order to follow the Nyquist criteria a sampling rate of 44.1kHz has been specified in the CD audio standard [11].

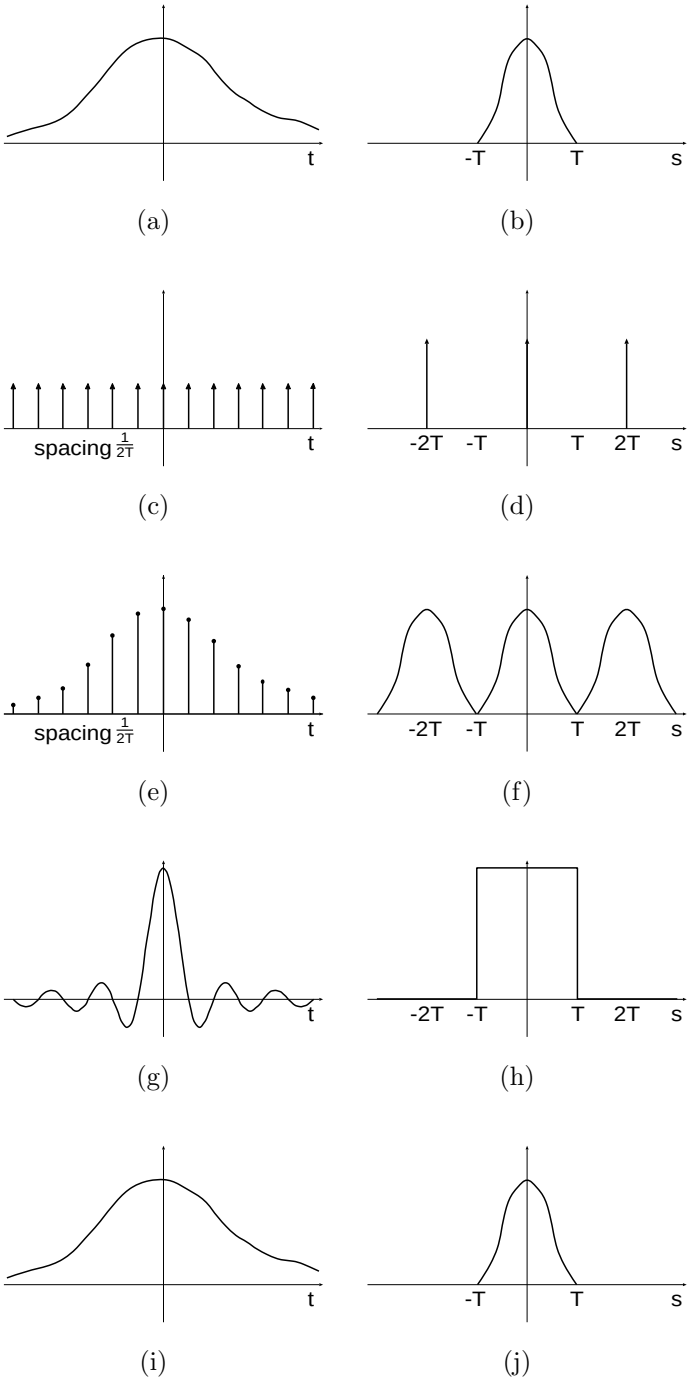


Figure 2.6: Sampling and reconstruction. The left column represents the spatial domain, whereas the right column displays the equivalent signals in the frequency domain. (a) initial function and (b) its frequency spectrum. (c) the sampling Shah-function and (d) its Fourier transform pair. (e) is the result after a multiplication of (a) and (c). Therefore (f) equals the convolution of (b) and (d). (g) the sinc function is a transform pair to (h) the box function. The multiplication of (f) with (h) creates (j) the spectrum of the original function. Hence a convolution of (e) with (g) leads to (i) the original function.

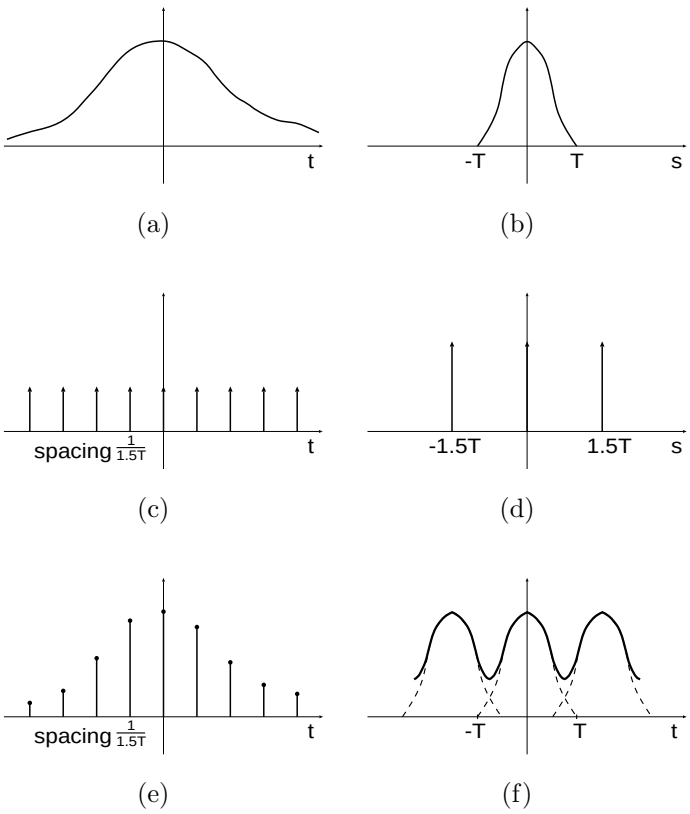


Figure 2.7: Sampling of a function below the Nyquist criteria. (a) and (b) show the same initial function and its spectrum as in Figure 2.6. (c) is the sampling Shah-function with a coarser spacing and (d) the Fourier transform pair which has a proportional finer spacing. The multiplication of (a) with (c) gives (e). The dual operation to the multiplication in spatial domain is the convolution in the frequency domain. Therefore the convolution of (b) with (d) gives (f). The closer spacing of the impulses in (d) leads to overlapping of the replicas of the spectrum in the frequency domain. The resulting function is indicated by a bold line in (f). The overlap of the replicas makes a perfect reconstruction of the initial function impossible. The introduced error is referred to as aliasing.

2.2 Optical Models for Direct Volume Rendering

The rendering equation (Equation 2.10) introduced by Kajiya [12] describes a physically based model of how the light emitted by one or several light sources propagates through a scene. It takes into account how rays get reflected on surfaces, the scattering of light from one surface to another, until the rays finally enter the observers eye.

$$I(x,x') = g(x,x')\Big[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\Big] \tag{2.10}$$

$I(x,x')$ is related to the intensity of light passing from point x' to point x, $g(x,x')$ is a "geometry" term, $\epsilon(x,x')$ is related to the intensity of emitted light from x' to x and $\rho(x,x',x'')$ is related to the intensity of light scattered from x'' to x via a patch of surface at x'.

As this equation contains integrals over the whole space and infinite recursions, it is practically not possible to evaluate it analytically. Therefore simplifications have to be made to make this approach more practical.

Blinn [3] first introduced a volume density scattering model to computer graphics. Equation 2.11 gives a form of the low-albedo volume rendering integral (VRI) used by Meißner et al. [25].

$$I_\lambda(x,r) = \int_0^L C_\lambda(s)\mu(s)e^{\left(-\int_0^s \mu(t)dt\right)}ds \tag{2.11}$$

The VRI analytically computes $I_\lambda(x,r)$ the amount of light of wavelength λ coming from ray direction r that is received at location x on the image plane. L is the length of the ray r, the volume is composed of particles with certain densities μ which receive light from all surrounding light sources and reflect this light towards the observer according to their specular and diffuse material properties. Additionally particles may also emit light on their own. So C_λ is the light of wavelength λ reflected and/or emitted at location s in the direction of r. To account for the higher reflectance of particles with larger densities, the reflected color is weighted by the particle density. The light scattered at s is then attenuated by the densities of the particles between s and the image plane. In general the VRI cannot be computed analytically as pointed out by Max [24], therefore practical volume rendering algorithms discretize the VRI into a series of intervals s_j with constant color $C_\lambda(s_j)$ and opacity $\alpha(s_j)$. Further the exponential function is approximated with the first two terms of the Taylor series expansion, which leads to Equation 2.12, known as the compositing equation [18].

$$I_\lambda(x,r) = \sum_{i=0}^{L/\Delta s} C_\lambda(s_i)\alpha(s_i)\cdot\prod_{j=0}^{i-1}(1-\alpha(s_j)) \tag{2.12}$$

It is also possible to look at compositing on a per-slice, instead of a per-ray basis. Each slice

represents a certain distance to the projection plane. The slices are composited either in a back to front (B2F) or a front to back (F2B) order onto the final image plane. B2F compositing is the naive solution, with a blending operator that is referred to as the "over" operator [28] (see Equation 2.13).

$$S_{i+1} = S_i(1 - \alpha_i) + C_i\alpha_i \tag{2.13}$$

S_i refers to the sum after i slices, and the voxel with the color C_i and the opacity α_i is composed over the already accumulated image.

A more sophisticated method is the F2B compositing which allows early ray termination that can accelerate the rendering of datasets which have many voxels with high opacity. During the F2B compositing the transparency α_{acc} of the already composited volume is accumulated for each pixel. If the α_{acc} value of a certain pixel reaches an value very close to 1.0 the compositing process for this pixel is stopped and the current color of the pixel is assumed to be the final result. Equation 2.14 shows how to add up the color component, and Equation 2.15 computes the accumulated opacity.

$$S_{i+1} = S_i + (1 - \alpha_{acc_i})C_i\alpha_i \tag{2.14}$$

$$\alpha_{acc_{i+1}} = \alpha_{acc_i} + (1 - \alpha_{acc_i})\alpha_i \tag{2.15}$$

S_i refers to the composited color of the first i slices, while C_i and α_i are the color and opacity of the currently processed sample. α_{acc_i} is the opacity of the composited slice packet after i slices.

2.3 Volume Rendering Techniques

This section gives an introduction to the main approaches in volume rendering. The field is separated into subsections according to the approach of image creation.

2.3.1 Image Order

Image Order techniques are focusing on the image plane as starting point of the algorithm. The basic idea is that for every pixel of the final image a ray is cast into the scene. The ray is sampled on its course and the simplified volume rendering equation, Equation 2.12, is evaluated.

Ray casting introduced by Levoy [17] is a very good representative of this approach, Figure 2.8 illustrates the concept. The computational complexity of these algorithms is governed

by the number of pixels in the final image.

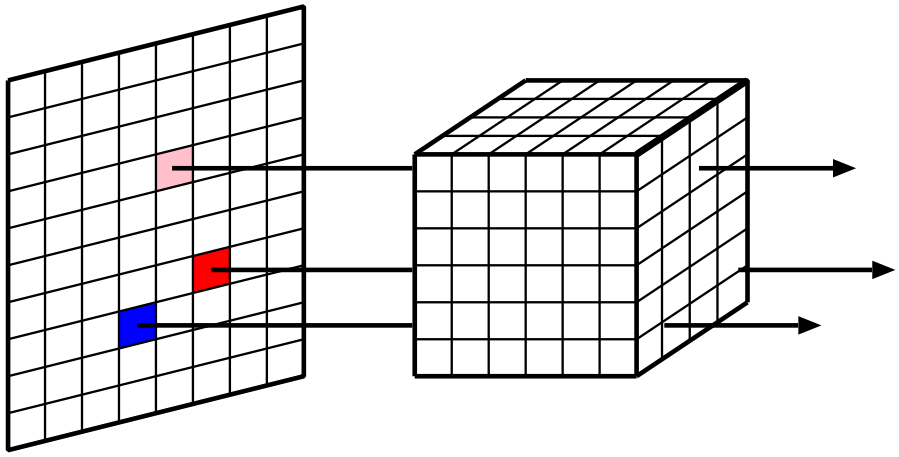


Figure 2.8: In image order techniques for each pixel in the final image one ray is cast through the scene. The final pixel color is gathered by evaluating the volume rendering integral along the ray.

2.3.2 Object Order

The volume data is the center of interest in *Object Order* methods. A rendering operation is performed for each voxel in the volume dataset. A representative of this approach is *Splatting* introduced by Westover [36]. Every voxel is seen as a particle and projected to the image plane, where it creates a footprint according to its color and opacity. Voxels closer to the image plane blend over the footprints left by voxels further away. This achieves an approximation to the simplified volume rendering equation. The aligned memory access can gain a significant acceleration in render time, compared to ray casting. The computational complexity of object order methods is predetermined by the number of voxels in the volume to be rendered. Figure 2.9 illustrates the concept of this method.

2.3.3 Hybrid Order

Image and object order approaches have several advantages and disadvantages. The aim of *Hybrid Order* techniques is to combine the advantages and to create a fast and relatively accurate rendering algorithm. The *Shear-Warp Factorization* introduced by Lacroute and Levoy [14] is the fastest known purely software based volume rendering algorithm.

The conceptual idea is to transform the volume into an intermediate coordinate system which is called the “sheared object space”. The definition of this space is that all viewing rays are parallel to the third coordinate axis. The transformation is illustrated in Figure 2.10 for the parallel projection. The horizontal lines represent slices of the volume data which

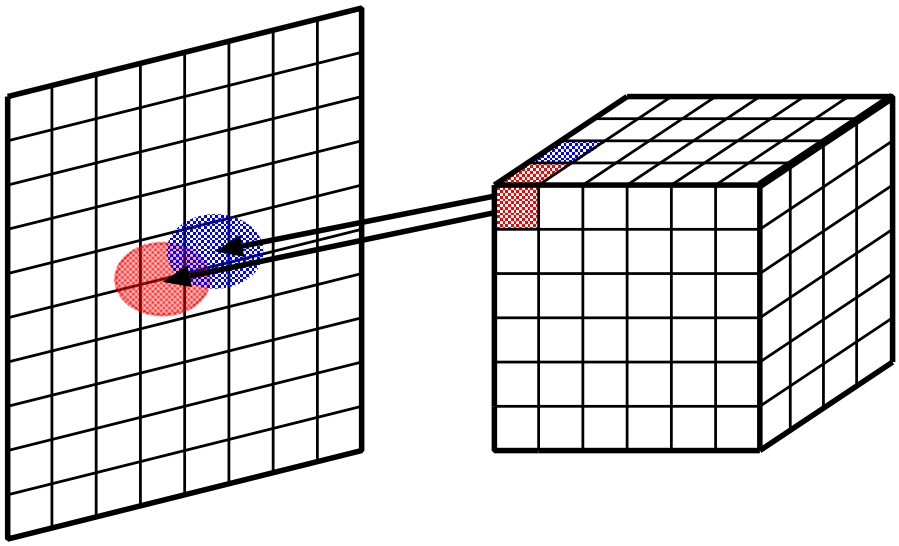


Figure 2.9: In object order techniques each voxel of the input volume is projected onto the image plane.

are intersected by the viewing rays. One stack of these slices representing the volume has to be stored for each coordinate axis. The stack most perpendicular to the viewing direction is selected and transformed in order to set the viewing rays perpendicular to the slices. They can now be composited into the intermediate image in a back-to-front order. The final warping step eliminates distortions in the intermediate image and performs eventual rotations around the viewing axis. A summary of these steps can be seen in Figure 2.11. The reason why this algorithm is considered a hybrid order technique is that voxel slices of the volume are composed into the intermediate image, which is considered an object order operation. Afterwards for each pixel in the final image the corresponding position in the intermediate image is computed which gives the final pixel color. This second warping step is the image order component.

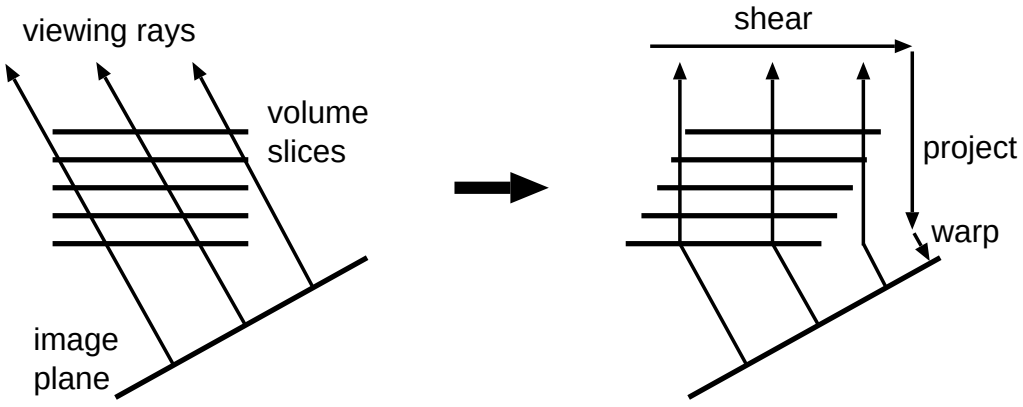


Figure 2.10: In the shear-warp factorization approach the volume is tranformed to the sheared object space by translating each slice in a way that the viewing rays become perpendicular to the slices.

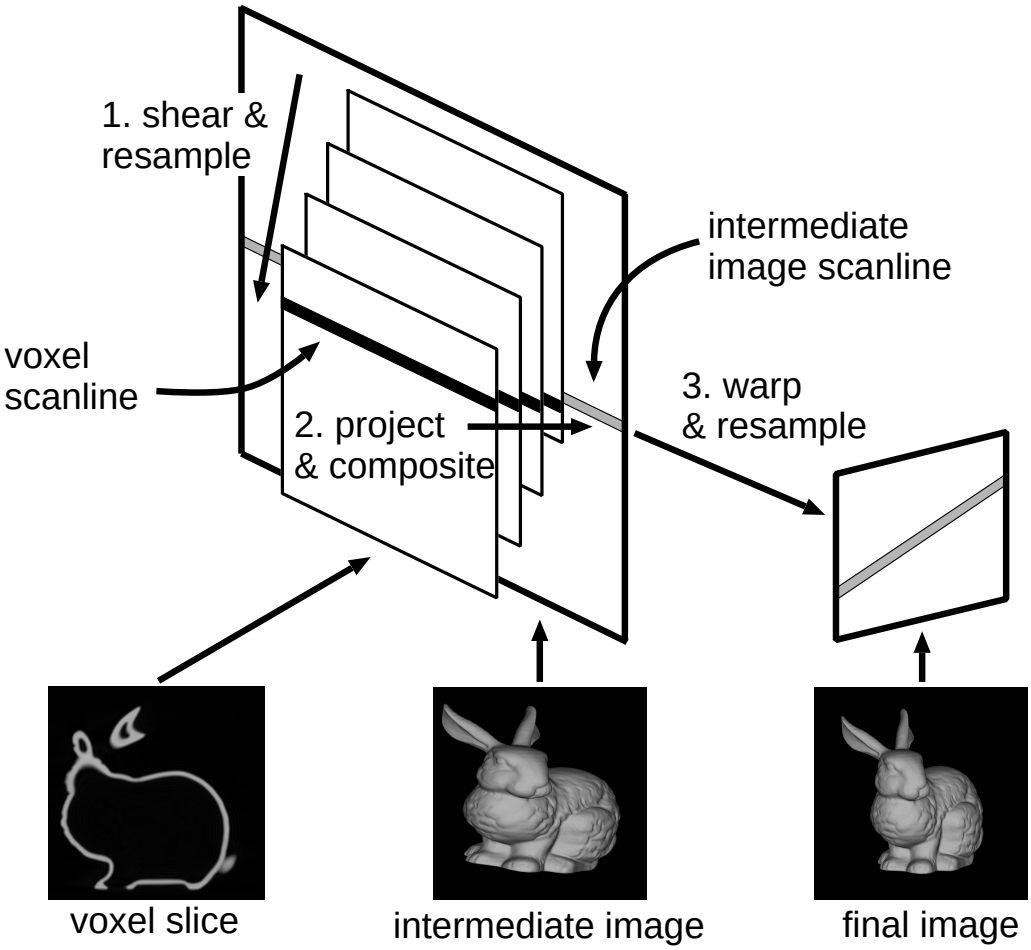


Figure 2.11: The three conceptual steps of the shear-warp algorithm: shear and resample the volume slices, project the resampled slices onto the intermediate image, and warp the intermediate image into the final image.

2.3.4 Texture Mapping

Through the availability of very powerful programmable graphics hardware the *Texture Mapping* approach has become very interesting. Early work was done by Cabral [4]. At the beginning only 2D textures intended to enhance surface graphics was supported. The concept is very similar to the shear-warp factorization (see Section 2.3.3). The volume is separated into one stack of slices for each main coordinate axis. The stack which is most perpendicular to the viewing axis is selected for rendering, this is done by mapping the slice information onto a stack of triangles which has the same geometric setup as the slices they represent. Transformation of the triangles and blending into the frame buffer is done on the graphics hardware. This method is illustrated in Figure 2.12.

Further development led to the introduction of graphics cards that feature 3D textures. The volume data is stored in the memory of the graphics hardware, if a triangle intersects the

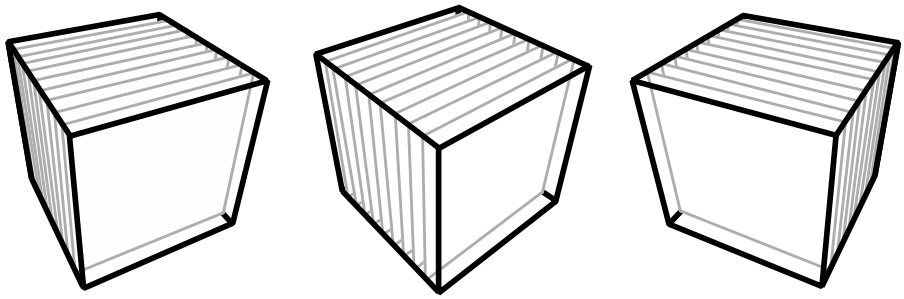


Figure 2.12: The 2D texture approach uses one stack of texture slices for each of the main coordinate axes. The one most parallel to the projection plane is displayed in a back-to-front order with alpha blending.

3D texture, the cutting plane is projected onto the triangle. To use this behavior for volume rendering, a stack of slices parallel to the projection plane intersecting the volume is drawn. The geometric transformation and the compositing is done in hardware again. The setup for different angles can be seen in Figure 2.13.

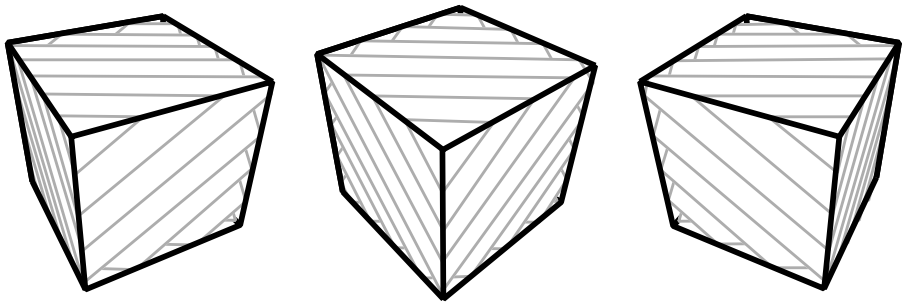


Figure 2.13: The 3D texture mapping method uses one 3D texture, which is mapped onto slices parallel to the projection plane. These slices are then displayed in a back-to-front order with alpha blending.

2.4 Fourier Transform in Volume Rendering

Frequency domain volume rendering (FDVR), often also referred to as Fourier volume rendering (FVR), is a volume rendering technique first introduced by Dunne et al. [7]. Malzbender [21], Levoy [19] and Totsuka [34] contributed in establishing this method in the following years. Recently Lee et al. [15], Westenberger and Roederik [35], Entezari et al. [8], Stark [31], Dornhofer [6] contributed improvements to this approach. The main enhancements where in adding shading and the extention to non Cartesian grids.

The FVR method is based on the *Projection Slice Theorem* of the Fourier transform, which states that a two-dimensional slice s, passing the origin of the frequency domain representation of a three-dimensional volume, inverse Fourier transformed, equals a projection of the whole

volume along the normal vector to s. If the size of the volume is N^3, then computational expense of this operation is $\mathcal{O}(N^2 logN)$ as compared to $\mathcal{O}(N^3)$ of the pure spatial domain equivalent.

Unfortunately even with the most recent improvements this method generates only "x-ray" like images, see Figure 2.14. The lack of occlusion and support of transfer functions are the major drawbacks of this method.

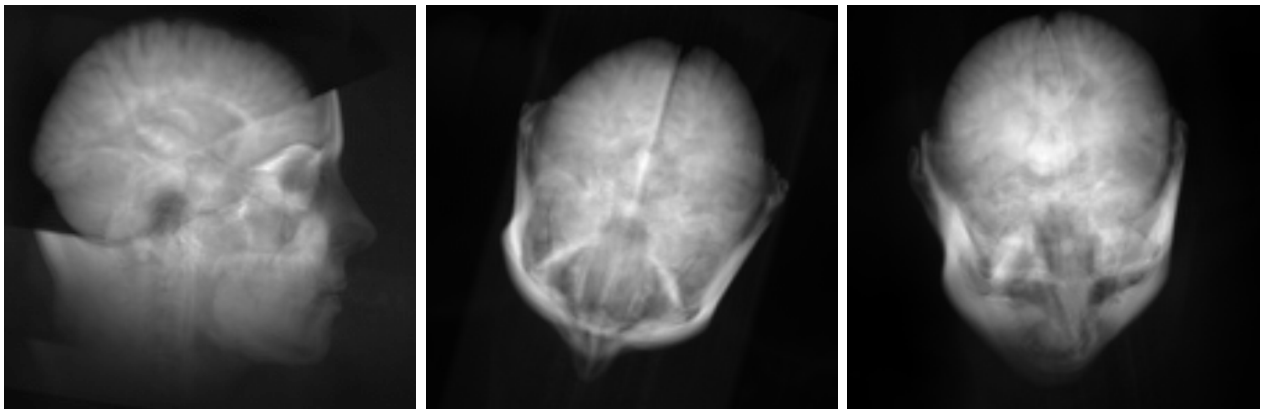


Figure 2.14: The UNC-brain dataset from different angles, using Fourier Volume Rendering.

During the work on this thesis a paper by Li et al. [20] was published that uses the same principle as our method, i.e., to perform the resampling of the volume in the frequency domain. Their approach is to decompose the transformation matrix into four shear operations. These four shear operations are performed by exploiting various frequency domain techniques and require multiple forward and backward Fourier transforms. In our approach the transformation matrix is factored according to the shear-warp factorization which requires only one shear operation to be executed in the frequency domain. Another issue is that if the volume is resampled by the application of shear operations it is necessary to add sufficient spatial domain zero padding to fully accommodate the rotated volume. The problem that arises if the spatial domain zero padding is too small, is that parts of the data volume pass over the border of the volume and through the periodicity of the dataset enter from the other side. This error is amplified by the consecutive shears. To allow arbitrary positions of the volume a symmetric spatial domain zero pad of $\frac{\sqrt{3}}{2}$ times the maximal volume resolution has to be applied. This creates an up to three times higher memory consumption as compared to our method that does not require spatial domain zero padding in that dimension. We further introduce a gradient estimation scheme that takes advantage of the derivative theorem of the Fourier transform which could probably be applied to their work.

Chapter 3

Fourier Transform

In mathematics you don't understand things. You just get used to them.

Johann von Neumann

The aim of this chapter is to provide a collection of equations and theorems of the Fourier transform used in later sections. In the interest of brevity is it not possible to give a full introduction into Fourier transforms in this work. A more comprehensive and detailed introduction to the Fourier transform can be found for example in Oppenheim and Schafers book [27].

3.1 Transform Pair

The Fourier transform links a signal with its representation in the frequency domain. There are several forms of the Fourier transform depending whether the input signal is continuous, finite or periodic. We define the following transforms accordingly:

Fourier Transform (FT):

$$X(\sigma) = \int_{-\infty}^{+\infty} x(t) e^{-2\pi i \sigma t} dt \tag{3.1}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\sigma) e^{2\pi i \sigma t} d\sigma \tag{3.2}$$

Discrete-Time Fourier Transform (DTFT):

$$\hat{X}(\sigma) = \sum_{n=-\infty}^{\infty} x[n] e^{-2\pi i \sigma n} \tag{3.3}$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \hat{X}(\sigma) e^{2\pi i \sigma n} d\sigma \tag{3.4}$$

Discrete Fourier Transform (DFT):

$$\hat{X}_M[\mu] = \sum_{n=0}^{N-1} \hat{x}_N[n] e^{-in\frac{2\pi\mu}{M}} \tag{3.5}$$

$$\hat{x}_N[n] = \frac{1}{M} \sum_{\mu=0}^{M-1} \hat{X}_M[\mu] e^{i\mu\frac{2\pi n}{N}} \tag{3.6}$$

To get $\hat{x}_N[n]$ back exactly from $\hat{X}_M[\mu]$ we need $M \geq N$, but in most applications typically $M = N$. The computational effort for this transform is $\mathcal{O}(NM)$, or $\mathcal{O}(N^2)$ when $N = M$. There are more sophisticated algorithms named Fast Fourier Transform (introduced by Cooley and Tukey [5]); usually denoted as FFT that accomplishes the transform in $\mathcal{O}(NlogN)$ time complexity. For the FFT, there are no restrictions on N, but the most well known version of the algorithm is the radix 2 transform, which assumes $N = 2^k$ [27].

3.2 Shift Theorem

To obtain the original data samples $\hat{f}_N[n]$ displaced by an offset a along the coordinate axis, the frequency domain representation $\hat{F}_N[\nu]$ is multiplied with a complex exponential function term, given by

$$\hat{f}_N[n-a] \iff e^{-ia\frac{2\pi\nu}{N}} \hat{F}_N[\nu] \tag{3.7}$$

$$e^{ia\frac{2\pi n}{N}} \hat{f}_N[n] \iff \hat{F}_N[\nu-a] \tag{3.8}$$

It is possible to displace the frequency domain representation $\hat{F}_N[\nu]$ by performing a similar multiplication in spatial domain, see Equation 3.8.

3.3 Convolution Theorem

Wolfram [23] states that "A convolution is an integral that expresses the amount of overlap of one function $g(t)$ as it is shifted over another function $f(t)$". The convolution of continuous signals is indicated by $*$, the symbol for the discrete periodic convolution is a $\circledast$. The *Convolution Theorem* states that if a convolution of two signals $f(t)$ and $g(t)$ in the spatial domain is equal to $y(t)$ (Equation 3.9), and the multiplication of their Fourier transform pairs $F(\sigma)$ and $G(\sigma)$ gives $Y(\sigma)$ (Equation 3.10), then $y(t)$ and $Y(\sigma)$ are transform pairs.

$$y(t) = f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau \tag{3.9}$$

$$Y(\sigma) = F(\sigma) G(\sigma) \tag{3.10}$$

The convolution is computed in different ways depending on the form of the involved signals. In our application we convolve only discrete periodic signals with the same period length N. In this case the periodic convolution as defined in Equation 3.11 for spatial domain and Equation 3.12 for the frequency domain is used.

$$\hat{y}_N[n] = \hat{f}_N[n] \circledast \hat{g}_N[n] = \sum_{m=0}^{N-1} \hat{f}_N[m]\hat{g}_N[n-m] \tag{3.11}$$

$$\hat{Y}_N[\nu] = N\hat{F}_N[\nu]\hat{G}_N[\nu] \tag{3.12}$$

The computational complexity of performing the convolution in spatial domain is $\mathcal{O}(N^2)$. The frequency domain approach includes a forward and inverse Fourier transform (each $\mathcal{O}(NlogN)$) plus the multiplication of the signals in the frequency domain ($\mathcal{O}(N)$). Therefore the cost of computing the convolution in frequency domain is $\mathcal{O}(NlogN)$, which is substantially faster than the convolution in spatial domain.

3.4 Packing Theorem

The *Packing Theorem* is used to change the number of samples in one period of a signal $\hat{f}_N[n]$ with period N to $\hat{f}_K[n]$ with period K. The packing operator is applied on the frequency domain representation $\hat{F}_N[\mu]$, of the signal $\hat{f}_N[n]$, to obtain $\hat{F}_K[\mu]$. In case of $K > N$, the extension of the signal period is performed by appending zero valued samples to the frequency domain representation of the signal, see Equation 3.14. If the resulting period K is smaller than N, samples representing high frequency components are removed from $\hat{F}_N[\mu]$, see Equation 3.15.

$$Pack_K\{\hat{F}_N[\mu]\} = \hat{F}_K[\mu] \tag{3.13}$$

with

$$\hat{F}_K[\mu] = \begin{cases} \frac{K}{N}\hat{F}_N[\mu] & 0 \leq \mu \bmod K \leq N-1 \\ 0 & N \leq \mu \bmod K \leq K-1 \end{cases} \tag{3.14}$$

for $\mu \in \mathbb{Z}$ and $K \geq N$.

$$\hat{F}_K[\mu] = \frac{K}{N}\hat{F}_N[\mu] \quad 0 \leq \mu \bmod K \leq N-1 \tag{3.15}$$

for $\mu \in \mathbb{Z}$ and $K \leq N$.

The factor $\frac{K}{N}$ keeps the values of $\hat{f}_K[n]$ at the scale of $\hat{f}_N[n]$. This operation is often also referred as zero padding. A drawback of this method is that arbitrary scaling of signals is not possible; the period of a given signal can only be changed in discrete steps.

3.5 Derivative Theorem

Taking the derivative of the discrete Fourier transform (Equation 3.6), gives

$$\hat{x'}_N[n] = \frac{1}{M}\sum_{\mu=0}^{M-1} \hat{X}_M[\mu] e^{i\mu\frac{2\pi n}{N}} \cdot i\mu 2\pi \frac{1}{N} \tag{3.16}$$

This leads to the *Derivative Theorem* for the DFT which states that a discrete periodic signal $\hat{x}_N[n]$ with a frequency domain representation of $\hat{X}_M[\mu]$ can be derived by multiplying with $i\mu 2\pi\frac{1}{N}$ in the frequency domain, see Equation 3.18.

$$\hat{x}_N[n] \iff \hat{X}_M[\mu] \tag{3.17}$$

$$\hat{x'}_N[n] \iff i\mu 2\pi \frac{1}{N}\hat{X}_M[\mu] \tag{3.18}$$

3.6 Windowing

If a signal is sampled below the Nyquist frequency the replicas of the fundamental period in the frequency domain overlap. Thus the perfect reconstruction of the original function is not possible. For more details see Section 2.1.2. The errors introduced when reconstructing a signal sampled below the Nyquist frequency are called aliasing. Unfortunately in practice it is often the case that signals are not band limited and therefore the Nyquist frequency is infinite. Every discrete sampling of such a not band limited signal introduces aliasing artifacts by definition. One possible appearance of these aliasing artifacts is the Gibbs phenomenon [27]. The Gibbs phenomenon is an overshooting of the reconstructed function which appears around discontinuities of the sampled function. It is also referred to as ringing. The appearance of the Gibbs phenomenon can be decreased through a multiplication of the Fourier series representation of the signal with a weighting window function.

Some commonly used windows are:

Rectangular

$$\hat{W}_N[\nu] = \begin{cases} 1 & 0 \leq \nu \leq M \\ 0 & otherwise \end{cases} \tag{3.19}$$

Bartlett(triangular)

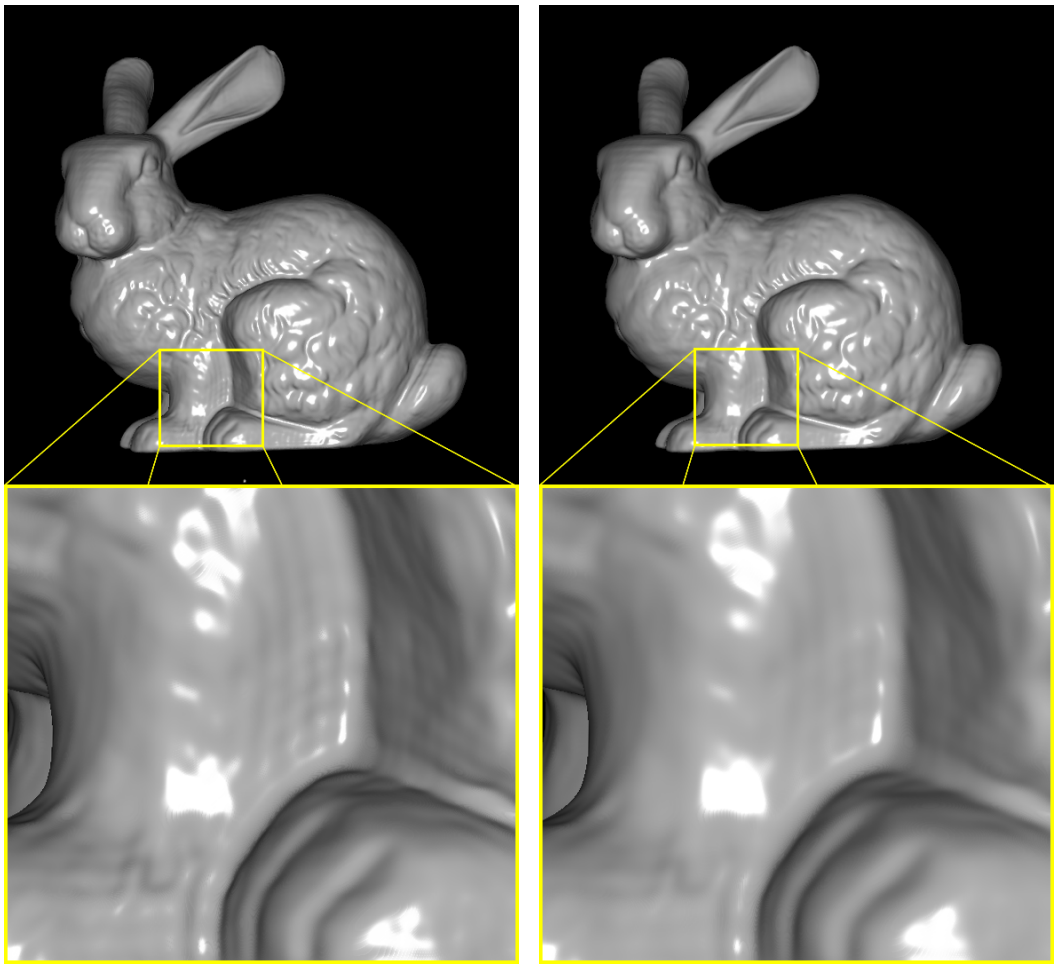
$$\hat{W}_N[\nu] = \begin{cases} \frac{2\nu}{M} & 0 \leq \nu \leq \frac{M}{2} \\ 2 - \frac{2\nu}{M} & \frac{M}{2} < \nu \leq M \\ 0 & otherwise \end{cases} \tag{3.20}$$

Hanning

$$\hat{W}_N[\nu] = \begin{cases} 0.5 - 0.5\cos(\frac{2\pi\nu}{M}) & 0 \leq \nu \leq M \\ 0 & otherwise \end{cases} \tag{3.21}$$

Hamming

$$\hat{W}_N[\nu] = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi\nu}{M}) & 0 \leq \nu \leq M \\ 0 & otherwise \end{cases} \tag{3.22}$$



(a) (b)

Figure 3.1: (a) The Stanford Bunny rendered with the new Fourier based rendering algorithm. Ringing artifacts are especially visible parallel to the hind leg. An application of the Hamming window in all three dimensions in the frequency domain representation of the dataset before starting the rendering process smooths these effects. (b) demonstrates the obtained rendering result after windowing. The images of the first row were rendered with a zoom factor of 4 and the enlarged sections with a zoom factor of 20.

Figure 3.1 displays two images which where rendered with the rendering algorithm introduced in this work. Figure 3.1(a) shows the rendering result obtained by direct application of

the proposed algorithm. Strong ringing artifacts are visible parallel to the hind leg. The image in Figure 3.1(b) was rendered after an application of the Hamming window in all three space directions in the frequency domain representation of the dataset. The ringing artifacts are very much reduced, but also some detail in the image is smoothed out. This new volume rendering algorithm is presented in detail in the following section. The Fourier transforms, theories and methods introduced in this section are put together to perform important parts of this new method.

Chapter 4

Shear-Warp Factorization in the Fourier Domain

Do not worry about your difficulties in Mathematics.
I can assure you mine are still greater.

Albert Einstein

Displaying data sampled on a three-dimensional grid is the general purpose of a volume rendering algorithm. Other requirements can vary depending on a given application. Some application might favor high frame rates over very accurate images rendered from the input volumes.

The focus of the algorithm presented in this chapter is to provide very high rendering quality by using theorems of the Fourier transform that are widely used in 1D and 2D signal processing.

4.1 Introduction

Our method is conceptually based on the shear-warp factorization introduced by Lacroute and Levoy [14]. This algorithm factorizes the viewing transformation of the volume from the object space into the image space into a permutation, a shear, and a warp transformation. The shear transformation, where most of the critical interpolation takes place, only uses translation of volume slices along the coordinate axes. In this work we propose a new method performing the shear transformation in the frequency domain. To further improve the quality of the resulting images, two modifications of the standard shear-warp approach are introduced. First, a method is proposed for calculating intermediate slices before the shear operation is applied.

This ensures that we obtain a steerable and viewing direction independent sampling distance along the viewing rays. Second, we introduce a technique to perform zooming in the standard object coordinate system as compared to zooming in the warping stage, which improves image quality significantly. Additionally a high quality gradient estimation scheme based on the derivative theorem of the Fourier transform is presented.

The advantage of performing these operations in the frequency domain is that the Fourier transform offers theorems that allow to shift, scale and differentiate signals with very high precision. This section builds the mathematical background for the shear-warp factorization and introduces the modified rendering pipeline. It describes each rendering stage and explains which theorems of the Fourier transform are applied.

4.2 Coordinate Systems

The naming conventions for the coordinate systems and axes used in this work correspond with the original mathematical groundwork done by Lacroute [13]. Figure 4.1 displays the coordinate systems used during the derivation of the shear-warp factorization. All four coordinate systems are right handed.

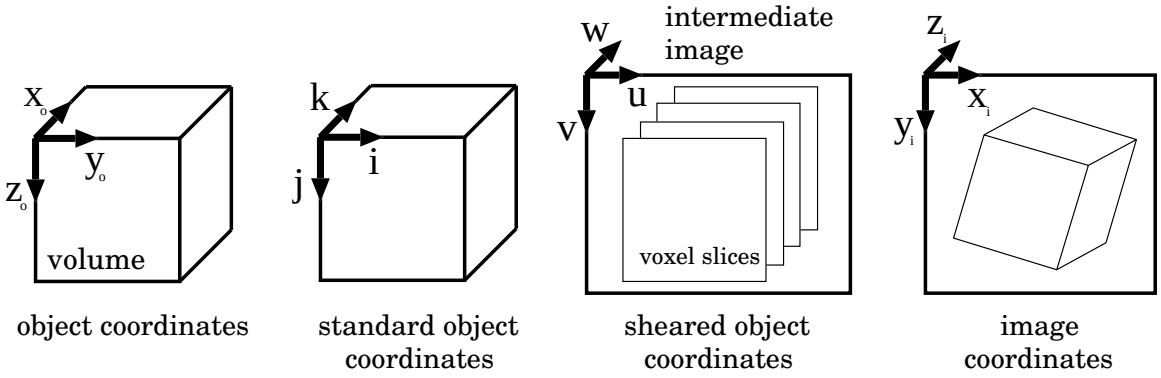


Figure 4.1: Coordinate systems used when deriving the shear-warp factorization.

The object coordinate system is the initial coordinate system of the volume. The origin of the coordinate system is located at the corner of the volume. The unit distance on each axis equals one voxel length along the corresponding axis. The three axes are labeled x_o, y_o and z_o.

The standard object coordinate system is created by permuting the axes of the object coordinate system, in such a way that the principal viewing axis becomes the third coordinate axis, which is the axis that is the closest to be parallel to the viewing direction. The axes are labeled i, j and k whereas k is the principal viewing axis.

We construct the sheared object coordinate system by shearing the standard object coordinate system with the shear coefficients obtained from the shear-warp factorization, see

Section 4.3.3. This coordinate system is also the coordinate system of the intermediate image, and its origin is located at the upper left corner. The axes are labeled u, v and w.

The image coordinate system is created by transforming the sheared object coordinate system with the warp matrix from the shear-warp factorization, see Section 4.3.5. This is the coordinate system of the final image. The origin of the image coordinate system is located at the upper-left corner of the final image. The axes names are x_i, y_i and z_i, whereas z_i is perpendicular to the image plane.

4.3 Mathematics of the Shear-Warp Factorization

In this section a summary of the equations needed to perform the shear-warp factorization is provided. For derivation and more detailed introduction see Lacroute's thesis [13].

4.3.1 The Viewing Transformation Matrix

The viewing transformation matrix M_{view} is a four-by-four matrix that transforms homogeneous points from object space to image space:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ w_i \end{bmatrix} = M_{view} \cdot \begin{bmatrix} x_o \\ y_o \\ z_o \\ w_o \end{bmatrix}$$

The standard shear-warp factorization supports affine and perspective factorization. Exact scaling of volume slices by arbitrary factors is necessary for perspective projection. Our method uses zero padding to perform zooming of volume slices. Therefore the size of slices can only be increased in discrete steps (see Section 3.4). This limitation is the reason that at the current state our rendering method only supports parallel projection. However, we can do perspective projection if we accept the introduction of small errors. A computational very expensive combination of the shifting theorem and an inverse Fourier transform for each grid row and column would allow it to achieve arbitrary scaling factors. This solution has not been investigated yet and is referred to Section 8 for future research.

4.3.2 Finding the principal viewing axis

The principal viewing axis is the axis that is closest to be parallel to the viewing direction. Let $\vec{v}_o$ be the viewing direction vector transformed to object space, and let m_{ij} be the elements of

the viewing transform matrix M_{view}. Then $\vec{v}_o$ is calculated from M_{view} in the following way:

$$\vec{v}_o = \begin{bmatrix} m_{12}m_{23} - m_{22}m_{13} \\ m_{21}m_{13} - m_{11}m_{23} \\ m_{11}m_{22} - m_{21}m_{12} \end{bmatrix}$$

Each line computes one component of the viewing vector $\vec{v}_o$. The component with the biggest absolute value indicates the principal viewing axis. After the principal viewing axis is found a permutation matrix P is selected from one of the following:

$$P_{x_o} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{y_o} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{z_o} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The subscript t of P_t (with $t \in \{x_o, y_o, z_o\}$) indicates for which principal viewing axis a permutation matrix is intended. After a P_t is selected it is referred to as P.

$$\begin{bmatrix} i \\ j \\ k \\ 0 \end{bmatrix} = P \cdot \begin{bmatrix} x_o \\ y_o \\ z_o \\ w_o \end{bmatrix}$$

The matrix P transforms object coordinates into standard object coordinates. The viewing vector $\vec{v}_o$ transformed by P is called $\vec{v}_{so}$. The actual volume data (that should be displayed during the rendering process) is transformed from the object coordinate system to the standard object coordinate system by moving each voxel from its position (x_o,y_o,z_o) in object coordinates, to the target position (i,j,k) in standard object coordinates. This rearrangement of data is a time consuming process. Therefore, in the standard shear-warp factorization three copies of the input volume are stored, each one of them rotated with one of the permutation matrices. After finding the principal viewing axis and selecting a permutation matrix, the corresponding volume is used for further processing. This alleviates the burden of computation at the expense of extra storage space.

4.3.3 The Shear Coefficients

The next step in the rendering process is to perform a shear in the I and J direction. The shear operation transforms the volume into the sheared object space where the viewing direction is perpendicular to the (u,v) plane.

Therefore let M'_{view} be a permuted viewing transformation matrix that transforms standard object coordinates to image space coordinates:

$$M'_{view} = M_{view}P^{-1}$$

The shearing coefficients s_i and s_j for the I and J axis are calculated from M'_{view}, where m'_{ij} are the elements of M'_{view}, in the following manner:

$$s_i = \frac{m'_{22}m'_{13} - m'_{12}m'_{23}}{m'_{11}m'_{22} - m'_{21}m'_{12}}$$

$$s_j = \frac{m'_{11}m'_{23} - m'_{21}m'_{13}}{m'_{11}m'_{22} - m'_{21}m'_{12}}$$

This shear operation transforms the standard object coordinate system into a "sheared" coordinate system. The resulting coordinate system is not advantageous because the origin is not located at a corner of the intermediate image. The origin of this sheared coordinate system is therefore translated to the upper-left corner of the intermediate image. There are four different cases for this translation depending on the signs of the shear coefficients s_i and s_j, see Table 4.1.

s_i	s_j	t_i	t_j
$s_i \geq 0$	$s_j \geq 0$	$t_i = 0$	$t_j = 0$
$s_i \geq 0$	$s_j < 0$	$t_i = 0$	$t_j = -s_j k_{max}$
$s_i < 0$	$s_j \geq 0$	$t_i = -s_i k_{max}$	$t_j = 0$
$s_i < 0$	$s_j < 0$	$t_i = -s_i k_{max}$	$t_j = -s_j k_{max}$

Table 4.1: The translation (t_i,t_j) gives the displacement from the origin of the standard object coordinate system to the origin of the sheared object coordinate system. The signs of s_i and s_j distinguish four different cases, in order to position the origin of the sheared object coordinate system at the upper-left corner of the intermediate image. k_{max} is the highest coordinate value of a slice in K direction in the standard object space.

The slices of the volume in standard object space are displaced by $c_k s_i + t_i$ in I direction and $c_k s_j + t_j$ in J direction, where c_k is the k coordinate of each slice in standard object space. The shear transformation followed by the translation, transforms standard object coordinates to sheared object coordinates.

4.3.4 Projection onto the Intermediate Image

The voxel slices in the sheared object space are composited along the w axis, which is the viewing direction as well, into the slice located at $w = 0$. The slices are sorted by their K coordinate in standard object space before compositing. The sign of the k component of the

viewing direction vector $\vec{v}_{so}$ ($v_{so,k}$) in the standard object coordinate system determines the stacking order. If $v_{so,k}$ is positive then the slice with the coordinate $k = 0$ is the front slice; otherwise the slice at the other end of the stack, with the coordinate $k = k_{max}$ is the front slice.

4.3.5 Warping Matrix

The matrix M_{warp2D} is calculated from M'_{view} (m'_{ij} are elements of M'_{view}), and the translators t_i and t_j:

$$M_{warp2D} = \begin{bmatrix} m'_{11} & m'_{12} & (m'_{14} - t_i m'_{11} - t_j m'_{12}) \\ m'_{21} & m'_{22} & (m'_{24} - t_i m'_{21} - t_j m'_{22}) \\ 0 & 0 & 1 \end{bmatrix}$$

The warping matrix M_{warp2D} finally converts the 2D intermediate image from the sheared object space to the image space. Linear interpolation or higher order filtering is used in this step to resample the intermediate image into the warped final image.

4.4 The Rendering Pipeline

As already mentioned the algorithm proposed in this thesis is based on the shear-warp factorization introduced by Lacroute and Levoy [14]. The shear-warp factorization was created to be one of the fastest software based rendering algorithms. It gains its performance by factoring the projection matrix, that transforms the volume from the object space into the image space, into several submatrices. The transformation described by each of these submatrices can be computed very effectively and through that an immanent increase in rendering speed is possible.

In our method we use the same submatrices as in the shear-warp factorization, the main focus however is to aim for highest possible quality. The shear-warp factorization has four rendering stages: permutation, shearing, compositing and warping. The permutation stage is a movement of voxels from one position to another according to a certain permutation matrix. This procedure creates no loss in quality of the representation of the data. In the shearing stage the volume is dismantled into slices which are resampled on a sheared grid. The quality of this resampling process depends very much on the filter used for the reconstruction of the signal. In the proposed method we present a way of how to perform the shear with frequency domain methods, which leads to an interpolation result in the theoretically best, *sinc*-filtered, quality. The next stage, i.e., the compositing, is equal to a numeric integration along the viewing rays. A major drawback of the standard shear-warp is that in this stage the numeric integration can only be calculated with one certain sampling distance. This sampling distance

is very coarse (> 1.0) and additionally viewing direction dependent. In our rendering pipeline we propose a resampling step that allows to perform the numerical integration along the rays with an arbitrary sampling distance, independent of the viewing direction. The last stage, i.e., the warping, transforms the intermediate image of the compositing stage into the final image. The warping in our method remains similar to the standard shear-warp warping. To maintain the high quality requirements a higher order spatial domain filer is used. The warping could be performed in the frequency domain by decomposing the warping matrix into three shear transformations [10]. One shear transformation would consist of a one-dimensional forward Fourier transformation of the intermediate image in the direction of the shear. Then each scan line of the image would be moved to its final position by exploiting the shifting theorem of the Fourier transform. Finally an inverse Fourier transform in the direction of the shear would complete one shear transformation. The quality loss caused by the resampling in this stage is small enough to not create visible artifacts in the final image, therefore spatial domain resampling provides sufficient accuracy.

The adapted rendering pipeline for the new frequency domain based method is presented in Figure 4.2. During this section a detailed explanation of every stage of this pipeline and its contribution to the rendering process is given. The first stage of the rendering pipeline starts from the standard object coordinate system, that means the volume is already permuted such that the K axis is the main viewing axis.

4.4.1 Multi-Dimensional Fourier Transform

The Fourier transform is a separable transformation. That means every M-dimensional Fourier transform can be composed from M one-dimensional transforms in every dimension respectively. To transform the volume $\hat{f}_{IJK}[i,j,k]$ in K direction, it is fragmented into $i \cdot j$ 1D signals $\hat{f}_{K_{ij}}[k]$ with a period of length K. The discrete Fourier transform (Equation 3.6) transforms $\hat{f}_{K_{ij}}[k]$ into $\hat{F}_{K_{ij}}[\kappa]$. The volume transformed in K direction $\hat{g}_{IJK}[i,j,\kappa]$ is obtained by reassembling the Fourier transformed signals $\hat{F}_{K_{ij}}[\kappa]$:

$$\hat{g}_{IJK}[i,j,\kappa] = \hat{F}_{K_{ij}}[\kappa] \qquad \forall i,j,\kappa$$

To transform the volume in all three space dimensions, a consecutive application of 1D Fourier transform in every dimension is performed. Through this transform the next pipeline stage, see Figure 4.2(b), is reached.

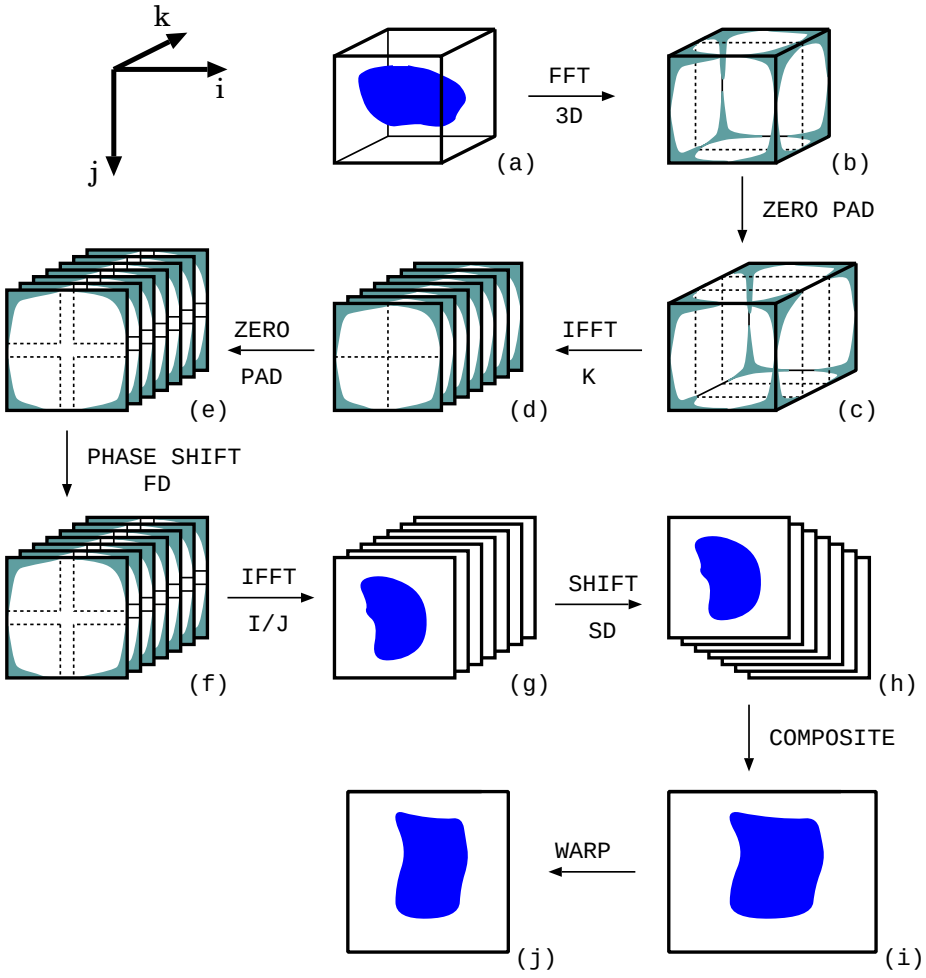


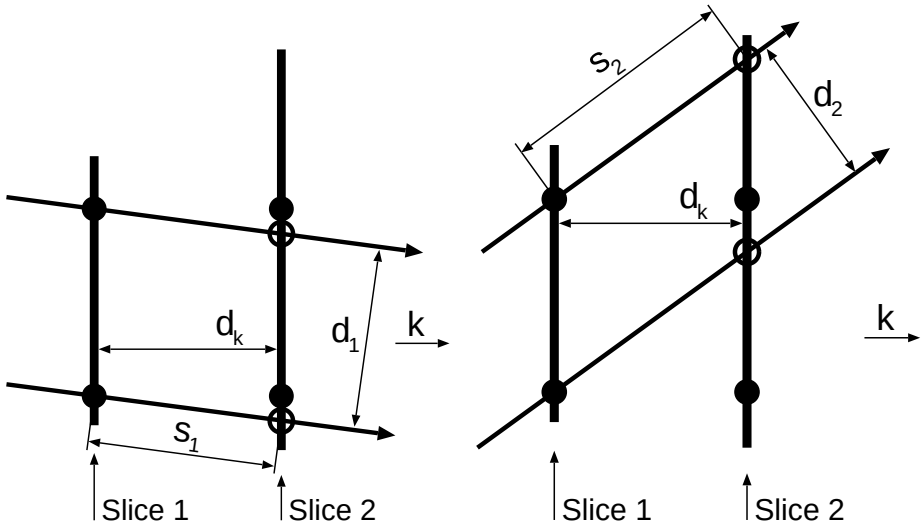
Figure 4.2: Rendering pipeline of the shear-warp factorization in the frequency domain.

4.4.2 Resampling in the Principal Viewing Axis

In the standard shear-warp factorization the number of volume slices along the principal viewing axis K is kept constant for performance reasons. Therefore the distance of the sampling points along the rays varies with the viewing direction, compare s_1 and s_2 in Figure 4.3.

This variation creates artifacts that are especially visible in animations. Resampling of the volume in K direction allows to select an arbitrary sampling distance along the rays independent of the viewing direction. Resampling creates additional volume slices and is done by exploiting the packing theorem (see Section 3.4).

To compute the amount of zero padding, necessary to obtain a certain sampling distance s', the sampling distance s along the viewing ray before resampling is calculated. The viewing vector $\vec{v}_{so}$, with its components $v_{so,i}$, $v_{so,j}$ and $v_{so,k}$ and the distance between the volume slices along the K axis d_k are necessary. We normalize $\vec{v}_{so}$ in K direction by dividing every component through $v_{so,k}$ and receive $\vec{v}'_{so}$, with its components $v'_{so,i}$, $v'_{so,j}$ and $v'_{so,k}$ ($v'_{so,k} = 1.0$). $|\vec{v}'_{so}|$ the



(a) (b)

Figure 4.3: The shear transform of the shear-warp factorization from two different viewing directions. Viewing rays are cast from the left, through the samples of the first slice, to the right. The second sample on each ray is interpolated within the second plane (black circles). Therefore only a two dimensional interpolation scheme is required, but an angle dependency of the sampling distance along the rays (s_1, s_2) and the distance between the rays (d_1, d_2) is introduced. d_k indicates the distance between two slices in K direction.

absolute length of the vector $\vec{v}'_{so}$ is the sample distance if the volume slices are 1.0 apart. A multiplication with d_k, the real distance between the volume slices, gives the final result. This setting is illustrated by Figure 4.4. The following equations summarize the computational procedure.

$$l = \sqrt{(v'_{so,i})^2 + (v'_{so,j})^2} \tag{4.1}$$

$$s = d_k\sqrt{(v'_{so,i})^2 + (v'_{so,j})^2 + (v'_{so,k})^2} \tag{4.2}$$

$$\frac{s'}{s} = \frac{K}{K'} \tag{4.3}$$

$$K' = \frac{sK}{s'} \tag{4.4}$$

l (see Figure 4.4) is the sum of the i and j component of $\vec{v}'_{so}$. s is the distance between two consecutive samples along the ray without resampling. s' is the desired sampling distance after zero padding. K is the number of volume slices in K direction which is increased to K' through zero padding. The ratio of s to s' is inverse proportional to the ratio of K to K', see Equation 4.3. This gives Equation 4.4, to compute the number of samples K' that are necessary

in K direction. The difference between K' and K is the amount of zero pad necessary for the selected sampling distance s'.

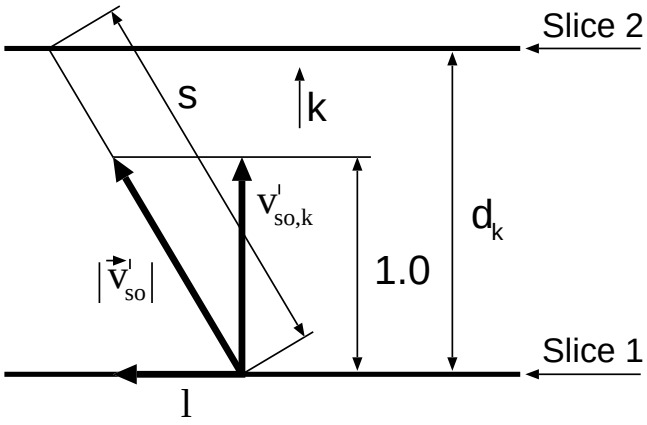


Figure 4.4: The viewing vector $\vec{v}_{so}$ intersects two slices. The distance of these two intersection points is calculated by normalizing $\vec{v}_{so}$ in K direction to get $\vec{v}'_{so}$. $|\vec{v}'_{so}|$ the absolute length of the vector $\vec{v}'_{so}$ is the distance of the intersection points if the slices are 1.0 apart. Scaling of $|\vec{v}'_{so}|$ with d_k, the real distance between the slices, gives the final result.

As zero padding can only be applied in discrete amounts, K' has to be rounded to an integer number $K'_{pad} \in \mathbb{N}$. The obtained sampling distance s'_{pad} after zero padding is related to s' through φ_k with

$$\varphi_k = \frac{K'}{K'_{pad}} \tag{4.5}$$

$$\frac{K'}{K'+0.5} < \varphi_k < \frac{K'}{K'-0.5} \tag{4.6}$$

for sizes of $K'_{pad} > 50$, the error introduced to the sampling distance is already below $\pm 1.0\%$ of K'.

The application of the zero pad brings us to the pipeline stage Figure 4.2(c). An inverse Fourier transform in K direction moves it further to Figure 4.2(d), the I and J direction of the volume remain in the frequency domain.

It is important to remember that this resampling in K direction changes the distance d_k between the slices to d_{kpad}, with

$$d'_{kpad} = d_k \frac{K}{K'_{pad}} \tag{4.7}$$

This is especially relevant in the shearing stage (see Section 4.4.4).

4.4.3 Resampling of Volume Slices in I and J Direction

In the standard shear-warp factorization zooming is performed by scaling the intermediate image. This approach leads to considerable blurring artifacts, especially for zoom factors greater than 2, as pointed out by Sweeney and Mueller [32]. In the proposed method zooming is performed earlier in the standard object space where the rescaling is applied to the volume slices. A volume slice $\hat{F}_{IJ_K}[\iota, \kappa]$ contains all the voxels of the volume that have the same k coordinate value. A desired zoom factor of z_{ij} is achieved by increasing the signal periods of each slice $\hat{F}_{IJ_K}[\iota, \kappa]$ to $F_{I'J'_K}[\iota, \kappa]$ with

$$I' = z_{ij} \cdot I \tag{4.8}$$

$$J' = z_{ij} \cdot J \tag{4.9}$$

This is accomplished by applying the packing theorem, see Section 3.4. As adding samples is only possible in discrete steps, zoom factors z_{ij} can only be achieved with limited precision. If I'_{pad} and J'_{pad} are the dimensions of the slice after the zero padding, then the error factors φ_i and φ_j are given through:

$$\varphi_i = \frac{I'}{I'_{pad}} \tag{4.10}$$

$$\varphi_j = \frac{J'}{J'_{pad}} \tag{4.11}$$

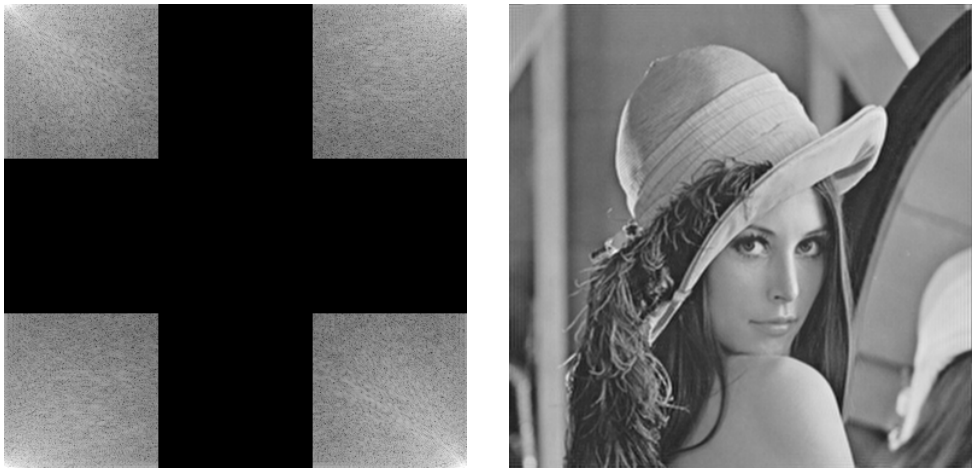
If $I'_{pad} > 50$ and $J'_{pad} > 50$ then the scaling error that is introduced is bellow $\pm 1.0\%$ of I' and J'. This error, if required, can be compensated by additional scaling in the warping step by φ_i and φ_j, see Section 4.4.6.

After the application of the zero pad the render process reaches the stage Figure 4.2(e).

Figure 4.5 demonstrates the scaling of a slice on a 2D gray scale image of Lenna [16]. First the image information is stored into $\hat{f}_{NM}[n, m]$. Then a 2D Fourier transform is applied to get $\hat{F}_{NM}[\nu, \mu]$. The packing theorem is used to increase the period length to get to $\hat{F}_{N'M'}[\nu', \mu']$. After a 2D inverse Fourier transform the scaled slice is available in $\hat{f}_{N'M'}[n', m']$. In this stage of the rendering pipeline only zero padding is performed, which corresponds to the step from Figure 4.5(b) to Figure 4.5(c). The zero pad is added in the middle of the image due to an asymmetric indexing scheme used in standard Fourier transform implementations [9]. The details about this indexing scheme are explained in Section 5.1.

An example of the difference in image quality of zooming in the standard object space, as compared to zooming at the level of the intermediate image can be seen in Section 6.2.5.

(a) (b)



(c) (d)

Figure 4.5: Process of zooming by zero padding in the frequency domain. (a) Start with a gray scale image of Lenna. The image information is stored in the real component of the spatial domain representation. (b) After 2D Fourier Transform a hermitian spectrum is obtained (displaying magnitude of the spectrum in logarithmic scale). (c) Zero pad in the high frequency area is added. (d) Resampled result image after the inverse Fourier transform.

4.4.4 Shearing

In this stage of the shear-warp factorization the volume is transformed from the standard object coordinate system to the sheared object space. This causes the viewing direction to be perpendicular to the slices of the volume, respectively the (v,u) plane. Performing the calculations explained in detail in Section 4.3.3, we acquire the shear coefficients (s_i, s_j) and the translation values (t_i, t_j). The displacements of the kth slice in I and J direction are computed as follows:

$$a_{ik} = c_k s_i + t_i \tag{4.12}$$

$$a_{jk} = c_k s_j + t_j \tag{4.13}$$

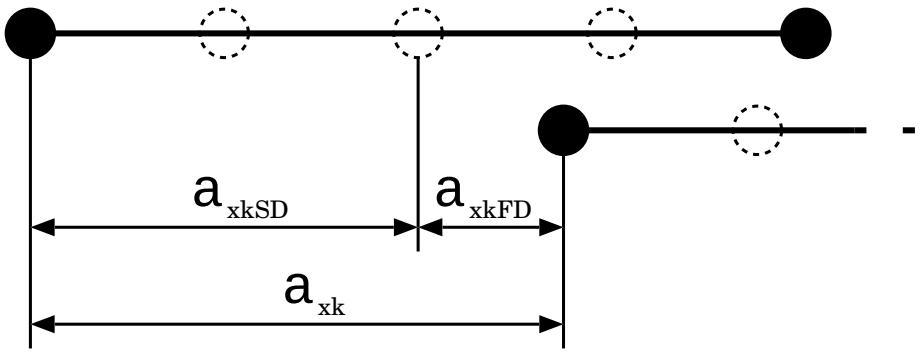


Figure 4.6: The shift of the signal by a_{xk} is split into a_{xkSD}, the multiple of whole voxel lengths and the remainder a_{xkFD}. The shift by a_{xkFD} is performed in the frequency domain, and the shift by a_{xkSD} in the spatial domain.

c_k indicates the coordinate value of the slice in K direction.

For both directions the shifts by a_{xk} ($x \in \{i, j\}$) are split into a multiple of the voxel lengths, a_{xkSD} and the remainder, which is then a fraction of a voxel step a_{xkFD}, see Figure 4.6. As the subscripts already indicate the shift for a_{xkFD} is performed in the frequency domain, and the step for a_{xkSD} in spatial domain. The shift in spatial domain is actually only a movement of the slice in full voxel steps. The actual interpolation part is performed in the frequency domain. The reason for this split of the shift is to avoid wrap around effects as presented in Figure 4.7. For the Fourier transform the volume slices are assumed to be periodic in i and j dimension. If the whole shift a_{xk} would be performed in the frequency domain more than one voxel of the neighboring period gets into the visible window area, which could lead to artifacts in the final image. To further limit the appearance of voxels from one side of the slice at the other end a spatial domain border around the volume of one voxel thickness can be added. This separates the periodic replicas in spatial domain.

When we apply the shifting theorem to perform the a_{xkFD} shift, the i and j dimensions are still in the frequency domain. After that we proceed to the rendering stage indicated in Figure 4.2(f). An inverse Fourier transform in I and J direction brings us to Figure 4.2(g). The a_{xkSD} step of the displacement is applied in spatial domain, completing the transformation of the volume to the sheared object space, see Figure 4.2(h).

4.4.5 Compositing

During this step the resampled slices are composited, along the w axis, into a 2D intermediate image. Transfer functions can be applied to the volume, as all kinds of algorithms to influence the appearance of the final image (i.e., shading, non-photorealistic effects, ...). In Section 4.4.7 the computation of high-quality gradients is explained, to obtain surface normals for lighting calculations.

Figure 4.7: The image of Lenna is the fundamental period of a 2D discrete periodic signal $\hat{f}_{NM}[n, m]$. The time shifting theorem of the Fourier transform applied to the frequency spectrum $\hat{F}_{NM}[\nu, \mu]$ moves $\hat{f}_{NM}[n, m]$ over the periodic domain. The black frame indicates a 'viewing window' or the data $\hat{f}_{NM}[n, m]$ holds after the shift. If one part of Lenna leaves the frame on one side, the next period of the signal moves in from the other end. This wrapping effect is especially visible when the displacement is bigger than one pixel.

The compositing of the slices along the w axis is one approach of calculating the integral along each viewing ray. During this process the density information of each slice $\hat{f}_{UV}[u, v]$ is transformed into an image $i_{UV}[u, v]$. Each pixel in $i_{UV}[u, v]$ has a color c and a transparency coefficient α. These images are then composited into the slice located at $w = 0$, by using the "over" operator [28]. The "over" operator states that if a pixel a with color c_a and transparency α_a is composited over a pixel b with color c_b and transparency α_b, the resulting pixel values c_c and α_c are given by:

$$c_c = c_a\alpha_a + c_b(1 - \alpha_a) \tag{4.14}$$

$$\alpha_c = \alpha_a + (1 - \alpha_a)\alpha_b \tag{4.15}$$

The order in which the slices are composited is determined by their w coordinate value. The

sign of the k component of the viewing vector $\vec{v}_{so}$ ($v_{so,k}$) in standard object space determines from which side of the stack the processing starts. If $v_{so,k}$ is positive then the slice with the coordinate $k = 0$ is the front slice; otherwise the slice at the other end of the stack, with the coordinate $k = k_{max}$ is the front slice. Compositing all slices using the "over" operator results into the non-warped intermediate image (see Figure 4.2(i)).

4.4.6 Warping

The 2D warping transformation applied to the intermediate image leads to the result image. The derivation of the necessary transformation matrix M_{warp2D} can be found in Section 4.3.5. This 3×3 matrix transforms data points from the sheared object space into the final image space. This transformation compensates the viewing-direction dependent scaling of the distances between the viewing rays (compare d_1 and d_2 in Figure 4.3), and performs the rotation component around the K axis.

If a compensation to the error in scaling in standard object space (Section 4.4.3) is required, a scaling matrix can be added to the warping matrix, with $s_u = \varphi_i$ and $s_v = \varphi_j$:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = M_{warp2D} \begin{bmatrix} s_u & 0 & 0 \\ 0 & s_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.16}$$

The combination of M_{warp2D} with the compensating scaling matrix is called $M_{warp2Dscale}$. $M'_{warp2Dscale}$ is the inverse matrix of this $M_{warp2Dscale}$ matrix. If M is the maximal volume extension (maximum of I, J and K in standard object space), and z_{ij} (see Section 4.4.3) is the scale factor applied to the volume, then an image buffer $f_{NN}[y_i, z_i]$ with $N = \sqrt{3} \cdot z_{ij} \cdot M$ holds the result image. This calculation is based on the assumption that each side of the volume is M long and the main diagonal of this cube is visible in its full length. Every pixel of $f_{NN}[y_i, z_i]$ in the image coordinate space is transformed to the sheared object space with $M'_{warp2Dscale}$. The image values at the obtained coordinates are interpolated from the pixel values of the intermediate image. In order to maintain high quality in this step as well, we use a higher order spatial domain filter for the resampling which is comparable to a Catmull-Rom spline [26] (D0 C3 4EF).

4.4.7 Gradients

Since the gradient is the partial derivative of the original function $\hat{f}_{IJK}[i, j, k]$ and ideal interpolation with the *sinc* will reconstruct that function, the gradient can be reconstructed exactly by using the derivate of the *sinc* as a reconstruction kernel [2]. This function is denoted as the *cosc* function [33]. The Fourier transform of the *cosc* function is a ramp in the frequency

domain. Applying the derivative theorem of the DFT in the three space dimensions gives the following equations:

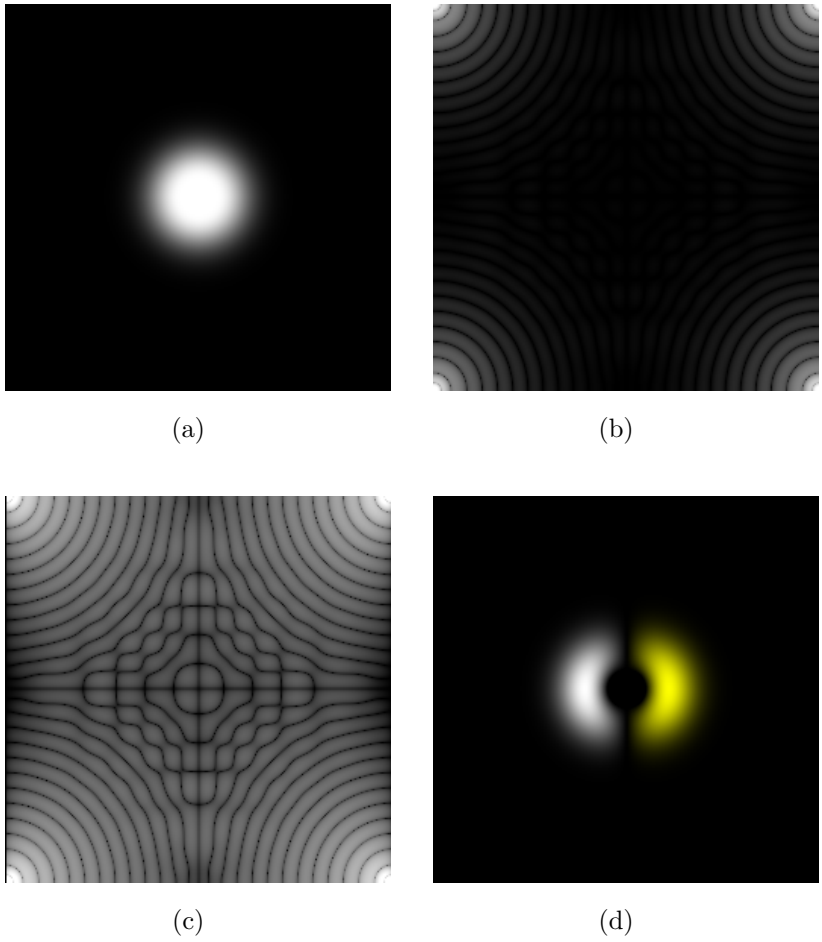
$$\frac{d}{dl}\hat{f}_{LMN}[l,m,n] \iff i\lambda 2\pi\frac{1}{L}\hat{F}_{LMN}[\lambda,\mu,\nu] \tag{4.17}$$

$$\frac{d}{dm}\hat{f}_{LMN}[l,m,n] \iff i\mu 2\pi\frac{1}{M}\hat{F}_{LMN}[\lambda,\mu,\nu] \tag{4.18}$$

$$\frac{d}{dn}\hat{f}_{LMN}[l,m,n] \iff i\nu 2\pi\frac{1}{N}\hat{F}_{LMN}[\lambda,\mu,\nu] \tag{4.19}$$

Three copies of the Fourier transformed original dataset are created, one for each dimension of the gradient. One of the three forms of the derivative theorem, see Equations 4.17, 4.18 and 4.19, is applied to each volume, in order to differentiate that volume in the direction of the gradient dimension it represents. Afterwards these gradient volumes are processed through the same rendering pipeline, as the density volume. At the compositing stage they are combined to a volume of gradient vectors. These gradient vectors are used with the processed original data to compute the intermediate image, see Section 4.4.5.

Figure 4.8 demonstrates how the derivative theorem is applied to a 2D slice. Figure 4.8(a) shows a 2D Gauß function. Figure 4.8(b) is the frequency domain representation obtained through a discrete Fourier transformation in x and y direction. The derivative theorem of the Fourier transform is applied in x direction to create Figure 4.8(c). After an inverse Fourier transform a x differentiated version of the 2D Gauß function is presented in Figure 4.8(d), where yellow indicates the negative sign.



(a) (b)

(c) (d)

Figure 4.8: Derivative reconstruction in the frequency domain. (a) Displays the image of a 2D Gauß function. (b) shows the function after a Fourier transform in 2 dimensions (displaying magnitude in logarithmic scale). (c) the application of the derivative theorem in X direction and (d) an inverse Fourier transform gives the derivative of the Gauß function in X direction (yellow indicates negative sign).

Chapter 5

Implementation

Machines take me by surprise with great frequency.

Alan Turing

The general idea of the algorithm introduced in this work is based on the standard shear-warp factorization. Therefore incorporating the new approach in an existing implementation does not create too many problems. But there are some issues concerning the Fourier transform, that are not straightforward. This section focuses on these problems that can arise when dealing with data in the frequency domain.

5.1 The Origin Problem

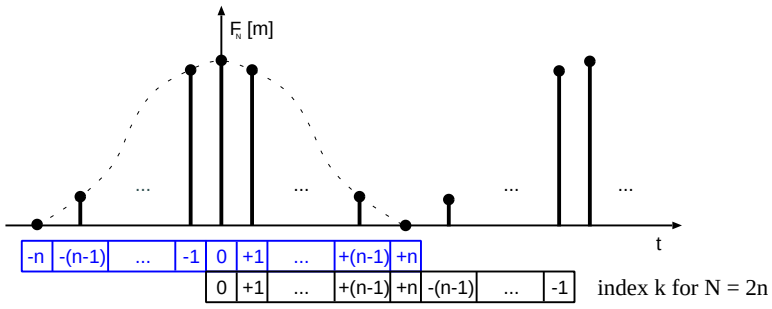
The definition of the discrete Fourier transform (DFT) creates an origin problem in the frequency and in the spatial domain. The fundamental period of the spectrum is located symmetrical around the origin. But most FFT libraries, like the FFTW [9] implementation, use a version of the DFT, see Equation 5.1 and Equation 5.2,

$$\hat{F}_N[\mu] = \sum_{n=0}^{N-1} \hat{f}_N[n] e^{-in\frac{2\pi\mu}{N}} \tag{5.1}$$

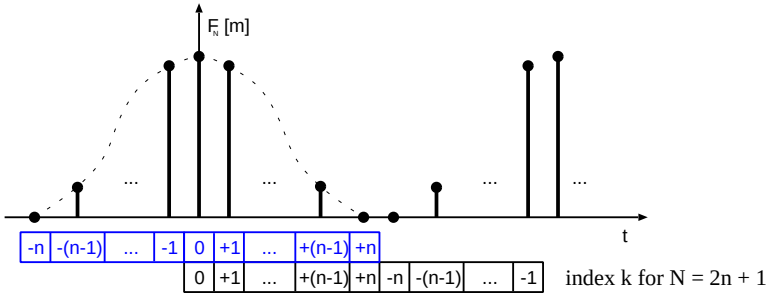
$$\hat{f}_N[n] = \frac{1}{N} \sum_{\mu=0}^{N-1} \hat{F}_N[\mu] e^{i\mu\frac{2\pi n}{N}} \tag{5.2}$$

where the data is managed asymmetrically.

The origin of the spectrum, the constant component of the signal, is located at the index 0 of the sample array. Then with increasing index the higher frequencies are captured. Up to



(a)



(b)

Figure 5.1: Asymmetric indexing scheme for an even (a) and an odd (b) number of samples.

half of the array where through the periodicity of the spectrum the negative edge of the first replica starts. The values for k are used when applying the shifting theorem and the derivative theorem. This situation is illustrated by Figure 5.1(a) for an even and by Figure 5.1(b) for an odd number of samples. In both figures the dotted line indicates the fundamental period, of the periodic spectrum, which is symmetric around the origin. The additional samples on the right represent the replicas of the fundamental period in the frequency domain, see Figure 2.6. The blue scheme displays the symmetric indexing of the fundamental period. The second scheme starts with the constant value at the first index position, and is used by most implementations. The reason for distinguishing an even and an odd case is that in the even case the two replicas share one sample in the middle of the array. This overlapping index of the replicas creates problems when applying Fourier transform theorems, like the shifting theorem, the derivative theorem or the packing theorem in the frequency domain. These problems are addressed in detail in Section 5.2.

5.2 Maintaining the Hermitian Property

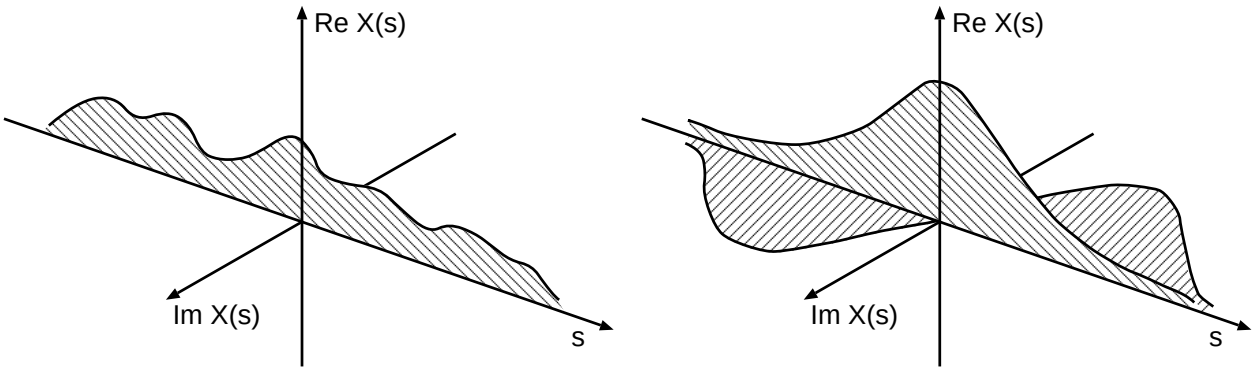
The input signal in the Fourier transform in general is complex. In this application the volume data in spatial domain is stored only in the real component and the imaginary component is set to zero. This, so called real function, is a Fourier transform pair to a Hermitian function. Therefore a Fourier transform of this spatial domain representation of the volume leads to a frequency domain representation which is Hermitian. The following equations clarify the meaning of the Hermitian property.

$$\hat{X}_N[\kappa] = \hat{X}_N[-\kappa]* \tag{5.3}$$

$$Re\{\hat{X}_N[\kappa]\} = Re\{\hat{X}_N[-\kappa]\} \tag{5.4}$$

$$Im\{\hat{X}_N[\kappa]\} = -Im\{\hat{X}_N[-\kappa]\} \tag{5.5}$$

Corresponding sample values, with index κ and $-\kappa$, are conjugate complex to each other, the real components are equal (Equation 5.4), and the imaginary components have inverse sign (Equation 5.5). Another way to look at the Hermitian property is to explore the real and

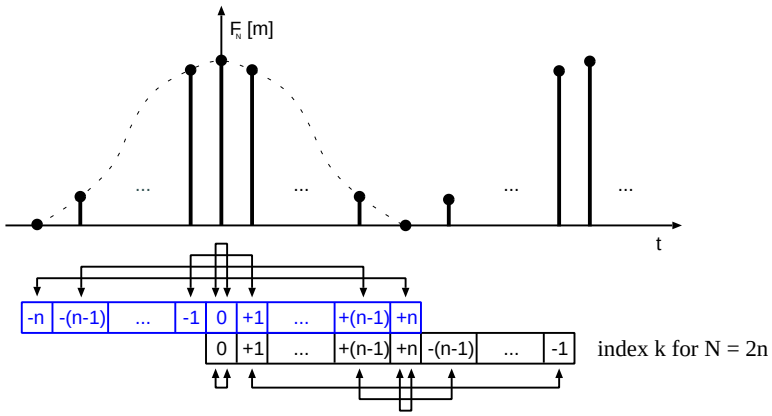


(a) (b)

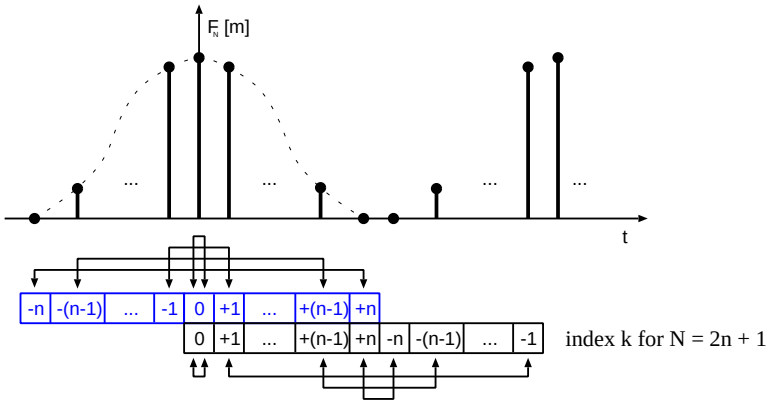
Figure 5.2: (a) A real function in spatial domain is transformed to a (b) Hermitian function in the frequency domain. If the real and the imaginary component of the Hermitian function are explored independently, then the real component forms an even function and the imaginary component an odd function.

imaginary component independently (see Figure 5.2). The complex function in Figure 5.2(b) consists of an even function in the real component, and an odd function in the imaginary component.

How the Hermitian relation applies to the indexing scheme in the discrete case is presented in Figure 5.3, there are two different patterns for even and odd number of samples.



(a)



(b)

Figure 5.3: Hermitian relation of functions with an even (a) and an odd (b) number of samples. The arrows indicate which samples have to be conjugate complex to each other to maintain the Hermitian property.

The reason for going into details about Hermitian functions is that they are dual to the real function by the Fourier transform. That means all operations applied in the frequency domain have to preserve the Hermitian property in order to obtain a real function after the inverse Fourier transform. The indices which need special attention are listed in Table 5.1. The first issue treats sample points that are conjugate complex to itself, and therefore the imaginary component has to be zero. That is taken care of by the Fourier transform itself, the result after the transformation to the frequency domain contains zeros at the correct positions. Some available packets like FFTW [9] have special real Fourier transforms that exploit the symmetry in the data to save roughly a factor of two in both time and storage. The interface for this special transform does not allocate memory for these zero indices, to save storage space.

The second issue has an effect when calculating the derivative, Section 4.4.7. If the length

N of the signal $\hat{X}_N[\mu]$ is even with $N = 2\beta$ then $\hat{X}_N[\beta]$ has to be conjugate complex to itself to maintain Hermitianity (see Figure 5.3(a)). The multiplication with $i\mu 2\pi\frac{1}{N}$ essentially switches the real and imaginary component. As the result still has to be Hermitian, $i\mu 2\pi\frac{1}{N}\hat{X}_N[\beta]$ again has to be conjugate complex to itself. This forces $Re\{\hat{X}_N[\beta]\}$ and $Im\{\hat{X}_N[\beta]\}$ to be zero. Another way of getting around this problem is to add one zero valued sample at the end of the signal in spatial domain, in order to create an odd number of samples (spatial domain zero padding).

	index	problem	counter measure
function	$\kappa = 0$ $\kappa = n$ if N is even	$\hat{X}_N[\kappa] = \hat{X}_N[\kappa]*$	$Im\{\hat{X}_N[\kappa]\} = 0$
derivative	$\kappa = n$ if N is even	$i\hat{X}_N[\kappa] = (i\hat{X}_N[\kappa])*$	$Re\{\hat{X}_N[\kappa]\} = 0$ $Im\{\hat{X}_N[\kappa]\} = 0$

Table 5.1: Problematic indices for the Hermitian property

5.3 Shift Effect

The shift effect is not only noticeable in resampling by zero padding. It is based on the fact that the spatial domain is assumed to be periodic. Therefore the first and the last sample of a signal are neighboring. If we interpolate between the samples to zoom the signal, new sample points are also introduced between these two border samples. As the sample at the origin remains on its position, and the new samples are appended, it creates the effect of shifting the function sideways. Figure 5.4 shows a schematic of how the shift effect is generated. It becomes more visible with increasing zoom factor. In Figure 5.5 an 3×3 pixel example image is zoomed to 258×258 pixel. The high zoom factor of 86 makes the shifting effect visible.

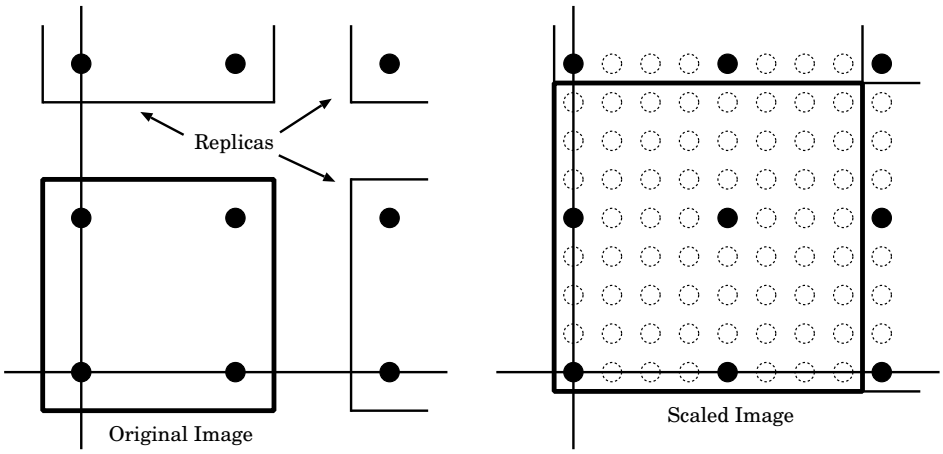
The presented rendering algorithm supports at the current state only parallel projection. Therefore every volume slice is scaled with the same scaling factor. As a counter measure to the shifting effect d pixel rows and collumns are removed at the top and the right border of the slice. With

$$d = z_{ij} - 1.0 \tag{5.6}$$

where z_{ij} is the zoom factor applied to a certain slice. This procedure at least removes eventual artifacts created at the border region.

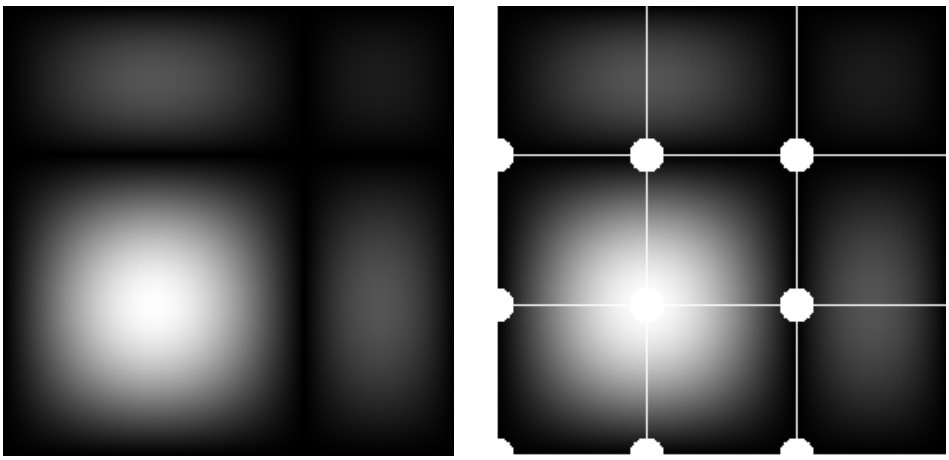
5.4 Memory Optimization

The method introduced in this work has a high demand on available memory and needs to access it in a non-linear fashion. This creates two starting points for possible optimizations,



(a) (b)

Figure 5.4: The spatial domain is considered as periodic in both dimensions. (a) shows a a 2 by 2 samples image. The black box indicates the fundamental period of the signal. New samples are inserted between the original sampling points through a scaling operation (b). As the sample at the origin remains at its position, an apparent effect of shifting of the image to the lower left corner is created.



(a) (b)

Figure 5.5: A 3×3 pixel image with a white pixel in the center and black pixel at the border is zoomed to a 258×258 pixel image by zero padding in the frequency domain. The resulting image (a) shows the shift effect to the lower left corner. In (b) the position of the pixel of the original image are highlighted.

the use and the overall demand of storage space.

5.4.1 Access Localization

The interfaces of the library used to calculate the Fourier transform, FFTW [9], expects the data to be passed in row-major format.

$$i(n_1, n_2, n_3) = n_3 + N_3(n_2 + N_2 n_1) \tag{5.7}$$

That means the linear memory index for three dimensions N_1, N_2 and N_3 with values n_1, n_2 and n_3 is calculated with Equation 5.7. If the input data volume is oriented such that (i, j, k) corresponds to (n_1, n_2, n_3), data points that belong to one slice are stored in one block of memory. In order to get the volume memory aligned for each main viewing direction, it has to be permutated in memory. The performance increases because the main computations are performed on a per slice basis.

5.4.2 Memory Consumption

The shear-warp factorization has high memory consumption because of the three volume stacks that are stored, one for each main coordinate axis. The demand in this application is even higher because every sample is represented through two floating point values.

The rendering times with this method are usually in the range of minutes, therefore it introduces only negligible overhead to store just one volume and permutate it for each frame according to the view vector. This leads to just one volume for the density information.

For the gradients usually three volumes are necessary, one for each dimension. If the derivative in I and J direction are calculated on the fly from the density slices, just before compositing, the overall number of volumes needed decreases to two copies, one to store the density information, and one for the derivative in K direction.

5.5 Shifting, Deriving, Windowing

The shifting theorem, the derivative theorem and windowing to smooth the data are defined as separable one-dimensional operations. That means to perform one of these operations to the volume $\hat{X}_{IJK}[\lambda, \mu, \nu]$ in J direction a linear array $\hat{F}_J[\mu]$ is computed, prepared with factors according to Section 3.2, Section 3.5 or Section 3.6. It is then applied to the volume with Equation 5.8.

$$\hat{X'}_{IJK}[\lambda, \mu, \nu] = \hat{X}_{IJK}[\lambda, \mu, \nu] \cdot \hat{F}_J[\mu] \quad \forall \lambda, \mu, \nu \tag{5.8}$$

To increase efficiency, especially for combinations of operations that have to be performed on several slices, these linear transformation arrays can be combined to two-dimensional transformation arrays as indicated in Table 5.2.

	$\hat{F}_I[0]$	$\hat{F}_I[1]$	...	$\hat{F}_I[I-1]$
$\hat{F}_J[0]$	$\hat{F}_{IJ}[0,0]$	$\hat{F}_{IJ}[1,0]$	...	$\hat{F}_{IJ}[I-1,0]$
$\hat{F}_J[1]$	$\hat{F}_{IJ}[0,1]$	$\hat{F}_{IJ}[1,1]$	...	$\hat{F}_{IJ}[I-1,1]$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\hat{F}_J[J-1]$	$\hat{F}_{IJ}[0,J-1]$	$\hat{F}_{IJ}[1,J-1]$	...	$\hat{F}_{IJ}[I-1,J-1]$

Table 5.2: The combination of two one-dimensional transformation arrays, $F_I[\lambda]$ in I direction and $F_J[\mu]$ in K direction, into a two-dimensional transformation array $F_{IJ}[\lambda,\mu]$ by multiplication of the corresponding entries.

Equation 5.9 displays how to combine the entries of two one-dimensional tranformation arrays into one two-dimensional transformation array.

$$\hat{F}_{IJ}[\lambda,\mu] = \hat{F}_I[\lambda] \cdot \hat{F}_J[\mu] \quad \forall\lambda,\mu \tag{5.9}$$

This combination and reuse of transform arrays is exploited in the current implementation especially for the shift operation. The expensive calculation of the exponential coefficient is performed only once, and the resulting transformation array is used to displace the density slice, and all three derivative slices.

5.6 Rendering Speed

Sometimes, especially for data exploration, fast rendering times are favored over high accuracy of the reconstruction. Even though the design of this algorithm, was not intended for speed, there are several methods to speed-up the rendering process. Most of the ideas are often used in ray casting and can be applied to this method.

5.6.1 Down Scaling in I, J Direction

The dimension of the volume is reduced in I and J direction by deletion of high frequency components in the frequency domain, eventual zooming is done in the warping step. In ray casting this is equal to a reduced number of rays cast through the scene.

5.6.2 Less, Constant Slices in K Direction

If a fine sampling distance along the rays is not necessary, then the resampling step in K direction (see Figure 4.2(c)) can be omitted. Therefore no new slices are created and the computation is less expensive. It is even possible to create a coarser sampling distance by removing high frequency components in the resampling stage which equals to a reduction of the resolution of the volume in K direction.

5.6.3 Multithreading

After the inverse Fourier transformation in K direction, Figure 4.2(d), the volume is dismantled into slices that are further processed independently. This independence is a good condition for parallel processing. The data of each slice is assigned to a single thread that can be executed in parallel. At the end, synchronization has to be done in a way that the threads project their results into the intermediate image according to the K coordinate of the slice they compute. Measurements of the speed gain are available in Section 6.2.3.

Chapter 6

Results

I may not have gone where I intended to go, but I think I have ended up where I intended to be.

Douglas Adams

In this section results and experiences with the application of the introduced technique in practice are presented. The reconstruction quality is compared to standard spatial domain filters. The filter kernels used in our implementation have been developed by Möller et al. [26]. Table 6.1 shows some of the used filters and their common names in literature.

D0 C0 1EF	Linear Interpolation
D0 C1 3EF	Catmull-Rom Spline
DN C2 2EF	B Spline

Table 6.1: Filter kernel used for comparison.

The naming conventions are the following, DN identifies an approximation filter, D0 is an interpolation filter and D1 gives the first derivative. The C value indicates how many times the reconstructed function can be derived and still remain continuous. A N-EF filter will reconstruct a polynomial function of $(N-1)$th or lower degree without errors.

6.1 Test datasets

Several 3D volume datasets are used to compare the reconstruction quality of the frequency domain method to standard spatial domain filtering. Two different kinds of datasets are used, synthetic datasets and CT-scans. The synthetic datasets are based on a continuous function that is sampled on a regular grid. The continuous function can be rendered by evaluating

the function at the sample points along the viewing rays. In this way creating a reference image to the images rendered from the sampled data. CT-scans are samples on a regular 3D grid, representing the sampled density function of a certain object. The scanner data is often provided in this format for volume rendering. Therefore comparison of the resulting images of the frequency domain based method to standard spatial domain rendering can give interesting insights.

6.1.1 Synthetic Test Datasets

To test the quality of the new interpolation method, the test function introduced by Marschner and Lobb [22] is used. This function, ρ_{ml} in Equation 6.1, is defined in three-dimensional space and sampled at 20 samples per unit distance in each dimension. The range for x, y, z is $-1 < x, y, z < 1$, with the constant values $f_M = 6$ and $\alpha = 0.25$. This continuous volume is discretized with 41 by 41 by 41 samples, which captures 99.8% of the signals energy.

$$\rho_{ml}(x, y, z) = \frac{1 - \sin(\frac{1}{2}\pi z) + \alpha(1 + \rho_r(\sqrt{x^2 + y^2}))}{2(1 + \alpha)} \tag{6.1}$$

with

$$\rho_r(r) = \cos(2\pi f_M \cos(\frac{1}{2}\pi r)) \tag{6.2}$$

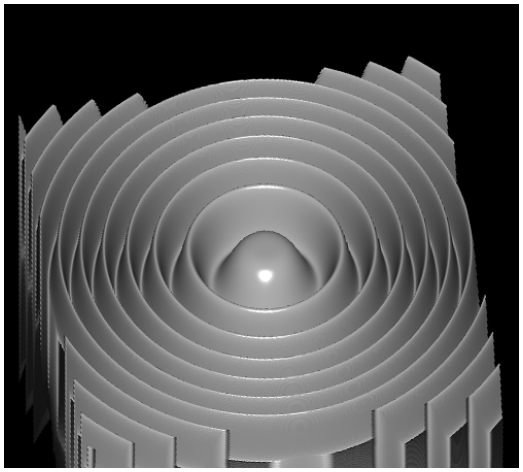
Figure 6.1 displays an image of the ρ_{ml} function, rendered with an iso-value of 0.5, a 10.0 times zoom and a sampling distance along the viewing rays of 0.05. The function ρ_{ml} was evaluated for every sample point during the rendering of this image. A standard test to judge the quality of a reconstruction filter is to render the sampled ρ_{ml} function and compare it to this reference image.

The ρ_{ml} dataset is sampled almost with Nyquist frequency. The conduct of the function in Z direction, with x and y set to the constant values c_1 and c_2, is a sine wave with low frequency (see Equation 6.3), only shifted by a constant value c_3 (depending on the values of x and y).

$$\rho(c_1, c_2, z) = \frac{1 - \sin(\frac{1}{2}\pi z) + \alpha(1 + \cos(2\pi f_M cos(\frac{1}{2}\pi\sqrt{c_1^2 + c_2^2}))}{2(1 + \alpha)} \tag{6.3}$$

$$= -\frac{\sin(\frac{1}{2}\pi z)}{2(1 + \alpha)} + c_3 \tag{6.4}$$

The assumption when using the discrete Fourier transform (DFT) (Section 3.1) is that the signal is periodic and discrete in spatial and frequency domain. Through the sampling of ρ_{ml} we get one period of this signal, which is half the period of a *sine* wave. In the zone between two of these periods (see Figure 6.2(a)), a jump in the signal is present. This discontinuity leads to



Description:
Synthetic test function introduced by Marschner and Lobb [22].
With a sampling of $41 \times 41 \times 41$
99.8% of the signals energy is captured.
Dimension Original:
$41 \times 41 \times 41$ double precision
Dimension Mirrored Version:
$41 \times 41 \times 80$ double precision

Figure 6.1: Characteristics of the synthetic datasets.

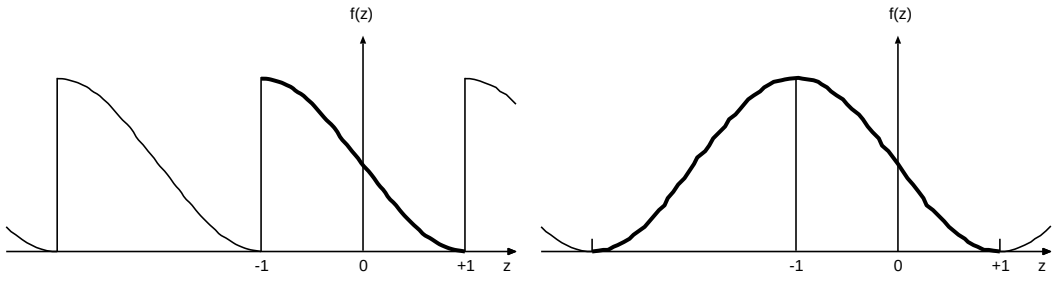
high frequency components and a not band-limited behavior. In order to avoid this problem, and to maintain compatibility with the original signal ρ_{ml}, the sample points are mirrored along the $z = -1$ plane (see Figure 6.2(b)). The samples at $z = -1$ are not copied, because they are positioned at the plane of reflection. Further the samples at $z = +1$ are not copied as well to create smooth transitions in the periodic sequence of sine waves. This creates a extended version of the test dataset with the dimensions of $41 \times 41 \times 80$ samples. This extended test dataset is used for the render benchmarks and is referred to as ρ_{mlext}.

6.1.2 Real-World Test Datasets

Three real-world dataset are used to demonstrate the reconstruction quality of the frequency domain based rendering method compared to standard spatial domain filtering. The first two datasets introduced in Figure 6.3 and Figure 6.4 were originally created by Marc Levoy and are provided for research purposes by "The Stanford volume data archive" [30].

The dataset introduced in Figure 6.5 is used in a visualization lab at the Vienna University of Technology [29].

The datasets from Figure 6.3 and Figure 6.4 where resampled to a size of $128 \times 128 \times N$ to get volumes with more manageable memory consumption and to amplify the interpolation artifacts



(a) (b)

Figure 6.2: Density characteristic of the test function in Z direction, (a) in the original version ρ_{ml} and (b) after mirroring the signal ρ_{mlext}.

introduced by the different reconstruction filters. The downsampling was done by removing of high frequency components in the frequency domain representation of the datasets. The Skewed Head dataset (Figure 6.5) was only used for one demonstration image and was not resampled.

6.2 Quality Comparison to Spatial Domain Filters

This section presents several experiments to show the advantages and disadvantages of the frequency domain based techniques compared to spatial domain filtering. The following subsections address issues that arise in volume rendering.

6.2.1 Super Sampling

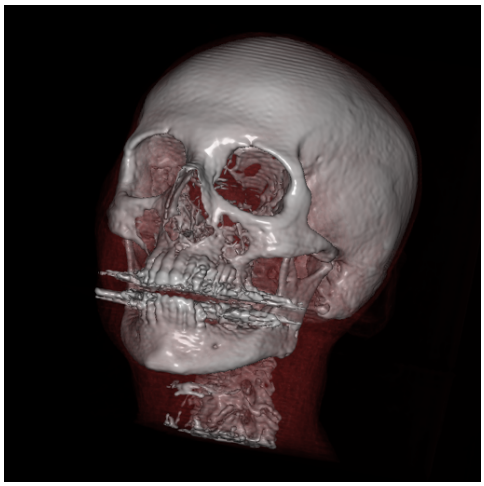
The purpose of this experiment is to compare the quality of zooming by zero padding in the frequency domain to interpolation with spatial domain filters.

As the newly introduced rendering algorithm is based on the shear-warp factorization, most of the rendering steps that are quality critical are performed on volume slices. Therefore this scaling experiment is performed on a volume slice of the ρ_{mlext} function.

Initially a 41×41 slice of the ρ_{mlext} function (see Section 6.1), at the position of $z = 0$ was taken. This slice was then zoomed by a factor f with $f \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$, in X and Y direction. The zooming of the volume slices by the factor f was done by zero padding in the frequency domain and by resampling with standard spatial domain filters. The synthetic function ρ_{mlext} with $z = 0$, was evaluated on an f times denser grid, to create a reference solution for this zooming operation. The reference solution was subtracted from the resampled slices. The spatial domain filters used in this experiment have a maximum filter extent of six samples. Samples outside of the 41×41 initial slices were assumed to be 0. These 0 valued samples influence the resampling in the border area of the resulting slices. To avoid too strong

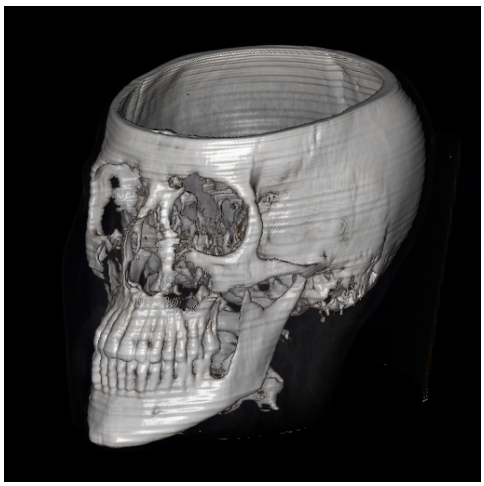
Description:
CT scan of the Stanford terra-cotta bunny.
The Stanford Bunny is famous as a
3D polygon mesh model, but also a CT scanned
dataset is available.
Dimension Original:
$512 \times 512 \times 360$ 12 Bit
Dimension Resampled:
$128 \times 128 \times 133$ 12 Bit

Figure 6.3: Characteristics of the Stanford Bunny dataset.



Description:
CT study of a cadaver head
Dimension Original:
$256 \times 256 \times 113$ 12 Bit
Dimension Resampled:
$128 \times 128 \times 113$ 12 Bit

Figure 6.4: Characteristics of the Stanford Head dataset.



Description:
CT scan of a Skewed Head
Dimension Original:
$184 \times 256 \times 170$ 12 Bit

Figure 6.5: Characteristics of the Skewed Head dataset.

influences, after scaling and subtraction of the reference image, a frame of 15 samples was set to 0 in the resulting slices. Finally the root-mean-square of these slices was used to draw Figure 6.6. It is possible to observe that the frequency domain based method has the lowest error. If the ρ_{mlext} dataset would have been sampled exactly according to the Nyquist criteria, a perfect scaling with the frequency domain based method would be possible. The difference to the synthetic reference image would be zero. The error we can observe in this experiment is referred to as aliasing.

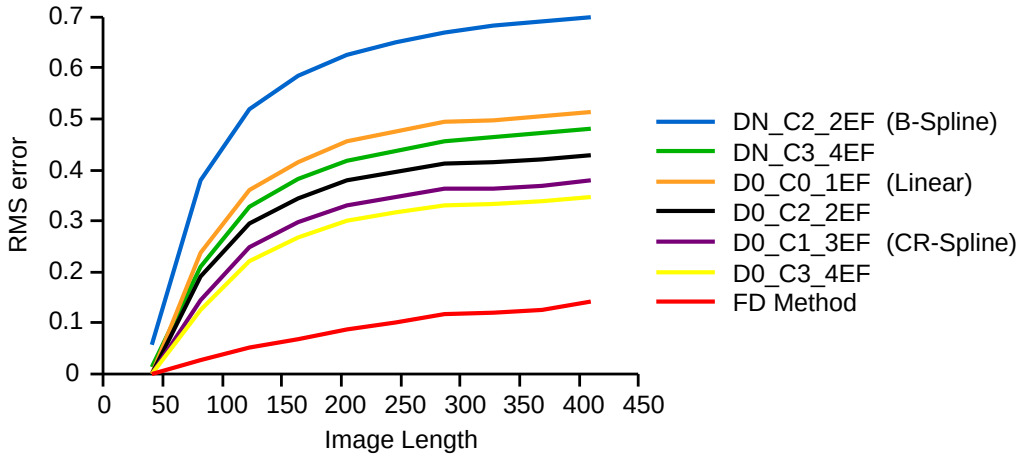


Figure 6.6: A 41×41 slice of the ρ_{mlext} function ($z = 0$) was zoomed with several spatial domain filters and by zero padding in the frequency domain. The root-mean-square of the difference image to the reevaluated ρ_{mlext} function ($z = 0$) was used to draw this diagram.

To visually demonstrate the quality of zooming with these methods, a slice ($x = 0$) of the ρ_{ml} and the ρ_{mlext} dataset was zoomed by a factor of 10.0 in Y and Z direction. To emulate the iso-surface extraction used by Marschner and Lobb [22], every data point with a value $f(x, y, z) < 0.5$ was set to black. The remaining values were rendered to a gray scale image (1.0 is white, 0.0 is black).

The ρ_{ml} dataset has a resolution of $41 \times 41 \times 41$, and the ρ_{mlext} dataset has a resolution of $41 \times 41 \times 80$. Therefore a slice taken at $x = 0$ from ρ_{ml} has a resolution of 41×41, a slice taken at $x = 0$ form ρ_{mlext} a resolution of 41×80. The lower half of the ρ_{mlext} slice is just a mirror of the upper one, as explained in Section 6.1. To create slices of the same size, for easier comparison, the lower half of the slices created from ρ_{mlext} was removed after the zooming process of the images in Figure 6.7 and Figure 6.8. The spatial domain filters where only applied to the ρ_{mlext} dataset.

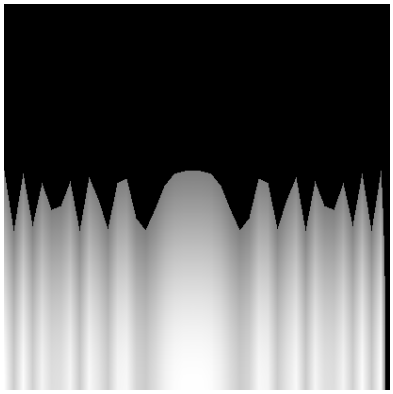
The slices of the first row in Figure 6.7, Figure 6.7(a), Figure 6.7(b) and Figure 6.7(c) were scaled with standard spatial domain filters. The reconstruction of the signal differs significantly form the synthetic reference image Figure 6.8(a).

The second row in Figure 6.7, is based on the standard ρ_{ml} test dataset. The slices where scaled by zero padding in the frequency domain. Each of the images shows strong ringing artifacts in Z direction (vertically). The reason for the strong ringing is a discontinuity in the dataset if periodicity in all three space dimension is assumed. For more details see Section 6.1.1. On top of the slices in the second row, the shifting effect described in Section 5.3 can be observed. The reason for this effect is, that during zooming new samples are created between existing ones. This happens between the last sample of the volume and the first sample of the next period as well (the dataset is assumed to be periodic in all three dimensions). This new samples are added at the end of the sample arrays (right and top).

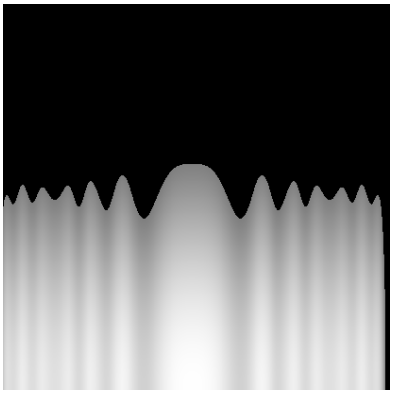
The third row in Figure 6.7, is based on the modified ρ_{ml} test dataset. Again the slices where scaled by zero padding in the frequency domain. There are no observable ringing artifacts. The shift effect is present again, but the additional samples have values below 0.5 therefore they are colored black.

As already mentioned the slices of the second and third row in Figure 6.7 were zoomed by zero padding in the frequency domain. Before zooming the slices in Figure 6.7(e) and Figure 6.7(h) a symmetric zero pad of one voxel was added at each end of the dataset in Y direction (left and right). This spatial domain zero pad is used to separate the spatial domain periods of the periodic signal. Spatial domain zero padding can be used to create an odd number of samples in one signal direction, this can give advantages explained in Section 5.2. The slices zoomed without spatial domain zero padding have a peak present at the very right side of the resulting image (see Figure 6.7(d) and Figure 6.7(g)). The most noticeable influence of the spatial domain zero padding is the shrinking of this peak.

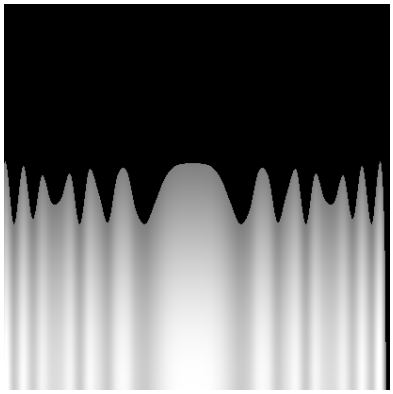
A common way of dealing with ringing artifacts is to apply a windowing filter in the frequency domain, as introduced in Section 3.6. The impact of windowing is demonstrated in Figure 6.7(f) and Figure 6.7(i) with a Hamming-Window filter. Some of the ringing in Figure 6.7(f) is smoothed out, but the conduct of the border edge is smoothed significantly too. The smoothing effect of the Hamming-Window is noticeable in an arched shortening of the peaks in the resulting image.



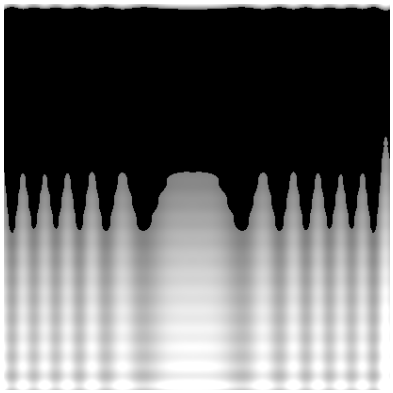
(a) D0 C0 1EF (Linear)



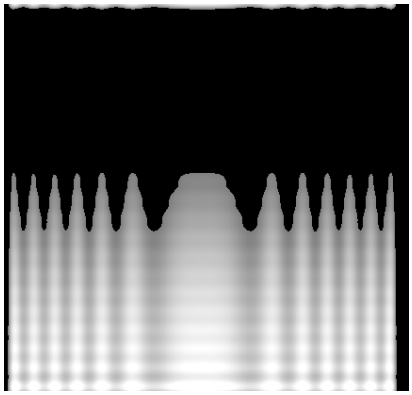
(b) DN C2 2EF (B-Spline)



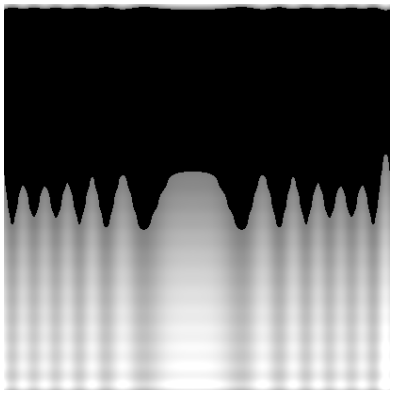
(c) D0 C1 3EF (CR-Spline)



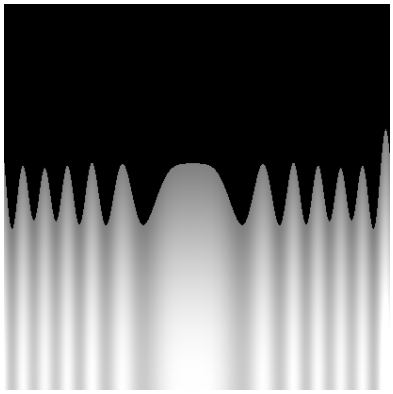
(d) FD based method, no zero pad in spatial domain, ρ_{ml}



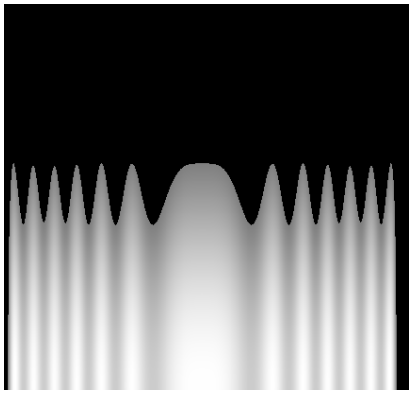
(e) FD based method, zero pad in spatial domain, ρ_{ml}



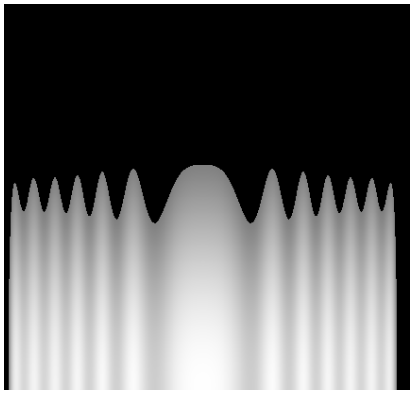
(f) FD based method, no zero pad in spatial domain, ρ_{ml}, Hamming-Window



(g) FD based method, no zero pad in spatial domain, ρ_{mlext}



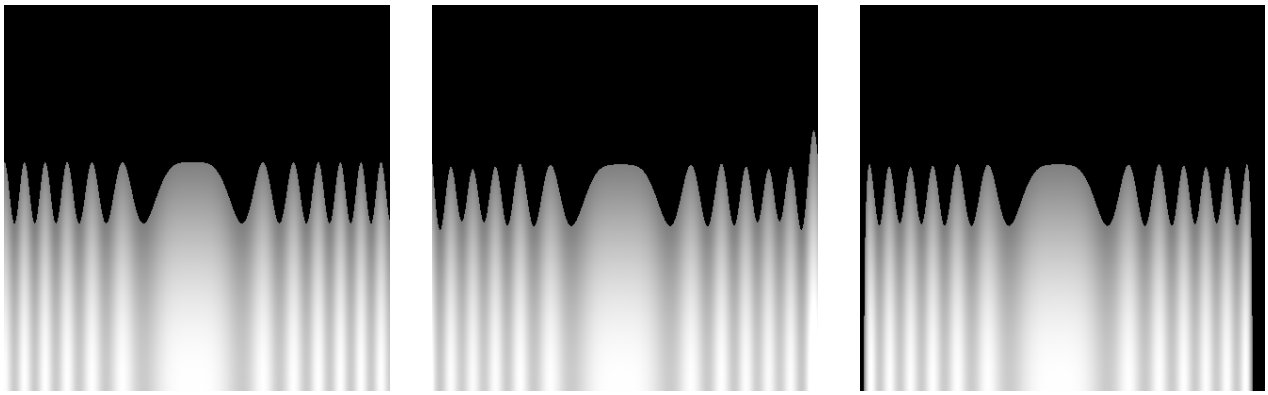
(h) FD based method, zero pad in spatial domain, ρ_{mlext}



(i) FD based method, zero pad in spatial domain, ρ_{mlext}, Hamming-Window

Figure 6.7: Resampled center slices ($x = 0$) of the ρ_{ml} and ρ_{mlext} dataset. The interpolation was done with different filters and zero padding in the frequency domain.

In Figure 6.8 the synthetic reference ρ_{mlext} (Figure 6.8(a)) is compared to the solutions of the frequency domain based resampling. The image in Figure 6.8(b) is obtained after zero padding in the frequency domain. This would be the optimal solution if the ρ_{mlext} function would have been sampled exactly according to the Nyquist criteria. In Figure 6.8(c) one voxel symmetric spatial domain zero padding was added to the slice taken from ρ_{mlext} before scaling it by zero padding in the frequency domain.



(a) Synthetic ρ_{mlext}

(b) FD based method, no zero pad in spatial domain, ρ_{mlext}

(c) FD based method, zero pad in spatial domain, ρ_{mlext}

Figure 6.8: Comparison of the synthetic function ρ_{mlext} to the slices resampled with frequency domain based methods.

6.2.2 Shifting

As mentioned earlier the quality critical operations of the introduced rendering algorithm are performed on volume slices. During the rendering process volume slices have to be resampled on a displaced grid. This process is simulated in this experiment while a comparison of the quality of the shifting with frequency domain techniques to spatial domain filtering is performed.

Initially a 41×41 slice of the ρ_{mlext} function, Section 6.1, with $z = 0$ was extracted. Sub-pixel shifts where applied in both X and Y direction. As a reference the function ρ_{mlext} was evaluated shifted by the same amount. All computed slices were compared to this reference solution. To avoid disturbing influences of the border regions, a 7 samples wide border area was set to 0. This size was chosen because it is the biggest used spatial domain filter kernel size (6) plus one additional sample. Therefore no sample that was influenced by information outside the 41×41 samples affects the result. The root-mean-square of these final error images was then used to draw Figure 6.9. It is easy to see that the frequency domain based method has the smallest deviation of the synthetic reference result. If ρ_{mlext} would have been sampled exactly according to the Nyquist criteria this difference would shrink to zero.

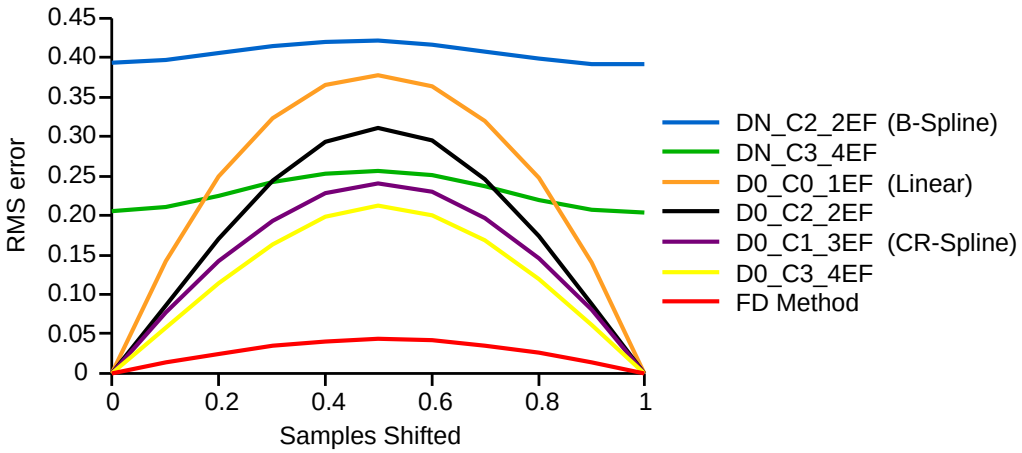


Figure 6.9: A 41×41 slice extracted from ρ_{mlext} with $z = 0$ image was shifted for sub-pixel steps with several spatial domain filters and phase shifts in the frequency domain. The result was compared to the ρ_{mlext} function evaluated on a grid shifted by the same amount. The root-mean-square of the difference image is shown in this diagram.

6.2.3 Rendering Speed

Even though speed was not the primary concern when developing this technique, a few timings should be provided.

The specification of the machines used for the measurements can be found in Table 6.2. For convenience they have been given the names Riga, Tallinn and Icarus.

	Riga	Tallinn	Icarus
CPUs	2 Intel Xeon(TM) 2.80 GHz	2 Intel Xeon(TM) 2.80 GHz hyper threading	64 Intel Itanium 2 1.50 GHz
Memory	2 GB	2 GB	60 GB

Table 6.2: Characteristics of the machines used for the performance measurements.

The dataset used for all the timing measurements was the ρ_{mlext} dataset with a resolution of $41 \times 41 \times 80$ voxels, as introduced in Section 6.1. The viewing vector was set to the main diagonal ($x = y = z = 1.0$), very close to a stack flip event. The coordinate axes (X,Y,Z) of the test dataset are directly related to the coordinate axes in standard object space (I,J,K). So the resolution in K direction N_K is 80 voxels. Further the sampling distance along the viewing rays was set to 0.1. It requires a $\sqrt{3} * 10.0$ times resampling in K direction. This can be done by adding a $\sqrt{3} * 10.0 * (N_K - 1)$ zero pad in frequency domain in K direction. For more detail on the resampling in the principal viewing axis see Section 4.4.2. The zoom factor for the scene was set to 10.0 in both directions I and J.

		Riga	Tallinn	Icarus
One Volume	K zero pad	0.087	0.069	0.029
	K derive	0.003	0.003	0.005
	K inverse FT	7.292	7.334	6.336
One Voxel Slice	I/J zero pad	0.006	0.005	0.002
	I/J phase shift	0.038	0.037	0.029
	I/J derive	0.004	0.003	0.006
	I/J inverse FT	0.106	0.104	0.078
	Compositing	0.006	0.005	0.015
	Warping	1.652	1.729	2.774
Entire Process	Preprocessing	8.952	8.721	7.570
	Rendering	888.225	854.540	773.294

Table 6.3: Timing of the rendering stages on three different machines. The time unit is seconds for each entry. The standard setup was rendered with one thread on each of the three machines.

Table 6.3 shows the timings of this setup rendered with one thread on each of the three machines Riga, Tallinn and Icarus. Every line in the table displays the timing of one stage of the rendering pipeline introduced in Section 4.4.

The timings in the first block are operations in K direction that are performed on the entire volume. In this rendering setup two volumes have to be processed, one for the density information, and another one for the derivative in K direction. The timing for calculating the derivative in K direction only counts for the second volume.

The second block of operations in I and J direction plus the compositing was timed on a per-slice basis. The number of slices that are processed depends on the resolution of the volume in K direction after the zero padding. As gradients are calculated this number of slices has to be multiplied by 4, one slice for the density information and one slice for the derivative in I, J and K direction. The resampling of the test dataset in K direction created a resolution of 1386 $(80 * \sqrt{3} * 10.0)$ slices. Taking the slices necessary for the gradient calculation into account, this adds up to 5544 slices for the entire rendering process.

In the third block the rendering process is split into preprocessing and actual rendering. The decision of where to split the rendering pipeline into preprocessing and actual rendering depends very much on which features are necessary for a certain rendering setup. If resampling in the K direction is required then the computation until rendering pipeline stage Figure 4.2(b), the three-dimensional Fourier transform, can be computed offline. If the resampling in K direction is set to a certain number of slices which should not change for different viewing directions, then all computation until rendering pipeline stage Figure 4.2(d) can be considered as preprocessing. This includes the three-dimensional Fourier transform, resampling by zero padding in K direction and the inverse transform in K direction. In Table 6.3 this second option for splitting the rendering pipeline was chosen.

Filter Kernel Extent Derivative Filter Kernel Extent	2 2	4 4	6 6	FD based method
Preprocessing	0.024	0.024	0.0236	8.951
Rendering	1883.693	2639.742	4553.953	888.225

Table 6.4: Timing comparison of the frequency domain based method to standard spatial domain filters with different kernel extents.

In Table 6.4 rendering times of the evaluation setup with standard spatial domain filters using several different kernel extents are compared to the frequency domain based method. In this comparative timing measurements the rendering pipeline introduced in this thesis was emulated with spatial domain filters. The filters were used for resampling of sample sequences. And the analytic derivatives of the interpolation filters were used to calculate derivatives. Therefore only the performance of the filtering is compared, not the performance of the new method to other algorithms (e.g., ray casting).

The spatial domain filtering was done with a moderately optimized C++ implementation, as compared to the high-performance FFTW [9] library, which is likely to influence the timing results in favor of the frequency domain based method. But the timings demonstrate that frequency domain methods are comparable in computational effort to spatial domain filtering. This is especially true when higher order filtering is used in spatial domain.

One drawback of the new method is that gradients are calculated for the whole volume, even if they are just used on a small portion of the voxels, like an iso surface, this leads to a considerable computational overhead that is not contributing to the final resulting image.

Multithreading

To demonstrate that the new method is suitable for parallel processing, a multithreaded implementation of the algorithm was executed on three multiprocessor machines. For their characteristics see Table 6.2. The standard setup for timing measurements with the ρ_{mlext} dataset was used, to maintain compatibility with Table 6.3 and Table 6.4. The scene was rendered by spawning p threads with $p \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. For more details on the parallelization of the rendering process see Section 5.6.3. The trend of the rendering times can be seen in Figure 6.10.

The rendering time of the dual CPU machine Riga remains at the same level for two or more threads.

The term hyperthreading refers to a second computation core on the CPU which enables the processor to compute two threads in parallel. Therefore the dual CPU hyperthreading machine Tallinn can process up to 4 threads in parallel. That is the reason why the speed up of the computation stagnates at the time level of 4 threads. The lower speed of Tallinn compared

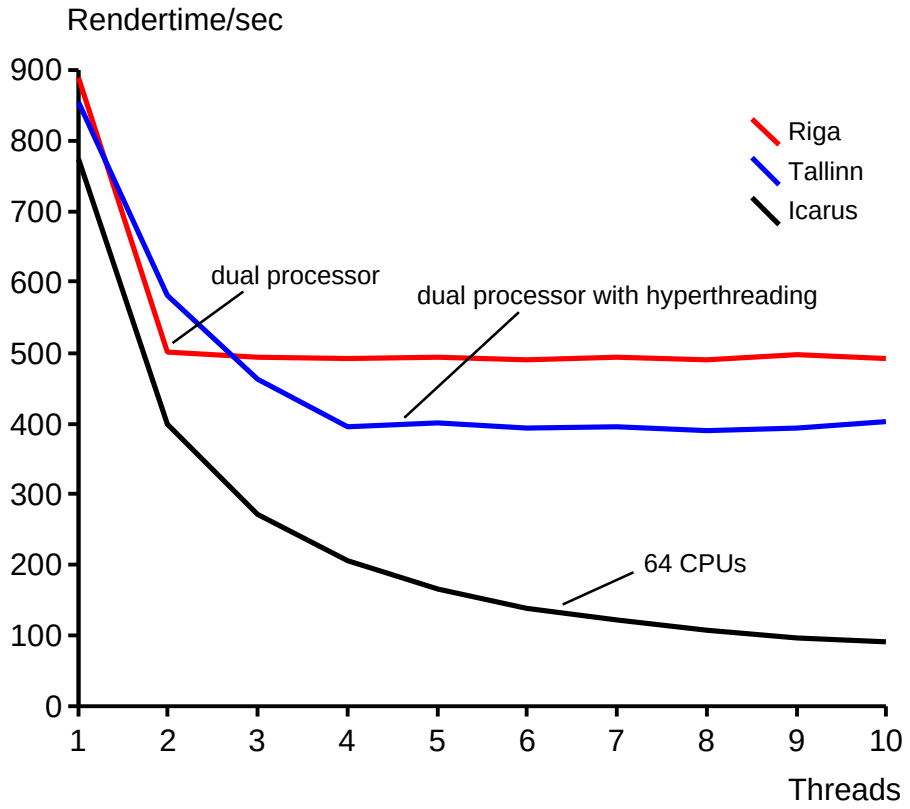


Figure 6.10: Test dataset rendered on three machines with an increasing number of parallel threads.

to Riga for two threads is probably caused by the scheduler who sometimes during the render process assigned both threads to the same CPU. As hyperthreading is not a perfect parallel computation this could impact the rendering speed in a negative way.

The timing of the 64 CPU mainframe Icarus behaves as expected. If the number of threads doubles, the rendering time is cut in half. It would be possible to use it up to 64 threads, but the concept is demonstrated.

6.2.4 Gradient Estimation

In the rendering algorithm introduced in this work the gradients are calculated by exploiting the derivative theorem of the Fourier transform (see Section 4.4.7). In order to compare the quality of this gradient estimation scheme, to standard methods that use analytic derivatives of interpolation filters in spatial domain, the ρ_{ml} and the ρ_{mlext} dataset were rendered as a benchmark. In Figure 6.11 the resulting images of the rendering with a sampling distance of 0.05 and a zoom factor of 10.0 are demonstrated. Figure 6.11(a), Figure 6.11(b), Figure 6.11(c) and Figure 6.11(d) demonstrate the result of spatial domain filters known from Marschner and Lobb [22].

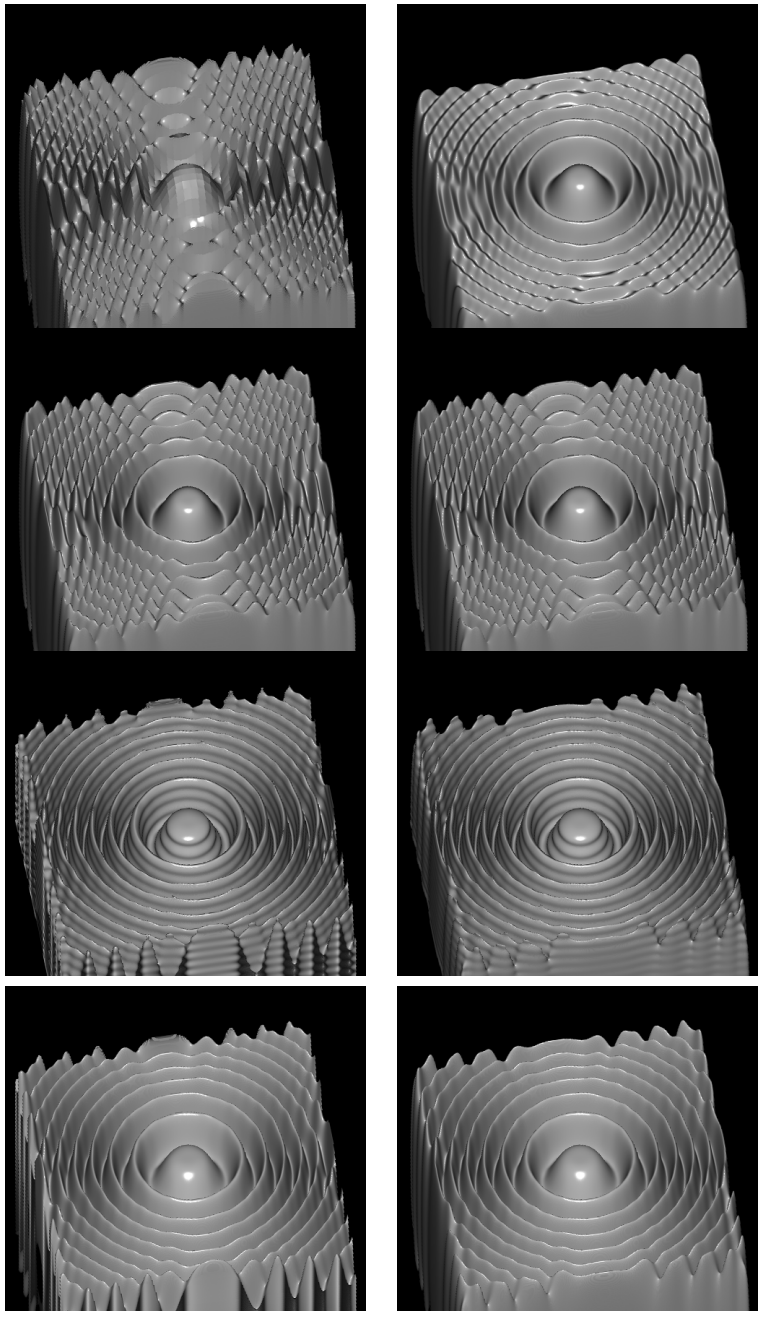
Figure 6.11(e) and Figure 6.11(f) show the rendering result of the original ρ_{ml} dataset. Strong ringing artifacts are visible in these images. The source of the ringing is a discontinuity in the dataset addressed in Section 6.1.1. These two images are a good example that the reconstruction with the *sinc* filter kernel can result in visually lower quality images than numerically less accurate interpolation methods.

Figure 6.11(g) and Figure 6.11(h) display the reconstruction quality of the frequency domain based method when the discontinuity of ρ_{ml} is resolved. These resulting images would be exactly the same as in the reference image (see Figure 6.1) if the synthetic dataset would have been sampled exactly according to the Nyquist criteria.

In Figure 6.11(f) and Figure 6.11(h) one voxel symmetric spatial domain zero pad was added in I and J direction, which apparently smooths the borders of the dataset (compare to Figure 6.7).

For better comparison of the accuracy of the gradient estimation, parallel to the normal rendering process a synthetic gradient vector was calculated by evaluating the derivatives of the analytic function. The angular discrepancy between the calculated and the analytic gradient was used to draw the images in Figure 6.12. The gray value of 255 represents an angular error of twenty degrees. The rendering setup for Figure 6.12 was exactly the same as in Figure 6.11.

The gradient estimation of the frequency domain method in Figure 6.12(g) and Figure 6.12(h) is apparently more accurate than the results of the spatial domain methods. Perfect reconstruction would be possible if the dataset is sampled according to the Nyquist criteria.



(a) D0 C0 1EF (Linear)

(b) D0 C2 2EF (B-Spline)

(c) D0 C1 3EF (CR-Spline)

(d) D0 C3 4EF

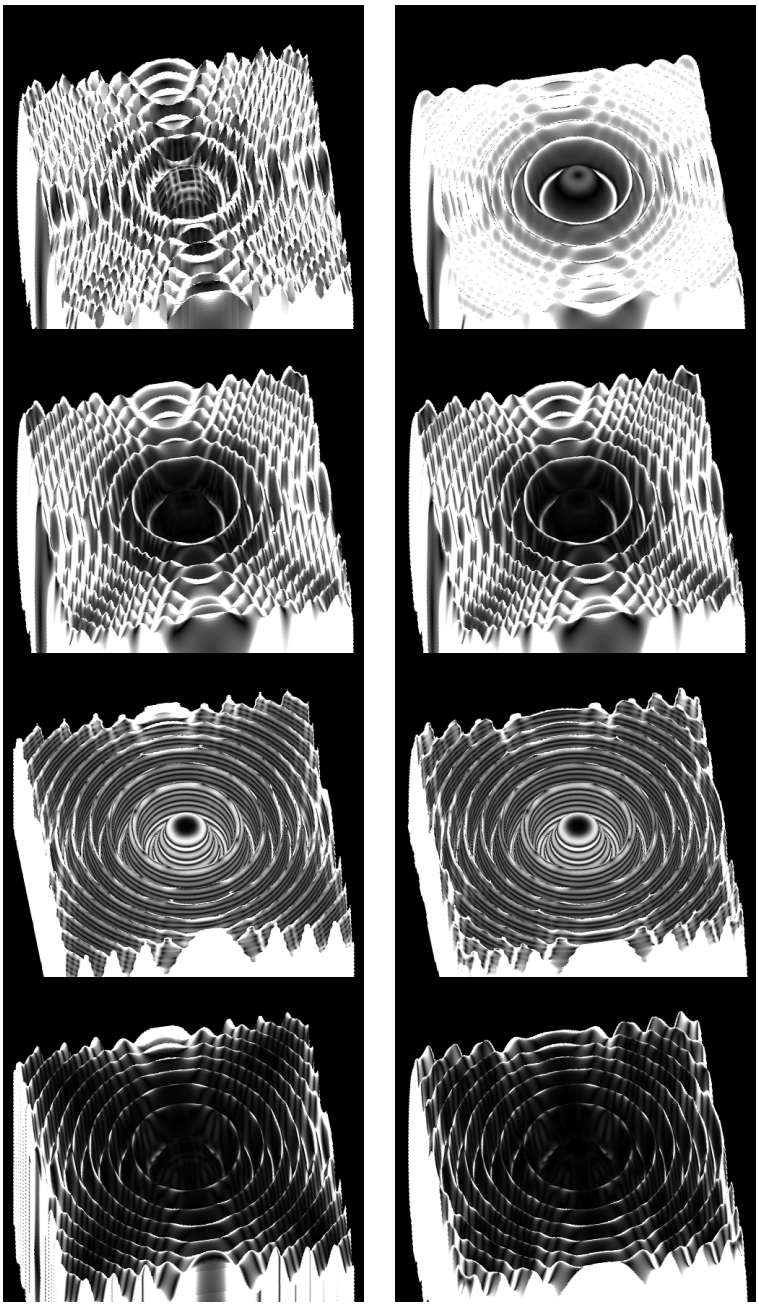
(e) FD based method, no zero pad in spatial domain, ρ_{ml}

(f) FD based method, zero pad in spatial domain, ρ_{ml}

(g) FD based method, no zero pad in spatial domain, ρ_{mlext}

(h) FD based method, zero pad in spatial domain, ρ_{mlext}

Figure 6.11: Iso surface extraction on the ρ_{ml} and the ρ_{mlext} dataset using spatial domain filters and the new frequency domain based method.



(a) D0 C0 1EF (Linear)

(b) DN C2 2EF (B-Spline)

(c) D0 C1 3EF (CR-Spline)

(d) D0 C3 4EF

(e) FD based method, no zero pad in spatial domain, ρ_{ml}

(f) FD based method, zero pad in spatial domain, ρ_{ml}]

(g) FD based method, no zero pad in spatial domain, ρ_{mlext}

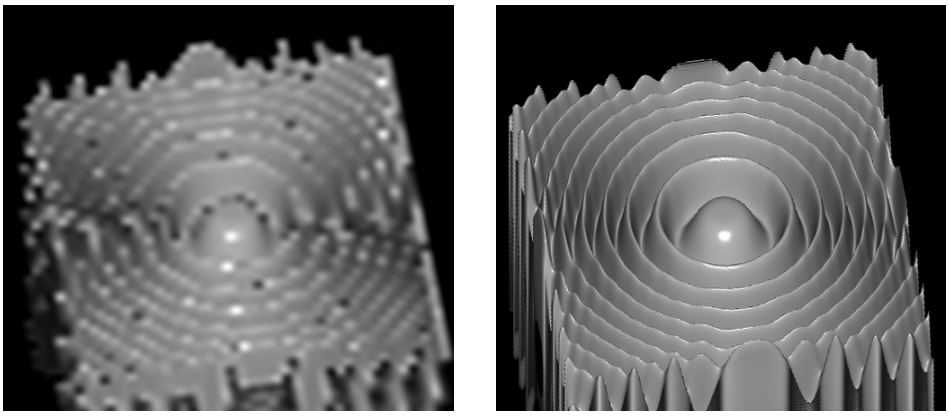
(h) FD based method, zero pad in spatial domain, ρ_{mlext}

Figure 6.12: Divergence images of the estimated gradient direction to the analytic reference gradient. The gray value of 255 represents an angular error of twenty degrees.

6.2.5 Zooming

In the standard shear-warp factorization as introduced by Lacroute [14] the voxels in the dataset and the pixel on the intermediate image have the same scale, and zooming is done only in the warping step. As mentioned by Sweeney and Mueller [32], this leads to considerable blurring artifacts at zoom factors greater than 2.0.

To achieve zooming with higher quality, in our approach every volume slice is resampled before the projection onto the intermediate image. This creates a noticeable impact on the rendering speed, but leads to significant quality improvements in the resulting images. In Figure 6.13 the ρ_{mlext} dataset was rendered with a considerable high zoom factor of 10.0. The strong bluring when this zoom is performed in the warping stage of the rendering pipeline can be observed in Figure 6.13(a). In contrast zooming of the volume slices does create much sharper images, see Figure 6.13(b). A drawback is that the superior image quality is bought by a computational effort that is proportional to the number of volume slices higher.



(a) (b)

Figure 6.13: (a) Zooming at the warping stage or (b) before compositing influences the quality of the resulting image but also the rendering speed.

6.2.6 Resulting Pictures

The intention of this thesis is to introduce a new high quality rendering method. Therefore one of the most interesting aspects is how it behaves when rendering real-world datasets. In this section some real dataset common in volume rendering are displayed with the new frequency domain based method, and compared to renderings done with traditional spatial domain filtering.

The images in Figure 6.14, Figure 6.15 and Figure 6.16 are rendering results of the Stanford Bunny dataset introduced in Section 6.1.2. The dataset is displayed with a 4.0 times, a 10.0

times and a 20.0 times zoom, with the sampling distance in K direction set to 0.05 It is visible in all three figures that the rendering quality of the frequency domain based method is superior to the standard spatial domain filtering. Especially in the 20.0 times zoomed images of Figure 6.16 the frequency domain based result has apparently less reconstruction artifacts around the eye area.

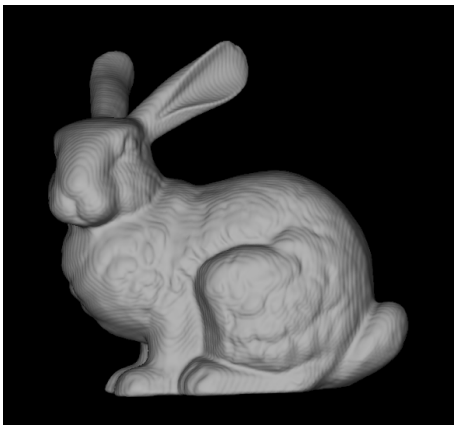
The images in Figure 6.17 are based on the Stanford Head dataset introduced in Section 6.1.2. The dataset is displayed with a zoom factor of 4.0 and a sampling distance in K direction of 0.05

The image in Figure 6.18 is based on the Skewed Head dataset presented in Section 6.1.2. It was added to demonstrate that the new frequency domain base method is capable of supporting transfer functions with translucent volume parts.

(a) D0 C0 1EF (Linear)

(b) DN C2 2EF (B-Spline)

(c) D0 C1 3EF (CR-Spline)

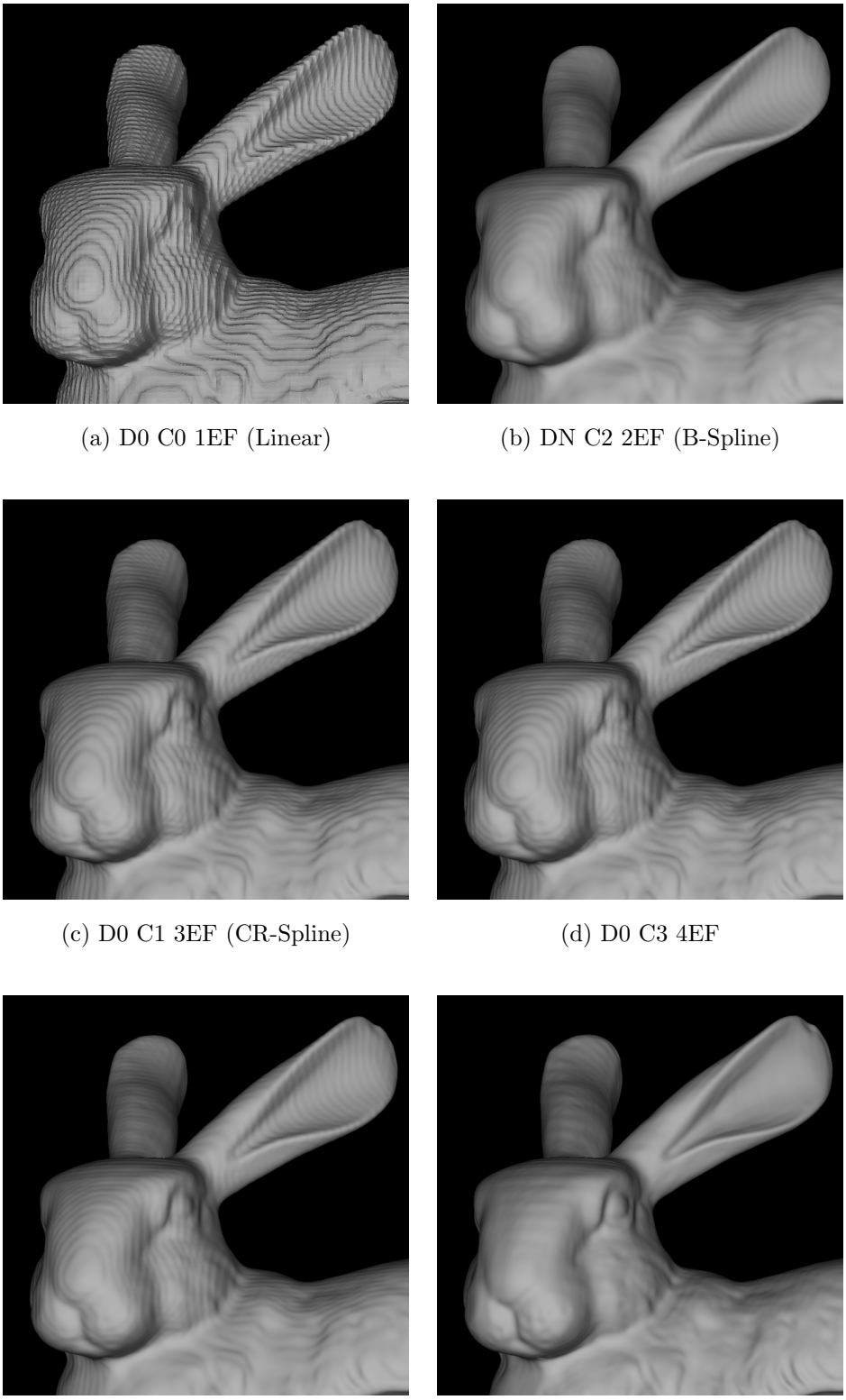


(d) D0 C3 4EF

(e) DN C3 4EF

(f) FD based method

Figure 6.14: The Stanford Bunny with resolution $128 \times 128 \times 133$, rendered with different filters and the new frequency domain based method. With a step length in Z direction of 0.05 and a zoom factor in X and Y direction of 4.0.



(a) D0 C0 1EF (Linear)

(b) DN C2 2EF (B-Spline)

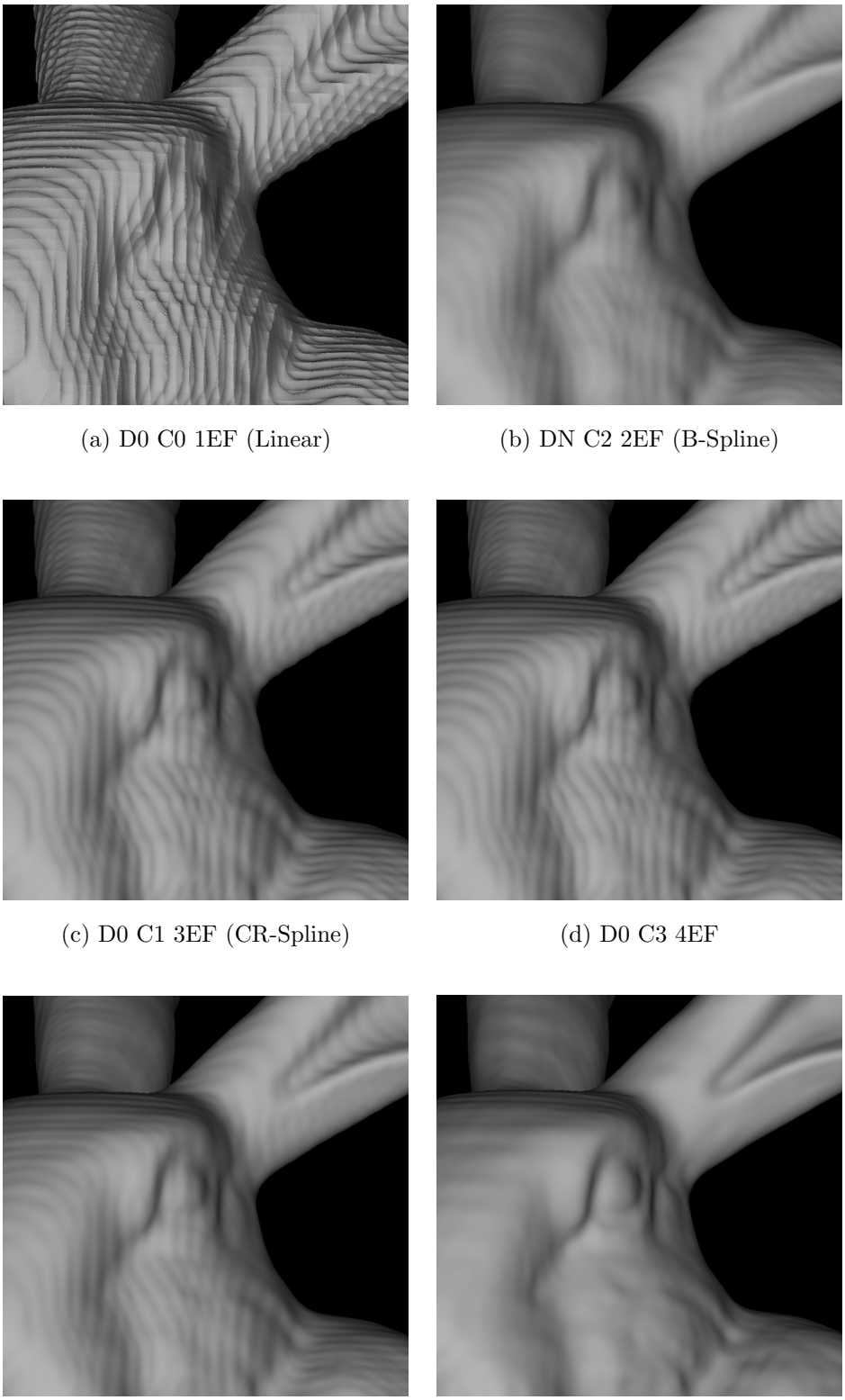
(c) D0 C1 3EF (CR-Spline)

(d) D0 C3 4EF

(e) DN C3 4EF

(f) FD based method

Figure 6.15: The Stanford Bunny with the render setup of Figure 6.14 but with a zoom factor in X and Y direction of 10.0.



(a) D0 C0 1EF (Linear)

(b) DN C2 2EF (B-Spline)

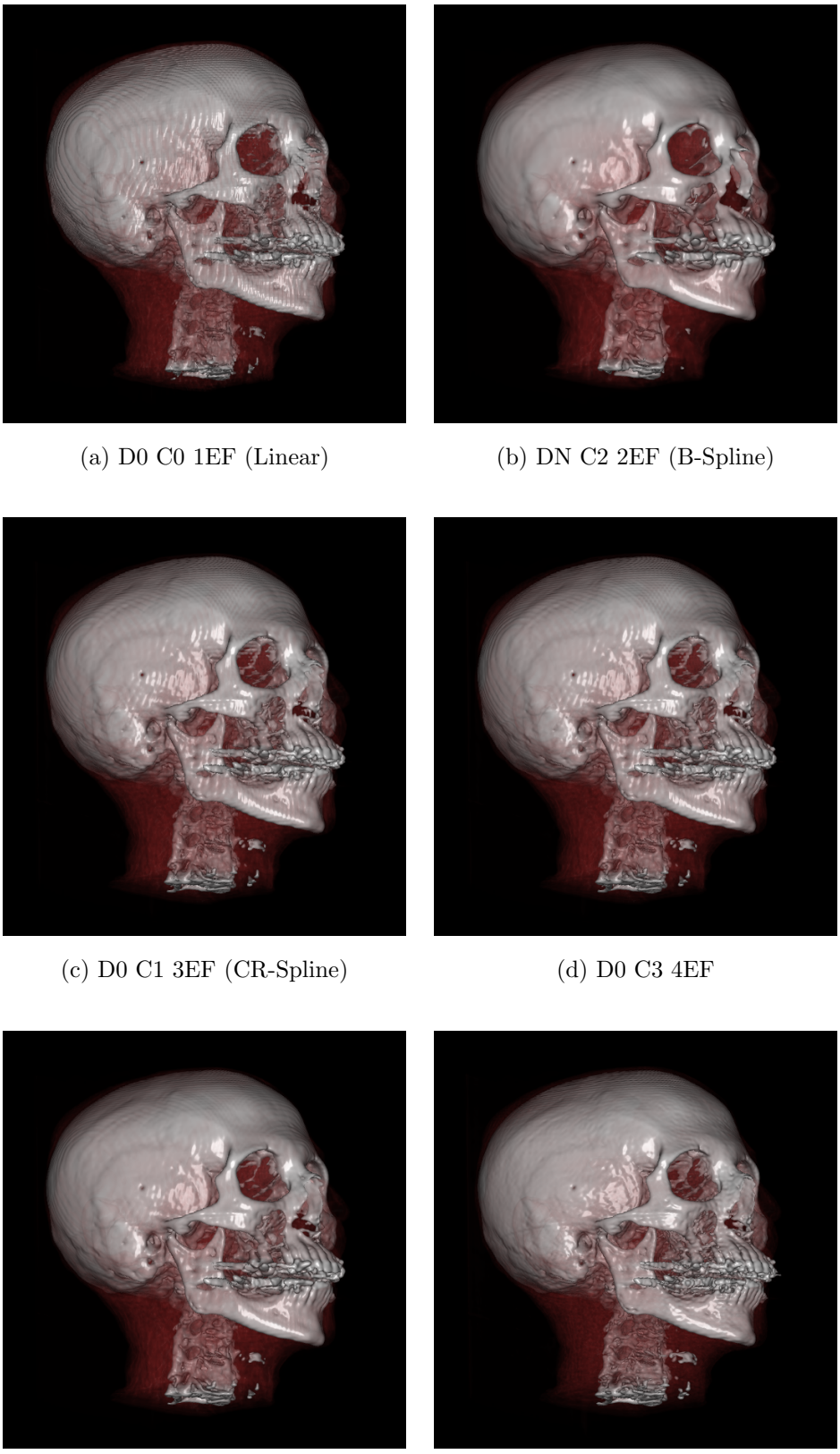
(c) D0 C1 3EF (CR-Spline)

(d) D0 C3 4EF

(e) DN C3 4EF

(f) FD based method

Figure 6.16: The Stanford Bunny with the render setup of Figure 6.14 but with a zoom factor in X and Y of 20.0.



(a) D0 C0 1EF (Linear)

(b) DN C2 2EF (B-Spline)

(c) D0 C1 3EF (CR-Spline)

(d) D0 C3 4EF

(e) DN C3 4EF

(f) FD based method

Figure 6.17: The Stanford Head dataset with resolution $128 \times 128 \times 113$, rendered with different filters and the new frequency domain based method. With a step length in Z direction of 0.05 and a zoom factor in X and Y direction of 4.0.

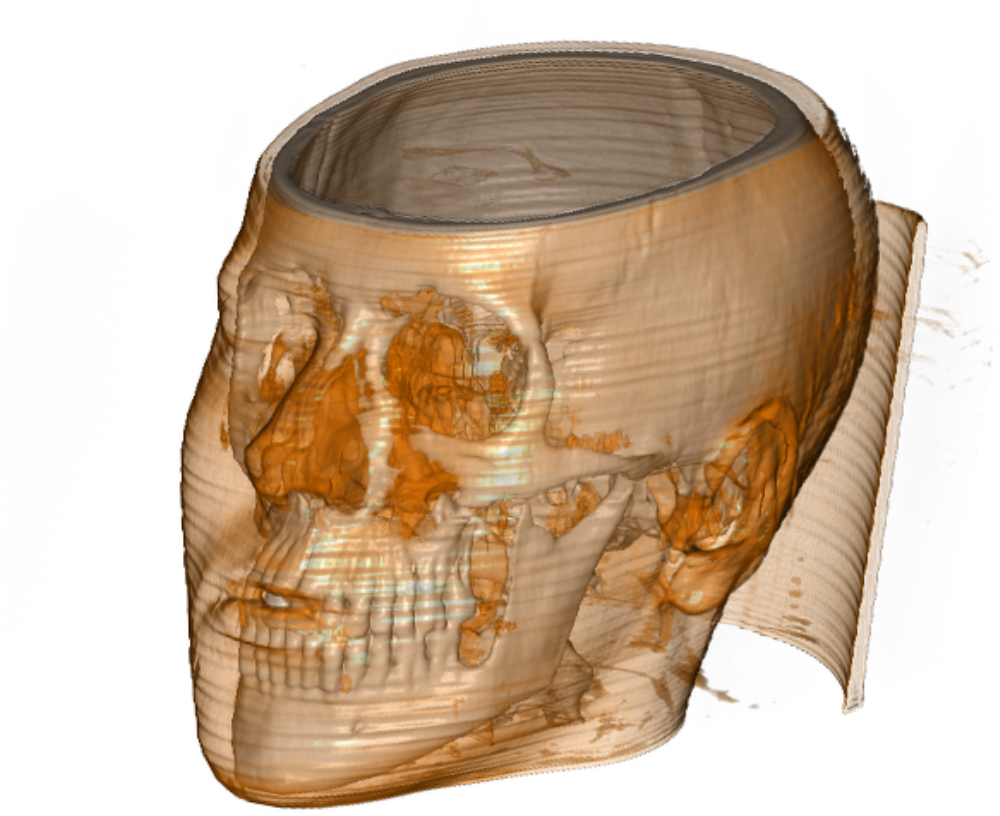


Figure 6.18: The Skewed Head dataset with resolution $184 \times 256 \times 170$, rendered with the new frequency domain based method. With a step length in Z direction of 0.05 and a zoom factor in X and Y direction of 2.0.

Chapter 7

Summary

All that we are is the result of what we have thought.

Buddha

The resampling of discrete signals is an important part of volume rendering algorithms that follow the ray casting approach introduced by Levoy [17]. For every pixel in the final image a viewing ray is cast into the scene. A numeric integration of the volume along the viewing rays is performed, which requires the computation of equidistant sample points along each viewing ray. The quality of the resulting image depends directly on the resampling filter used in this stage of the rendering process. Research has been done to improve the design of these spatial domain filters [26], and to evaluate and compare the quality of reconstruction filters [22].

As a basic principle all of these filtering methods are approximations of the *sinc* filter, which provides the perfect signal reconstruction. Unfortunately the *sinc* filter has infinite extent in the spatial domain, therefore it is usually dismissed as a theoretical solution. However the frequency domain representation of the *sinc* filter is a simple box filter. This leads to the assumption that volume rendering with *sinc* filter quality is possible if the resampling is done in the frequency domain. The Fourier transform was introduced for volume rendering by Dunne et al. [7]. The proposed algorithm, later referred to as frequency domain volume rendering (FDVR), or Fourier volume rendering (FVR), was further established by Malzbender [21], Levoy [19] and Totsuka and Levoy [34]. The FVR method is based on the projection slice theorem of the Fourier transform, which states that projection in the spatial domain is equivalent to slicing in the frequency domain. Therefore a two-dimensional slice s, passing the origin of the frequency domain representation of a three-dimensional volume, is resampled. An inverse Fourier transform of this slice s, is equivalent to a projection of the whole volume along the normal vector to s. If the size of the volume is N^3, then computational expense of this operation

is $\mathcal{O}(N^2 logN)$ as compared to $\mathcal{O}(N^3)$ of the pure spatial domain equivalent. Therefore the computational complexity of frequency domain volume rendering is lower than other traditional volume rendering approaches.

Unfortunately even with the most recent improvements by Lee et al. [15], Westenberger and Roederik [35] and Entezari et al. [8] which have added lighting effects, this method generates only "x-ray" like images (see Figure 2.14). The lack of occlusion and support of transfer functions are the major drawbacks of this method.

As the projection slice theorem does not provide the resampling quality required for high-quality ray casting we have to focus on other theorems of the Fourier transform. From the rich variety of theorems of the Fourier transform, the time shifting theorem, the packing theorem better known as zero-padding in the frequency domain, and the derivative theorem, presented by Oppenheim and Schafer [27], were selected and assembled to a volume rendering algorithm that performs resampling with *sinc* filter quality. This new method is conceptually based on the shear-warp factorization introduced by Lacroute and Levoy [14]. In the shear-warp factorization the viewing transformation of the volume from the object space into the image space is decomposed into a permutation, a shear, and a warp transformation. The shear transformation, where most of the critical interpolation takes place, only uses translation of volume slices along the coordinate axes. In this work we propose a new method performing the shear transformation in the frequency domain. To further improve the quality of the resulting images, two additional modifications of the standard shear-warp approach are introduced. First, a method is proposed for resampling intermediate slices before the shear operation is applied. This ensures that we obtain a steerable and viewing direction independent sampling distance along the viewing rays. Second, we introduce a technique to perform zooming in the standard object coordinate system as compared to zooming in the warping stage, which improves image quality significantly. Additionally a high-quality gradient estimation scheme based on the derivative theorem of the Fourier transform is presented.

During the work on this thesis a paper by Li et al. [20] was published that uses similar principles to our method, to perform the resampling of the volume in the frequency domain. Their approach is to decompose the transformation matrix into four shear operations. These four shear operations are performed by exploiting various frequency domain techniques and require multiple forward and backward Fourier transforms. In our approach the transformation matrix is factored according to the shear-warp factorization which requires only one shear operation to be executed in the frequency domain. Another issue is that if the volume is resampled by the application of shear operations it is necessary to add sufficient spatial domain zero-padding to fully accommodate the rotated volume. The problem, that arises if the spatial domain zero-padding is too small, is that parts of the data volume pass over the border of the volume and through the periodicity of the dataset enter from the other side. This error is

amplified by the consecutive shears. To allow arbitrary positions of the viewpoint, a symmetric spatial domain zero-pad of about $\frac{\sqrt{3}}{2}$ times the maximal volume resolution has to be applied. This creates an up to three times higher memory consumption as compared to our method that does not require spatial domain zero-padding of that amount. We further introduce a gradient estimation scheme that takes advantage of the derivative theorem of the Fourier transform, which could be also applied to their work.

7.1 The Rendering Pipeline

The algorithm proposed in this thesis is based on the shear-warp factorization introduced by Lacroute and Levoy [14]. The shear-warp factorization was created to be one of the fastest software based rendering algorithms. It gains its performance by factoring the projection matrix, that transforms the volume from the object space into the image space, into several submatrices. The transformation described by each of these submatrices can be computed very effectively and through that an imminent increase in rendering speed is possible.

In our method we use the same submatrices as in the shear-warp factorization, the main focus however is on high reconstruction quality in contrast to rendering speed. The shear-warp factorization has four rendering stages: permutation, shearing, compositing and warping. The permutation stage is a movement of voxels from one position in the volume to another, according to a certain permutation matrix, which is a lossless transformation. In the shearing stage the volume is dismantled into slices which are resampled on a sheared grid. The quality of this resampling process depends very much on the filter used for the reconstruction of the signal. In the proposed method we present a way of how to perform the shear by applying the time shifting theorem in the frequency domain. The next stage, i.e., the compositing, is equal to a numeric integration along the viewing rays. A major drawback of the standard shear-warp is that the sampling distance for the numeric integration can not be changed. Further this sampling distance is very coarse (≥ 1.0) and additionally viewing direction dependent. In our rendering pipeline we propose a resampling step that allows to perform the numerical integration along the rays with an arbitrary sampling distance, independent of the viewing direction. The last stage, i.e., the warping, transforms the intermediate image of the compositing stage into the final image. The warping in our method remains similar to the standard shear-warp warping, but to maintain the high-quality requirements a higher-order spatial domain filter is used. The quality loss through the resampling of the intermediate image in this stage does not create visible artifacts in the final image.

The adapted rendering pipeline for the new frequency domain based method is presented in Figure 4.2. During this section a detailed presentation of every stage of this pipeline and its contribution to the rendering process is given. The first stage of the rendering pipeline starts

from the standard object coordinate system, this means the volume is already permuted such that the K axis is the main viewing axis.

Multi-Dimensional Fourier Transform

The Fourier transform is a separable transformation. Every M-dimensional Fourier transform can be composed from M one-dimensional transforms in every dimension respectively. To transform the volume in all three space dimensions, a consecutive application of a 1D Fourier transform in each dimension is performed. Through the 3D Fourier transform the next pipeline stage is reached (see Figure 4.2(b)).

Resampling in the Principal Viewing Axis

In the standard shear-warp factorization the number of volume slices along the principal viewing axis K is kept constant for performance reasons. Therefore the distance of the sampling points along the rays vary with the viewing direction. Figure 4.3 shows how the sampling distance (s_1 vs. s_2) varies according to the view point settings. This variation can create artifacts that are especially visible in animations. Resampling of the volume in K direction allows to select an arbitrary sampling distance along the rays independent of the viewing direction. Resampling creates additional volume slices and is done by exploiting the packing theorem in the frequency domain.

To compute the necessary size of the zero-pad area, the sampling distance s along the viewing rays before resampling is calculated. The viewing vector $\vec{v}_{so}$, with its components $v_{so,i}$, $v_{so,j}$ and $v_{so,k}$ and the distance between the volume slices along the K axis d_k are needed. $\vec{v}_{so}$ is normalized in K direction by dividing each component through $v_{so,k}$ this gives $\vec{v}'_{so}$, with its components $v'_{so,i}$, $v'_{so,j}$ and $v'_{so,k}$ ($v'_{so,k} = 1.0$). The absolute length $|\vec{v}'_{so}|$ of the vector $\vec{v}'_{so}$ is the sampling distance if the volume slices are 1.0 apart. A multiplication with d_k, the real distance between the volume slices, gives the sampling distance before resampling. The setup for this calculation is illustrated by Figure 4.4.

The sampling distance before resampling s and the desired sampling distance s' are inversely proportional to the number of samples in K direction before resampling K, and the number of samples in K direction after the resampling K'. With Equation 4.4 the number of samples after zero-padding K' is calculated. The difference between K' and K is the amount of zero-pad necessary for the selected sampling distance s'. As zero-padding can only be applied in discrete amounts, K' has to be rounded to the closest integer number $K'_{pad} \in \mathbb{N}$. For sizes of $K'_{pad} > 50$, the error introduced to the sampling distance is already below $\pm 1.0\%$ of K'.

After the application of the zero-pad the pipeline stage Figure 4.2(c) is reached. The next stage is an inverse Fourier transform in K direction, Figure 4.2(d), while the I and J direction of

the volume remain in the frequency domain. It is important to remember that this resampling in K direction changes the distance between the slices which is especially relevant in the shearing stage.

Resampling of Volume Slices in I and J Direction

In the standard shear-warp factorization zooming is performed by scaling of the intermediate image. This approach leads to considerable blurring artifacts, especially for zoom factors greater than 2.0, as pointed out by Sweeney and Mueller [32]. In the proposed method zooming is performed earlier, in the standard object space where the rescaling is applied to the volume slices. A desired zoom factor is achieved by increasing the signal periods of each slice by zero-padding in the frequency domain. As adding samples is only possible in discrete steps a certain zoom factor can only be achieved with limited precision. If $I'_{pad} > 50$ and $J'_{pad} > 50$ then the deviation of the zoom factor is below $\pm 1.0\%$ of I' and J'. This error, if required, can be compensated by additional scaling in the warping step. After the application of the zero-pad the render process reaches the next stage (see Figure 4.2(e)).

Shearing

In the shearing stage of the shear-warp factorization the volume is transformed from the standard object space to the sheared object space. This causes the viewing direction to be perpendicular to the slices of the volume, respectively the (v,u) plane. Performing the calculations explained in detail in the Section 4.3.3, we acquire the shear coefficients (s_i, s_j) and the translation values (t_i, t_j). The values a_{ik} and a_{jk} describe the displacement of the k-th slice in i and j direction. For both directions the shifts by a_{xk} $(x \in \{i, j\})$ are split into a multiple of the voxel lengths, a_{xkSD} and the remainder, which is then a fraction of a voxel step a_{xkFD}, see Figure 4.6. As the subscripts already indicate the shift for a_{xkFD} is performed in the frequency domain, and the shift for a_{xkSD} in spatial domain. The shift in the spatial domain is actually only a movement of the slice in full voxel steps. The interpolation part is performed in the frequency domain. The reason for this split is to keep the shift in the frequency domain as small as possible to limit wrap around effects (see Figure 4.7). These effects appear because in the Fourier transform the volume slices are assumed to be periodic in I and J direction. If the whole shift a_{xk} would be performed in the frequency domain, a voxel leaving the visible window area on one side enters from the opposite side. This cyclic shift would lead to artifacts in the final image. To further limit the wrap-around effects a symmetric spatial domain zero-pad of one voxel has to be added to separate the periodic replicas in the spatial domain.

With the application of the shifting theorem to perform the a_{xkFD} shift, the rendering stage Figure 4.2(f) is reached. An inverse Fourier transform in I and J direction moves the rendering

process to stage Figure 4.2(g). The a_{xkSD} part of the slice displacement is applied in the spatial domain, which completes the transformation of the volume to the sheared object space, see Figure 4.2(h).

Compositing

During the compositing stage the resampled slices are blended into a 2D intermediate image, along the w axis. Transfer functions can be applied to the volume, as all kinds of algorithms to influence the appearance of the final image (i.e., shading, non-photorealistic effects, ...). In this work a high quality gradient estimation scheme is introduced, to obtain surface normals for lighting calculations.

The compositing of the slices along the w axis is comparable to a numeric integration along the viewing rays. During this process the density information of each slice $\hat{f}_{UV}[u, v]$ is transformed into an image $i_{UV}[u, v]$. Each pixel in $i_{UV}[u, v]$ has a color c and a transparency coefficient α. Every image is then composited into the slice located at $w = 0$, with the "over" operator [28]. The "over" operator states for a pixel a with color c_a and transparency α_a how it is composited over a pixel b with color c_b and transparency α_b. The resulting pixel values c_c and α_c can be computed with Equation 4.14 and Equation 4.15. The order in which the slices are composited is determined by their w coordinate value. The sign of the k component of the viewing vector $\vec{v}_{so}$ ($v_{so,k}$) in standard object space determines from which side of the stack the processing starts. If $v_{so,k}$ is positive then the slice with the coordinate $k = 0$ is the front slice; otherwise the slice at the other end of the stack, with the coordinate $k = k_{max}$ is the first slice. Compositing all slices using the "over" operator results into the non-warped intermediate image (see Figure 4.2(i)). This form of compositing is known as back to front (B2F) compositing, because it starts with the volume slice most apart from the view point. In standard ray casting the volume samples are composited in a reverse order, called front to back (F2B). The F2B compositing allows to stop the compositing process for one pixel as soon as the opacity of the allready composited volume samples along the ray reaches a certain threshold. This avoids the interpolation of samples further along the ray and is called early ray termination. The frequency domain based method perfoms a global resampling of the volume, therefore all volume slices are computed as a whole. A switch from B2F to F2B compositing would only create a negligible gain in rendering speed.

Warping

The 2D warping transformation applied to the intermediate image leads to the resulting image. The warping matrix transforms data points from the sheared object space into the final image space. This transformation compensates the viewing direction dependent scaling of the dis-

tances between the viewing rays (compare d_1 and d_2 in Figure 4.3), and performs the rotation component around the K axis.

If M is the maximal volume extension (maximum of I, J and K in standard object space), and z_{ij} is the scale factor applied to the volume, then an image buffer $f_{NN}[x_i, y_i]$ with $N = \sqrt{3} \cdot z_{ij} \cdot M$ can accommodate the resulting image. This calculation is based on the assumption that each side of the volume has length M and the main diagonal of the volume data cuboid is visible in its full length. Every pixel of $f_{NN}[x_i, y_i]$ in the image-coordinate space is transformed to the sheared-object space with the inverse of the warping matrix. The pixel values at the obtained coordinates are interpolated from the pixel values of the intermediate image. In order to maintain high quality in this step as well, a higher-order spatial domain filter is used for the resampling comparable to a Catmull-Rom spline [1].

Gradients

Since the gradient is the partial derivative of the original function and ideal interpolation with the *sinc* filter will reconstruct that function, the gradient can be reconstructed exactly by using the derivate of the *sinc* as a reconstruction kernel [2].

For the computation of the gradient vectors, three copies of the original dataset are created and Fourier transformed in all three space dimensions. Each one of these volumes is used to calculate one component of the gradient vector. Therefore each volume is derived in one of the three space dimensions by the application of the derivative theorem of the Fourier transform. Subsequently these gradient volumes are processed through the same rendering pipeline, as the density volume. The gradient volumes are combined to a volume of gradient vectors at the compositing stage. These gradient vectors are used with the processed original data to compute the intermediate image.

7.2 Implementation

The input signal in the Fourier transform in general is complex. In this application the volume data in the spatial domain is stored only in the real component and the imaginary component is set to zero. The real function is a Fourier transform pair to a Hermitian function [27]. Therefore a Fourier transform of this spatial domain representation of the volume leads to a frequency domain representation which is Hermitian. The Hermitian property signifies that corresponding sample values, with index k and $-k$, are conjugate complex to each other. The real components are equal, and the imaginary components have inverse sign. The reason for going into details about Hermitian functions is that it is dual to the real function by the Fourier transform. It means that all operations applied in the frequency domain have to preserve the

Hermitian property in order to obtain a real function after the inverse Fourier transform. The theorems of the Fourier transform that are used during the rendering process are symmetric around the origin and therefore do not alter the Hermitian property of a signal. But because of the periodicity of the signal there are certain indices in the discrete spectrum which need special attention. First there are sample points that are conjugate complex to themselves, and therefore the imaginary component has to be zero. This is the sample point at the origin, and if the the signal has length N with $N = 2k$ the sample point at index k. The second issue has an effect when calculating the derivative. This is done by exploiting the derivative theorem of the Fourier transform, which is a multiplication with $i\mu 2\pi\frac{1}{N}$ in the frequency domain. This multiplication with $i\mu 2\pi\frac{1}{N}$ essentially switches the real and imaginary component. After this switch, if the the signal has length N with $N = 2k$, the sample point at k still has to be conjugate complex to itself. To ensure the Hermitian property if the signal length is even, the real and imaginary component of the sample point at position k, has to be set to zero. Another way of getting around this problem is to add one zero valued sample at the end of the signal in the spatial domain, in order to create an odd number of samples (spatial domain zero-padding).

7.3 Results

In this section results and experiences with the application of the high quality resampling in the Fourier domain are presented. The reconstruction quality of the frequency domain techniques are compared to standard spatial domain filtering.

Test datasets

Several 3D datasets are used to compare the reconstruction quality of the frequency domain method to standard spatial domain filtering. Two categories of datasets are used, synthetic datasets and CT-scans.

Synthetic Test Datasets

To test the quality of the new interpolation method, the test function introduced by Marschner and Lobb [22] is used. In this work we refer to it as ρ_{ml}. This dataset is sampled almost according to the Nyquist criteria, 99.8% of the signal energy is captured.

When using the discrete Fourier transform (DFT) the assumption is that the signal is periodic and discrete in the spatial and the frequency domain. Through the sampling of ρ_{ml} we get one period of this signal, which is half the period of a sine wave in Z direction. In the zone between two of these periods, in Z direction, a jump in the signal is present. This discontinuity leads to high frequency components and a not band-limited behavior. In order to demonstrate

the impact of this discontinuity, but to maintain compatibility with the original signal ρ_{ml} a second synthetic dataset was created. The sample points of ρ_{ml} are mirrored along the $z = -1$ plane. The samples at $z = -1$ are not copied, because they are positioned at the plane of reflection. Further the samples at $z = +1$ are not copied as well to create smooth transitions in the periodic sequence of sine waves. This creates an extended version of the test dataset with the dimensions of $41 \times 41 \times 80$ samples. The extended test dataset is referred to as ρ_{mlext}.

Real-World Test Datasets

Three real-world datasets are used to demonstrate the reconstruction quality of the frequency domain based rendering method as compared to standard spatial domain filtering. The first two datasets introduced in Figure 6.3 and Figure 6.4 were originally created by Levoy and are provided for research purposes by "The Stanford volume data archive" [30].

The dataset introduced in Figure 6.5 is used in a visualization lab at the Vienna University of Technology [29].

Quality Comparison to Spatial Domain Filters

This section presents several experiments to show the advantages and disadvantages of the frequency domain based techniques as compared to spatial domain filtering. The following subsections address issues that arise in volume rendering.

Super Sampling

The purpose of this experiment is to compare the quality of zooming by zero-padding in the frequency domain to interpolation with spatial domain filters. As the newly introduced rendering algorithm is based on the shear-warp factorization, most of the rendering steps that are quality critical are performed on volume slices. To visually demonstrate the quality of zooming with these methods, a slice ($x = 0$) of the ρ_{ml} and the ρ_{mlext} dataset was zoomed by a factor of 10.0 in Y and Z direction. To emulate the iso-surface extraction used by Marschner and Lobb [22], the range of voxel values $f(x, y, z)$ from 0.0 to 1.0 was mapped to the gray scale color range 0 to 255. Afterwards the color of the data points with a value of $f(x, y, z) < 0.5$ was set to black.

The ρ_{ml} dataset has a resolution of $41 \times 41 \times 41$, and the ρ_{mlext} dataset has a resolution of $41 \times 41 \times 80$. Therefore a slice taken at $x = 0$ from ρ_{ml} has a resolution of 41×41, a slice taken at $x = 0$ from ρ_{mlext} a resolution of 41×80. The lower half of the ρ_{mlext} slice is just a mirror of the upper one. To create slices of the same size, for easier comparison, the lower half of the slices created from ρ_{mlext} was removed after the zooming process of the images in Figure 6.7 and Figure 6.8.

Rendering Speed

Even though speed was not the primary concern when developing this technique, the rendering performance compared to standard spatial domain resampling was analyzed.

Table 6.3 shows the timings of the standard setup rendered with one thread on three different machines. Every line in the table displays the timing of one stage of the rendering pipeline. The timings in the first block are operations in K direction that are performed on the entire volume. In this rendering setup two volumes have to be processed, one for the density information, and another one for the derivative in K direction. The timing for calculating the derivative in K direction only counts for the second volume. The second block of operations in I and J direction plus the compositing was timed on a per-slice basis. The number of slices that are processed depends on the resolution of the volume in K direction after the zero-padding. As gradients are calculated, the number of slices has to be multiplied by 4, i.e., one slice for the density information and one slice for the derivative in each particular I, J, and K direction. In the third block the rendering process is split into preprocessing and actual rendering. The decision of where to split the rendering pipeline into preprocessing and actual rendering depends very much on which features are necessary for the rendering setup. If resampling in the K direction is required then the computation until rendering pipeline stage Figure 4.2(b), the three-dimensional Fourier transform, can be computed offline. If the resampling in K direction is set to a certain number of slices which should not change for different viewing directions, then all computations until rendering pipeline stage Figure 4.2(d) can be considered as preprocessing. This includes the three-dimensional Fourier transform, resampling by zero-padding in K direction and the inverse transform in K direction. In Table 6.3 the second option for splitting the rendering pipeline was chosen.

In Table 6.4 rendering timings of the evaluation setup with standard spatial domain filters using several different kernel extents are compared to the frequency domain based method. In this comparative timing measurements the rendering pipeline introduced in this thesis was emulated with spatial domain filters. The filters were used for resampling of sample sequences. And the analytic derivatives of the interpolation filters were used to calculate derivatives. Therefore only the performance of the filtering is compared, not the performance of the new method to other algorithms (e.g., traditional ray casting).

The spatial domain filtering was done with a moderately optimized C++ implementation, as compared to the high-performance FFTW [9] library, which is likely to influence the timing results in favor of the frequency domain based method. But the timings demonstrate that the frequency domain methods are comparable in computational effort to spatial domain filtering. This is especially true when higher order filtering is used in the spatial domain.

One drawback of the new method is that the gradients are calculated for the whole volume, even if they are just used on a small portion of the voxels, like an iso-surface. This leads to a

considerable computational overhead which is not contributing to the final resulting image.

Gradient Estimation

In the rendering algorithm introduced in this work the gradients are calculated by exploiting the derivative theorem of the Fourier transform. In order to compare the quality of the frequency domain gradient estimation scheme, to standard methods that use analytic derivatives of interpolation filters in the spatial domain, the ρ_{ml} and the ρ_{mlext} dataset were rendered as a benchmark. In Figure 6.11 the resulting images of the rendering with a sampling distance of 0.05 and a zoom factor of 10.0 are demonstrated. For better comparison of the accuracy of the gradient estimation, parallel to the normal rendering process a synthetic gradient vector was calculated by evaluating the derivatives of the analytic function. The angular discrepancy between the calculated and the analytic gradient was used to draw the images in Figure 6.12. The gray value of 255 represents an angular error of 20°. The rendering setup for Figure 6.12 was exactly the same as in Figure 6.11.

The gradient estimation of the frequency domain method in Figure 6.12(g) and Figure 6.12(h) is apparently more accurate than the results of the spatial domain methods. Perfect reconstruction would be possible if the dataset is sampled according to the Nyquist criteria.

Resulting Pictures

One of the most interesting aspects, when introducing a new rendering algorithm, is how it behaves with real-world datasets. Some datasets common in volume rendering are displayed with the new frequency domain based method. For comparison the same computation was done with traditional spatial domain filtering.

The images in Figure 6.14, Figure 6.15 and Figure 6.16 are based on the Stanford Bunny dataset. The dataset is displayed with a 4.0, 10.0, and 20.0 times zoom, with the sampling distance in K direction set to 0.05. It is visible in all three figures that the rendering quality of the frequency domain based method is superior to the standard spatial domain filtering. Especially in the 20.0 times zoomed images of Figure 6.16 the frequency domain based result has apparently less reconstruction artifacts around the eye area.

The images in Figure 6.17 are based on the Stanford Head dataset introduced in Section 6.1.2. The dataset is displayed with zoom factor of 4.0 and sampling distance in K direction of 0.05.

The image in Figure 6.18 is based on the Skewed Head dataset presented in Section 6.1.2. It was added to demonstrate that the new frequency domain based method is capable of supporting transfer functions with translucent volume parts.

Chapter 8

Conclusions and Future Work

The future belongs to those who
believe in the beauty of their dreams.

Eleanor Roosevelt

We have developed a volume rendering algorithm that performs high-quality resampling in the frequency domain. We have demonstrated that this method can render well sampled volume datasets with higher quality than standard spatial domain resampling. Further a high-quality gradient estimation scheme, that provides very accurate surface normals for lighting calculations, was introduced.

The resampling and derivative computations are performed with *sinc* filter quality. This filter allows perfect reconstruction of the original signal if the sampling was done according to the Nyquist criteria. The Nyquist criteria states that in order to perfectly capture a signal, the sampling frequency has to be more than twice the highest frequency component of the original signal. Unfortunately, signals often have an unlimited frequency spectrum. These signals can only be captured with limited accuracy. Reconstruction from the samples of these signals with the *sinc* filter does not produce the original signal. The difference between the original signal and the reconstructed signal is called aliasing. Strong aliasing artifacts can cause such strong visual distortions that filtering with a numerical less accurate filter can result in more appealing resulting images. This effect was especially visible in the synthetic volume rendering test dataset ρ_{ml} introduced by Marschner and Lobb [22]. A standard test to judge the quality of a reconstruction filter is to render the sampled ρ_{ml} function and compare it to a reference image rendered by reevaluating the synthetic function. The conduct of the standard ρ_{ml} dataset in Z direction is half a period of a sine wave. For the Fourier transform it is assumed that the signal is periodic in all three space dimension. A discontinuity in Z direction is present in the standard ρ_{ml} dataset, through this periodicity, which creates very strong visual effects in the

resulting images. An alteration was applied to the dataset to remove this discontinuity. With this new dataset the capabilities of the new rendering algorithm for well sampled datasets was demonstrated.

Handling data in the frequency domain requires particular carefulness. Minor variation of the frequency domain can cause surprising effects in the spatial domain. The input data for the Fourier transform in general is complex. In our application the volume data in the spatial domain is stored in the real component, while the imaginary component is set to zero. This form of representation which has only non-zero values in the real component is called a real function. The transform pair to a real function by the Fourier transform is a Hermitian function. A function is Hermitian means that sample values at the index k are conjugate complex to the sample values at the index $-k$. This Hermitian property in the frequency domain has to be preserved to obtain a real function in the spatial domain. This is additionally complicated by the fact that standard Fourier transform implementations use an asymmetric indexing scheme. The basic concept of this scheme is that the origin of the data in the spatial and the frequency domain respectively correlate with the zero index of the sample array.

In the presented method resampling is done by zero padding in the frequency domain. Resampling to a higher resolution creates new sample points that are inserted between the existing samples. The first and the last sample point of a signal are neighbors through the periodicity in the spatial domain. During a resampling new sample points are inserted also between these two samples. The sample at index zero remains at its position after resampling, therefore the new samples introduced between the first and the last sample are appended to the sampling array. This creates, especially visible for high scaling factors, an apparent shifting effect toward the zero index. The counter measure against this shift effect, in the current implementation, is to remove the samples introduced between the first and the last index.

As usual there are still aspects that are worthwhile to investigate in more detail but where not completely addressed in this work.

- Reduction of memory consumption. This could be done by exploiting the fact that the data is given as a real function which leads to a Hermitian spectrum. A Hermitian spectrum has high symmetries so that up to halve of the memory could be conserved. The impact on the separability of the Fourier transform on such a data structure has to be investigated.

- Extension of the technique to perspective projection. Therefore scaling of the slices for arbitrary factors has to be derived. One possible solution could be to align each volume slice with the target grid through phase shifts in the frequency domain. For each inverse Fourier transform a column and a line vector of the resampled voxel slice is obtained. This would slow-down the render process considerably. A limitation of the viewing angle

to certain values that blend well with the already existing scaling by zero padding is a second approach to solve this issue.

- Improve rendering speed for zooming. At the moment, if images are zoomed, and only a small section of interest is to be displayed, the whole image has to be calculated, which obviously creates an unbearable slowdown. Partial inverse Fourier transforms could improve the rendering speed in these occasions.

Bibliography

[1] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann, Palo Alto, CA, 1988(?).

[2] Mark J. Bentum, Barthold B. A. Lichtenbelt, and Tom Malzbender. Frequency analysis of gradient estimators in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242–254, 1996.

[3] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics*, 16(3):21–29, 1982.

[4] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceeding of the 1994 IEEE Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, 1994.

[5] James W. Cooley and John W. Tukey. An algorithm for machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, April 1965.

[6] Alois Dornhofer. A discrete Fourier transform pair for arbitrary sampling geometries with applications to frequency domain volume rendering on the body-centered cubic lattice. Master's thesis, Vienna University of Technology, 2003.

[7] Shane Dunne, Sandy Napel, and Brian Rutt. Fast reprojection of volume data. In *Proceedings of the 1st International Conference on Visualization in Biomedical Computing*, pages 11–18, 1990.

[8] Alireza Entezari, Randy Scoggins, Torsten Möller, and Raghu Machiraju. Shading for Fourier volume rendering. In *Proceedings of the 2002 IEEE Symposium on Volume Visualization*, pages 131–138. IEEE, 2002.

[9] Fastest Fourier transform in the west (FFTW), 2004. http://www.fftw.org/.

[10] Pat Hanrahan. Three-pass affine transforms for volume rendering. In *Proceedings of the 1990 workshop on Volume Visualization*, pages 71–78. ACM Press, 1990.

[11] ISO/IEC. Audio recording - compact disc digital audio system. Technical Report 60908, ISO/IEC, 1999-02.

[12] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, 1986.

[13] Philippe G. Lacroute. *Fast volume rendering using a shear-warp factorization of the viewing transformation*. PhD thesis, Computer Systems Laboratory, Department of Electrical Engineering and Computer Science, Stanford University, 1995.

[14] Philippe G. Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics*, 28(Annual Conference Series):451–458, 1994.

[15] Zhenghong Lee, Pedro J. Diaz, and Errol M. Bellon. Analysis of Fourier volume rendering. In *IEEE Proceedings of the 1996 Fifteenth Southern Biomedical Engineering Conference*, pages 469–472, 1996.

[16] The Lenna story, 2004. http://www.lenna.org/.

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

[18] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, 1990.

[19] Marc Levoy. Volume rendering using the Fourier projection-slice theorem. In *Proceedings of Graphics Interface '92*, pages 61–69, 1992.

[20] Aili Li, Klaus Mueller, and Thomas Ernst. Methods for efficient, high quality volume resampling in the frequency domain. In *Proceedings of the 2004 IEEE Visualization*, pages 3–10. IEEE, 2004.

[21] Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, 1993.

[22] Stephen R. Marschner and Richard J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of Visualization '94*, pages 100–107. IEEE, 1994.

[23] Wolfram Research, 2004. http://mathworld.wolfram.com/.

[24] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[25] Michael Meißner, Jian Huang, Dirk Bartz, Klaus Mueller, and Roger Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE Symposium on Volume Visualization*, pages 81–90, 2000.

[26] Torsten Möller, Klaus Mueller, Yair Kurzion, Raghu Machiraju, and Roni Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 143–151, 1998.

[27] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, USA, 1989.

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

[29] Skewed Head dataset, 2004. http://www.cg.tuwien.ac.at/courses/Visualisierung/Angaben/Bsp1.html.

[30] The Stanford volume data archive, 2004. http://graphics.stanford.edu/data/voldata/voldata.html.

[31] Paul Stark. Fourier volume rendering of irregular data sets. Master's thesis, School of Computing Science, Simon Fraser University, 2002.

[32] John Sweeney and Klaus Mueller. Shear-warp deluxe: The shear-warp algorithm revisited. In *Proceedings of the 2002 Symposium on Data Visualisation*, pages 95–104. Eurographics Association, 2002.

[33] Thomas Theußl, Helwig Hauser, and Eduard Gröller. Mastering windows: improving reconstruction. In *Proceedings of the 2000 IEEE Symposium on Volume Visualization*, pages 101–108. ACM Press, 2000.

[34] Takashi Totsuka and Marc Levoy. Frequency domain volume rendering. *Computer Graphics*, 27(Annual Conference Series):271–278, 1993.

[35] Michael A. Westenberg and Jos B. T. M. Roerdink. Frequency domain volume rendering by the wavelet x-ray transform. *IEEE Transactions on Image Processing*, 9(7):1249–1261, 2000.

[36] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376, 1990.