UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

VRVis Research Center

GPU-Based Direct Volume Rendering of Industrial CT Data

Studienarbeit

im Studiengang Computervisualistik

vorgelegt von

Thomas Höllt

Betreuer: Dipl.-Ing. Dr. Markus Hadwiger
VRVis Forschungs-GmbH

Koblenz, im Juli 2007

Erklärung

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	■	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	■	☐

. . . .

(Ort, Datum) (Unterschrift)

Abstract

Computed tomography (CT) and magnetic resonance imaging (MRI) in the medical area deliver huge amounts of data, which doctors have to handle in a short time. This data can be visualised efficiently with direct volume rendering. Consequently most direct volume rendering applications on the market are specialised on medical tasks or integrated in medical visualisation environments. Highly evolved applications for tasks like diagnosis or surgery simulation are available in this area.

In the last years, however, another area is making increasing use of computed tomography. Companies like phoenix|x-ray, founded in 1999 produce CT-scanners especially dedicated to industrial applications like non destructive material testing (NDT). Of course an application like NDT has different demands on the visualisation than a typical medical application. For example a typical task for non destructive testing would be to highlight air inclusions (pores) in a casting. These inclusions usually cover a very small area and are very hard to classify only based on their density value as this would also highlight the air around the casting.

This thesis presents multiple approaches to improve the rendering of industrial CT data, most of them based on higher dimensional transfer functions. Therefore the existing volume renderer application of VRVis was extended with a user interface to create such transfer functions and existing render modes were adapted to profit from the new transfer functions. These approaches are especially suited to improve the visualisation of surfaces and material boundaries as well as pores. The resulting renderings make it very easy to identify these features while preserving interactive framerates.

Kurzfassung

Computer Tomographie und Magnet Resonanz Tomographie, im medizinischen Bereich liefern riesige Mengen an Daten, die Ärzte in kurzer Zeit verarbeiten müssen. Diese Daten können mit Hilfe von Direct Volume Rendering effizient verarbeitet werden. Folgerichtig sind die meisten auf dem Markt befindlichen Direct Volume Rendering Anwendungen auf medizinische Aufgaben ausgerichtet oder in Umgebungen zur medizinischen Visualisierung integriert. In diesem Bereich gibt es hochentwikkelte Applikationen für den Einsatz in der Diagnostik oder zur Operationssimulation.

In den letzten Jahren wurde der Nutzen von Computer Tomographie aber auch in einem anderen Bereich entdeckt. Firmen wie die 1999 gegründete phoenix|x-ray stellen CT-Scanner speziell für den Einsatz im Industriellen Bereich mit Anwendungen wie zerstörungsfreier Werkstoffprüfung (NDT) her. Natürlich stellt eine Anwendung wie NDT andere Ansprüche an die Visualisierung als eine typische medizinische Anwendung. Beispielhaft für eine NDT Anwendung wäre die Markierung von Luft-Einschlüssen in einem Gußteil. Üblicherweise bedecken diese Einschlüsse nur eine sehr kleine Fläche und sind sehr schwer nur anhand ihres Dichtewerts zu klassifizieren, da die das Gußteil umgebende Luft so ebenfalls klassifiziert und damit hervorgehoben würde.

In dieser Studienarbeit werden verschiedenste Ansätze vorgestellt, die zum Ziel haben das Rendering von industriellen CT-Daten zu verbessern. Die Mehrzahl hiervon basiert auf höher dimensionalen Transferunktionen. Die exisiterende Volumenrendering Anwendung des VRVis wurde hierfür um eine Benutzerschnittstelle zur Erstellung solcher Transferfunktionen erweitert und die existierenden Rendermodi an diese angepasst.
Diese Ansätze sind speziell darauf ausgerichtet die Visualisierung von O-

berflächen, Materialgrenzen und Einschlüsse zu verbessern. Die resultierenden Renderings machen es sehr einfach diese Merkmale zu identifizieren wobei das Rendering interaktiv bleibt.

Contents

List of Figures

Chapter 1

Introduction

This thesis describes the development of GPU accelerated direct volume rendering modes especially focused on visualisation for industrial purposes, i.e. non-destructive testing (NDT), mainly incorporating higher dimensional transfer functions. The implemented rendering modes produce interactive frame rates on modern consumer graphics hardware (GPUs).

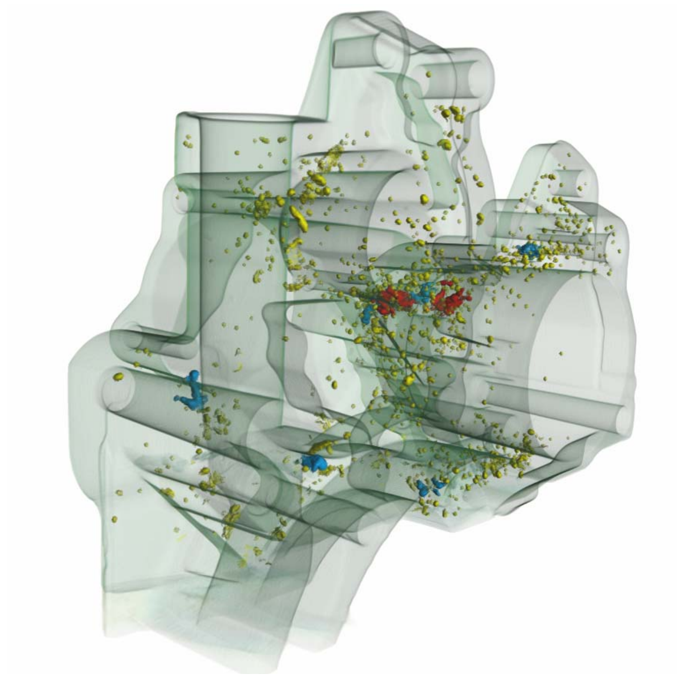


Figure 1.1: Visualisation of pores of different sizes in a cast housing.

1.1 Problem Statement and Objectives

Industrial material testing mainly consists of two large sectors:

The first one is destructive material testing, like crash tests or hardness tests. These tests are carried out until the specimen's failure. For example material of a sample of a charge of castings is ablated slice by slice to detect pores or to specify the material quality in general. Of course, as this leads to the destruction of the part this can only be done exemplarily.

The second sector is non destructive testing, which is important for testing the quality of parts which are actually in use. Mostly these are tests for safety reasons, like the testing of parts of an aeroplane or buildings. The British Institute for Non-Destructive Testing [BIN07] describes the covered methods as follows:

> The subject of NDT has no clearly defined boundaries; it ranges from simple techniques such as visual examination of surfaces, through [...] radiography, ultrasonic testing, magnetic particle crack detection, to [...] the measurement of Barkhausen noise and positron annihilation.

Normally non-destructive testing results are a lot harder to interpret than the results gained from destructive testing. However since a few years computed tomography is exploited for non destructive testing, covered within radiography. The visualisation of data gained from computed tomography was a very important topic in computer graphics, especially with the rapid development of consumer graphics hardware which made direct volume rendering of these data possible at real time frame rates. Despite the research being focused on medical data a lot of the know how can be ported to the visualisation of industrial data. Thus for example with simple isosurface rendering destructive testing methods like the slice by slice ablation mentioned before can be simulated making the interpretation just as simple as when ablating the real material.

In this thesis the development of render modes especially focused on the rendering of industrial CT data is described. This means that render modes especially suited for the visualisation of boundaries on the one hand and pores on the other hand will be developed. Besides a small extension to the existing render modes for surface visualisation the main work was dedicated to the integration of two dimensional transfer functions which are a key extension to improve classification of the above mentioned features.

1.2 Structure

This thesis is structured in the following way. Chapter 2 gives a brief overview of the fundamentals of this work. It starts with a description of computed tomography and its utilisation in the industrial area, followed by a short overview of direct volume rendering, an important technique to visualise the images gained from CT, focusing mainly on hardware accelerated raycasting as this will be the method used in this work. Very important for image quality in direct volume rendering are so called transfer functions. This concept will also be explained in this chapter. Finally the VRVis hardware volume renderer, acting as basis for this work will be presented.

The following Chapter 3 constitutes the main part of this work. The new render modes and the changes in the hardware volume renderer will be described in this chapter alongside their implementation details. It is divided in five sections. The graphical user interface for the creation of two dimensional transfer functions will be described first. Secondly a technique to correctly illuminate clipping surfaces in volume rendering using gradient based lighting is presented. Part three, four and five each describes a new render mode exploiting a two dimensional transfer function.

The results of this work are presented in Chapter 4. First an overview of the performance of the new render modes in comparison to existing ones will be given, followed by an evaluation of image quality of the new render modes.

Chapter 5 summarises the results and gives an outlook into possible future work.

Finally in Appendix A a usage guide to the newly implemented panel for creating two dimensional transfer functions is given. All the features of the panel are presented here without the implementation details.

Chapter 2

Fundamentals

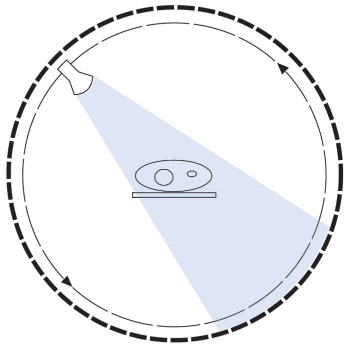
In the following chapter the fundamentals of this work are presented. In Section 2.1 Industrial Computed Tomography (CT) is introduced. The development of CT imaging and the differences between medical and industrial solutions are shown. Section 2.2 gives a brief overview of direct volume rendering, an approach to visualise the images CT delivers in 3D. As the transfer function is not only the dominating factor for the visualisation in most direct volume rendering approaches but especially in the render modes which will be presented in Chapter 3, Section 2.3 will explain the concept in detail. Finally Section 2.4 introduces the VRVis Hardware Volume Renderer which serves as basis for this work.

2.1 High Resolution Industrial CT

Computed Tomography is an imaging method traditionally mainly used in the medical area. It is a further development of the conventional x-ray imaging first done by Wilhelm Conrad Röntgen in 1895 [HP06, slide 13]. Multiple images of the same object from different perspectives are taken to reconstruct a 3D image.

The setup of a typical medical CT-scanner can be seen in Figure 2.1(b): While the object is fixed in the centre an x-ray tube is rotating around the object emitting radiation which is then measured by either a detector rotating with the tube or a detector ring covering the whole area. The first CT scanner like this was built by Godfrey Hounsfield and Allan Cormack in 1971. Since then the image quality and processing speed took a giant

(a) Somatom



(b) setup

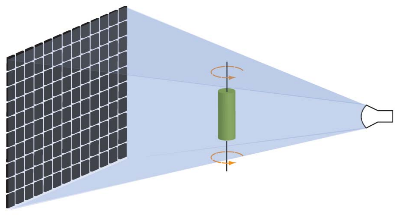
Figure 2.1: Medical CT scanner *Somatom* by Siemens Medical and the typical setup. Image (a) courtesy of Siemens AG, Medical Solutions.

leap forward. While in 1972 each slice measured 80 by 80 pixels with only 8 intensity levels and the reconstruction had to be done overnight, today CT scanners like the Siemens Somatom shown in Figure 2.1(a) deliver up to 64 slices per rotation, 512 square pixels each and the reconstruction can be done on-the-fly [HP06, slides 44 & 45].

In contrast to the early years after Röntgen discovered the use of x-rays when x-ray radiographs were en vogue, today the dangers of radiation are well known and exposure is minimised whenever possible. However in industrial applications like non destructive material testing it is not necessarily important to restrict the radiation the subject is exposed to. For this purpose industrial CT scanners like the phoenix|x-ray nanotom shown in Figure 2.2(a) were developed. While the functionality of both of these systems is very similar the setup is a bit different. Industrial scanners usually rotate the subject while the x-ray tube and the detector (either a row- or a flatpanel-detector) are in a fixed position (see Figure 2.2(b)). The tube and the detectors can be moved towards the subject to influence the mapping value, and in some setups parallel to the subject to make a multipass scan and stitch the resulting images to one large image.

While Siemens claims to have the industry's highest spatial resolution of 0.33mm [Sie06] the nanotome can reach a voxelsize as small as 0.5μ. This increase of resolution nearly by the factor seven of course leads to very large data sets whose handling is one of the main tasks, besides the different visualisation demands for an industry focused rendering solution.

(a) nanotom



(b) setup

Figure 2.2: Industrial CT scanner *nanotom* by phoenix|x-ray and its setup. Image (a) courtesy of phoenix|x-ray Systems + Services GmbH.

2.2 Direct Volume Rendering

Direct volume rendering (as well as indirect volume rendering) is a way to visualise three-dimensional volume data on a two-dimensional computer screen. While indirect volume rendering methods like the marching cubes algorithm [LC87] extract a surface from the volume data in a preprocessing step before rendering, direct volume rendering methods operate directly on the volume data. The goal of this approach is to simulate the light transport through the medium. This is done by using optical models to simplify the light transport equation [EHK$^+$06, p8, formula 1.6]:

$$\frac{dI(s)}{ds} = -\kappa(s)I(s) + q(s) \tag{2.1}$$

A very common model for direct volume rendering is the so called emission-absorption-model. This model is based on a low albedo situation [Bli82] meaning scattering and higher order lighting effects can be ignored. The effect of any element of the participating medium is reduced to an emitting and an absorbing part (Figure 2.3 illustrates this model). Thus the light transport equation can be approximated by

$$I = \int_{s_0}^{s_n} c(t)e^{-\tau(t,s_n)}dt \tag{2.2}$$

with the emission coefficient c and the absorption defined by the optical depth τ. $c(t)$ actually returns an rgb colour due to the use of a transfer function, mapping the density value of the volume data to an rgbα-colour (see Section 2.3 for more details). The α value returned by the transfer function determines the optical depth.

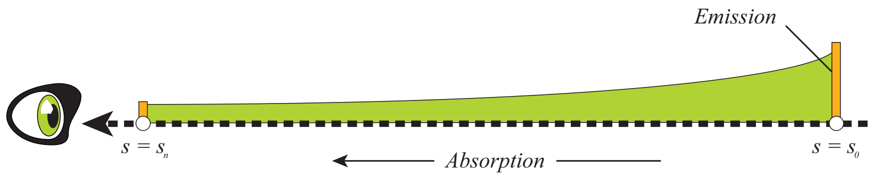


Figure 2.3: Illustration of the emission-absorption-model [EHK$^+$04].

Due to its very good image quality and the fact that interactive framerates can be achieved when using modern programmable graphics hardware one of the most popular direct volume rendering methods is *raycasting* [Lev88]. As an image order approach for every pixel in the rendering a ray is cast through the volume compositing the resulting value from sample points on its way.

The first raycaster done on graphics hardware [KW03] used a multipass approach. With the implementation of shader model 3.0 in the nVIDIA Geforce 6-series [nVI04] introduced in 2004 and later on in the ATI Radeon 1x00-series [AMD05] in 2005 it was possible to implement single pass approaches as these graphic processing units support loops as well as dynamic branching in the fragment shader. Loops are essential for casting the ray through the volume in a single pass, while dynamic branching can be used to improve performance for example by stopping the raycasting process early (early ray temrination).

The main raycasting pass proceeds in the following way: The volume is loaded onto the graphics board stored in a three dimensional texture. The entry position for each ray as well as the direction and the length are computed by rendering a bounding box of the volume (in $[0, 1]$ range) with the colour of each vertex in (r, g, b) equal to its position in (x, y, z)-volume-coordinates. The frontfaces (Figure 2.4(a)) of this box are rendered into a texture representing the starting positions of the rays. The difference of the backfaces (Figure 2.4(b)) and the frontfaces results in the direction texture (Figure 2.4(c)), which holds the direction of each vector encoded in its rgb values as well as the length in the corresponding α-value.

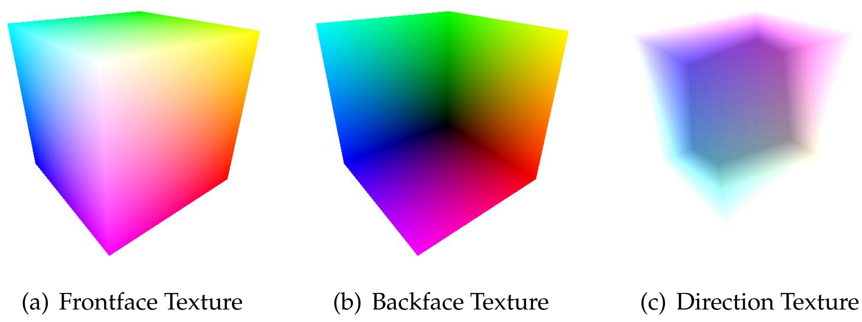


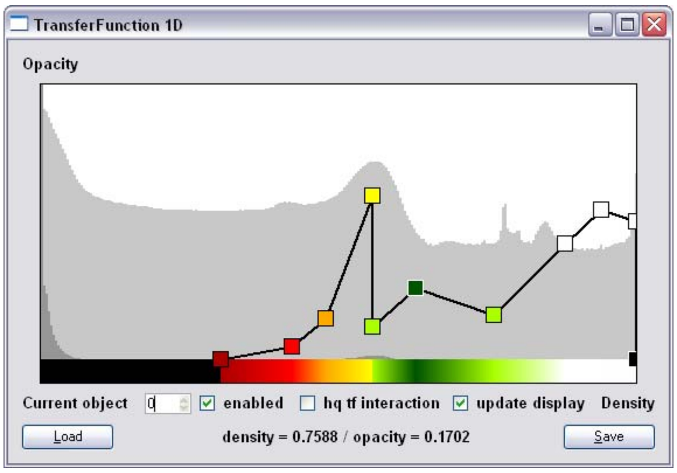
Figure 2.4: The raycasting geometry textures. The length of the vectors is stored in the alpha channel of the direction texture (white background) [Sch05].

The fragment program uses the frontface- and direction-texture to specify sampling points and to step through the volume according to the sampledistance. Along the ray, density values are fetched from the volume with hardware-native trilinear interpolation, mapped to rgbα-values according to the transfer function and finally composited to approximate the volume rendering integral (Equation 2.1).

As an image order approach raycasting benefits hugely from the parallel architecture of modern graphic processing units as every pixel can be computed without any knowledge of other pixels. With ATIs Radeon 19x0-series featuring up to 16 pixel shader processors, with three arithmetic-logical-units each (meaning 16 rays can be cast simultaneously) interactive rendering speeds can be achieved even with more sophisticated extended implementations of the emission absorption model.

2.3 Transfer Functions

As shown in the previous section direct volume rendering is a very powerful way to visualise 3D volume data. However the quality of the final rendering highly depends on the mapping of the available data properties to optical properties in the rendering. This is usually done with a so called *transfer function*. The most obvious data property of course is the density $f(\vec{x})$ of the voxels for each position $\vec{x} \in \mathbb{R}^3$ in the volume. Thus the density was used as the domain of the transfer function in nearly all of the early volume rendering applications defining the transfer function as $T : f(\vec{x}) \rightarrow (r, g, b, \alpha)$. As the domain is part of the one-dimensional set $\mathbb{R}$ the transfer function is called one-dimensional, too. A very common interface to create such a transfer function is shown in Figure 2.5. The x-axis represents the density in the data field. Multiple control-points with a colour value (in rgb) can be created. The y-position of every control point defines the α-value or opacity. Between two adjacent control points colour and opacity are interpolated.



density = 0.7588 / opacity = 0.1702

Figure 2.5: The 1D transfer function panel of the VRVis hardware volume renderer.

To guide the user defining the transfer function a logarithmic histogram is used as background of the panel.

With the density as the domain of the transfer function it is possible to reach a very good rendering quality, but different materials of the same density can not be distinguished. Also the visualisation of material borders which are very important for the depth perception is usually very hard. Multidimensional transfer functions can help to overcome these problems. However, with more dimensions the choices for the domain are bigger and so it s not surprising that in the last years a lot of different approaches for multidimensional transfer functions have been presented. Some of them are presented in this section.

The creation of multidimensional transfer functions needs a more sophisticated user interface than a simple 1D panel. The most common interfaces for two dimensional transfer function interaction are very similar the one presented in [KKH01] (see Figure 2.6) and use the two axes for the two dimensions of the domain, while the opacity is moved from the second axis into widgets, which also hold the colour information. These widgets are used to draw in the 2D domain. The standard histogram is also replaced by a two dimensional histogram with the same axes as the coordinate system, where accumulations of values are indicated by higher intensities in the histogram.

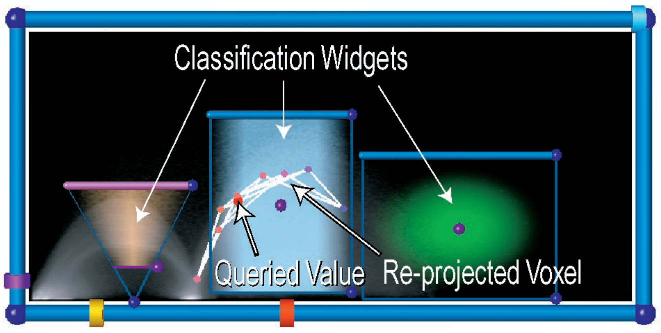


Figure 2.6: The 2D transfer function panel presented in [KKH01].

2.3.1 Density, Gradient Magnitude Transfer Function

The most common approach for two-dimensional transfer functions introduced in [Lev88] uses the gradient magnitude gm as the second dimension of the domain besides the density d:

$$T : (d, gm) \rightarrow (r, g, b, \alpha) \tag{2.3}$$

This is very useful to visualise boundaries of different materials as the gradient magnitude in homogeneous areas approaches zero, while it reaches its maximum on the edge between two materials (see Figure 2.7).

These characteristics lead to histograms which make it often very easy to recognise the features of a volume. Figure 2.8(a) shows such a histogram. In this figure, instead of just adding up the intensity of point accumulations a colour encoding was used where purple represents the lowest quantities and red the highest. Homogeneous areas in the volume usually add up to a big accumulation at the bottom of the histogram. The transitions between two materials sum up to an arch connecting the density of each material at the bottom.

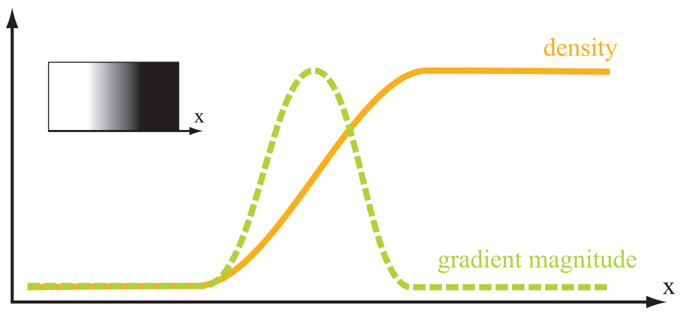


Figure 2.7: The density and gradient magnitude functions of an example 1D cutout from a volume (small image).

2.3.2 Lighting Transfer Functions

In [LM04] the authors propose to use a higher dimensional transfer function to improve the lighting. While keeping the density of the actual voxel the domain is extended by the densities of the neighbouring voxels in positive and negative gradient direction. Although this is actually a 3D transfer function

$$T : (d, d_l, d_h) \rightarrow (r, g, b, \alpha, k_a, k_d, k_s) \tag{2.4}$$

with d_l and d_h the lower and the higher density of the neighbouring voxels in gradient direction and k_a, k_d, k_s the ambient, diffuse and specular constants of the Phong [Pho75] lighting model used for rendering, it can be separated into the standard one dimensional

$$T_1 : d \rightarrow (r, g, b, \alpha) \tag{2.5}$$

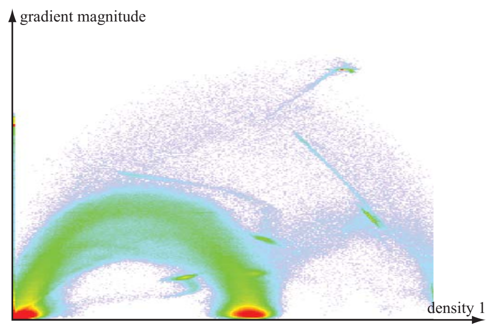
and a two dimensional

$$T_2 : (d_l, d_h) \rightarrow (k_a, k_d, k_s) \tag{2.6}$$

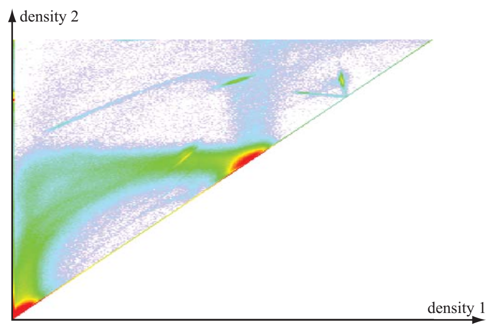
transfer function. This not only reduces the storage capacity required but also makes it easier to create such a transfer function as the already introduced panels for one- and two-dimensional transfer functions can be used in combination.

A typical histogram for the 2D part can be seen in 2.8(b). The empty part in the histogram results from an extension taken from [ŠBSG06]. The axes of the histogram are not the densities in positive and negative gradient direction but the two densities are sorted so that the x-position is always the lower and the y-position the higher value. Now the homogeneous regions in the volume are mapped onto the bisector between the two axes, as the neighbours of the voxels have similar densities. Similarly to the gradient magnitude based histogram these regions in the histogram are connected. Again these connections, here looking more like steps than arches, represent the transitions between materials in the volume.

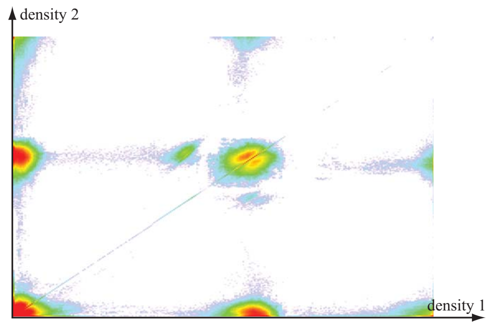
Another modification proposed in [LM04] could be made by using two parallel axes instead of cartesian coordinates and draw a line connecting the axes for each value pair instead of a point.



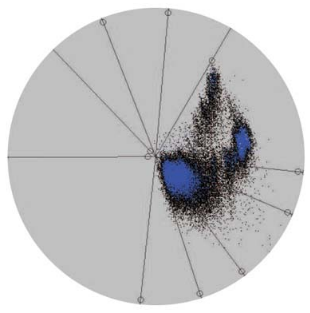
(a) Density / Gradient Magnitude Based Histogram.



(b) One Step LH Histogram.



(c) Mirrored LH Histogram.



(d) Multidimension Histogram in Star Coordinates.

Figure 2.8: Different multidimensional histograms.

2.3.3 Low High (LH) Transfer Functions

The low high (LH) transfer function presented in [ŠBSG06] is an approach very similar to the just introduced lighting transfer functions. It is also based on a low and a high density value, but instead of using these values for the lighting they are used to assign colour and opacity to a voxel.

$$T : (d_l, d_h) \rightarrow (r, g, b, \alpha) \tag{2.7}$$

Also the the low and the high value are not based on simple neighbour fetches but instead the gradient field is followed in both directions until a homogeneous area is reached and the density values of these homogeneous areas are used as low and high values, respectively.

In the histogram this means that the homogeneous areas in the volume are still mapped to blobs on the bisector while the ″steps″ connecting two blobs are collapsed to blobs with the low-value as x- and high-value as y-coordinate. In [ŠBG06] the authors modify this approach by not sorting the coordinate tuple by their value but instead test on which side of the transition the voxel is (if it is not in an homogeneous area). This means that on one side of the bisector in the histogram now the blobs represent the part of the transition belonging to one material while the voxels of the same transition belonging to the other material are mapped onto the other side of the bisector (see Figure 2.8(c)).

Now a boundary can be easily visualised in the rendering by just placing a fitting widget over a single corresponding blob.

2.3.4 Higher Dimensional Transfer Functions

The main problem with higher dimensional transfer functions is to create a user interface that makes it easy for the user to define them. While a 2D transfer function can be created by painting or placing widgets on a plane, a *real* 3D transfer function (which can't be split up into multiple functions like in 2.3.1) with the same user interface approach using cartesian coordinates would require the user to paint in 3D space. Even considering this as doable (though very hard), what if more than three dimensions are required? Such a transfer function for example could include the position in the volume in addition to the density in the domain defining a four-dimensional transfer function, or in multi volume rendering, such as PET-CT, features of all volumes could be taken into account in one transfer function over all volumes. The authors of [BCL$^+$06] used the concept of star coordinates [Kan01] to design a user interface for creating n-dimensional transfer functions by painting on a two dimensional circle.

Mathematically an n-dimensional axis aligned box

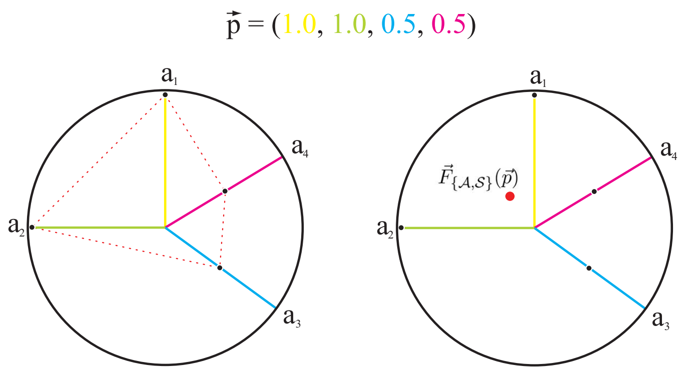
$$\Xi = [min_1, max_1] \times ... \times [min_n, max_n] \in \mathbb{R}^n \tag{2.8}$$

is mapped onto a plane in $\mathbb{R}^2$ by a family of Functions $F_{\{\mathcal{A},\mathcal{S}\}}$, where $\mathcal{A} = \{\vec{a_1}, ..., \vec{a_n}\}$ is a set of n unitary vectors of $\mathbb{R}^2$, $\vec{a_i}$ representing the i-th axis of $\mathbb{R}^n$ and $\mathcal{S} = \{s_1, ..., s_n\}$ a set of n non-negative real numbers, s_i representing a scale factor for $\vec{a_i}$ in the star coordinate system.

A Point $\vec{p} = (x_1, ..., x_n)$ is then mapped into the star coordinate system with

$$F_{\{\mathcal{A},\mathcal{S}\}}(\vec{p}) = \frac{1}{n}\sum_{i=1}^{n} \frac{(x_i - min_i)}{(max_i - min_i)} s_i \vec{a_i} \tag{2.9}$$

In Figure 2.9 the mapping of the point $\vec{p} = (1, 1, 0.5, 0.5) \in \Xi = [0, 2]^4$ in an exemplary star coordinate system is illustrated.



$\vec{p} = (1.0, 1.0, 0.5, 0.5)$
a_1
a_2
a_3
a_4
a_1
a_2
a_3
a_4
$\vec{F}_{\{\mathcal{A},\mathcal{S}\}}(\vec{p})$

Figure 2.9: Star coordinates mapping of a point in $\mathbb{R}^4$.

The actual transfer function is then created in two steps. First the user modifies the coordinate system by dragging the axes in position and defining the scale factor for each axis, so that the interesting parts in the histogram are visible (see 2.8(d)). After that the transfer function is defined by painting on the 2d plane just like in the before mentioned approaches. So mathematically the creation of the transfer function

$$T : \Xi \rightarrow (r, g, b, \alpha) \tag{2.10}$$

is substituted by the composition of the functions

$$G \circ F_{\{\mathcal{A},\mathcal{S}\}} : \Xi \rightarrow (r, g, b, \alpha) \tag{2.11}$$

with G describing the actual mapping from the n-dimensional space to the colour space.

2.4 The Hardware Volume Renderer

The Hardware Volume Renderer (HVR), in development over the past five years at the VRVis Research Center is a highly advanced application for the visualisation of volume data and besides that it also supports completely hardware based segmentation [Kla06].

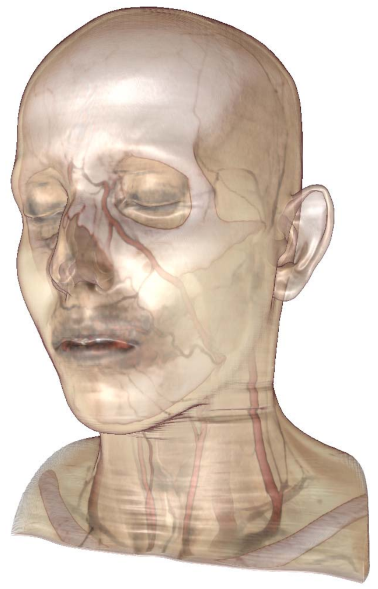
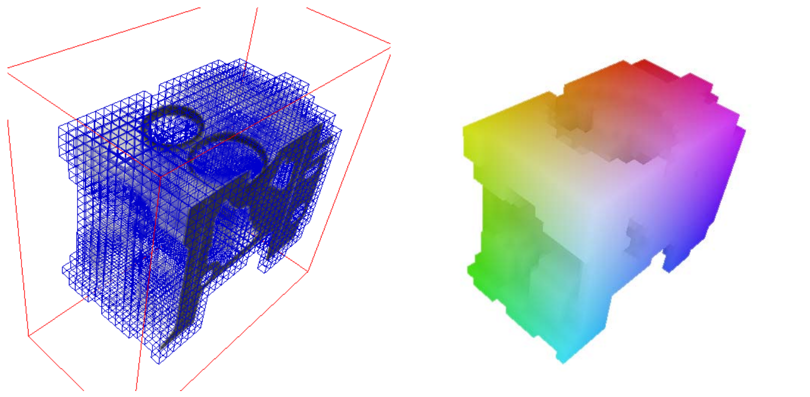


Figure 2.10: Volume rendering of a human head data set of size 512 x 512 x 333. Interactive framerates can be achieved with all modern graphics cards.

This rendering framework provides multiple visualisation models, such as raycasting based isosurface rendering and multiple extensions of the emission-absorption based direct volume rendering. Such are for example shaded modes based on local illumination and also based on global illumination for example with the use of deep shadow maps [HKSB06] or gradient magnitude modulated rendering, which is a simple two dimensional transfer function. It is also possible to use multiple rendering modes in combination to improve the visualisation of different objects [HBH03].



(a) Block structure based on a transfer function. (b) Frontface texture with advanced bounding geometry.

Figure 2.11: The block structure for empty space skipping for the engine data set and the resulting frontface texture.

To achieve interactive rendering speeds a lot of acceleration techniques, e.g., from [KW03] are implemented. One of the most important is empty space skipping and early ray termination based on a more sophisticated bounding geometry [Sch05]. Before passing the volume to the graphics card it is divided into equally sized blocks. Based on the transfer function or the isosurface parameters it is decided for each block whether it will be rendered or not. Using this information a bounding geometry is built. The front- and backface-textures described in Section 2.2 are then created based on this geometry instead of a simple bounding box. Image 2.11(b) shows the frontfaces-texture adapted to the well known engine data set.

This blocking structure can also be used for rendering large data sets. If a block will not be rendered and this is already known before the data set is passed to the GPU, why should it even be passed to the graphics card? Therefore the blocks that will be used are stored in a new texture that will be passed to the graphics board. However, the GPU now needs to know the position each block was at in the first place so a second (much smaller) volume texture is created storing for every original block whether if it is active or not and if it is active where it is stored in the new texture. To reduce memory consumption this block-structure is not the same as the one for the bounding geometry, but the blocks are larger. Usually the size of one of these larger blocks is 32x32x32 voxels, while the smaller blocks for the bounding geometry are 8x8x8 voxels.

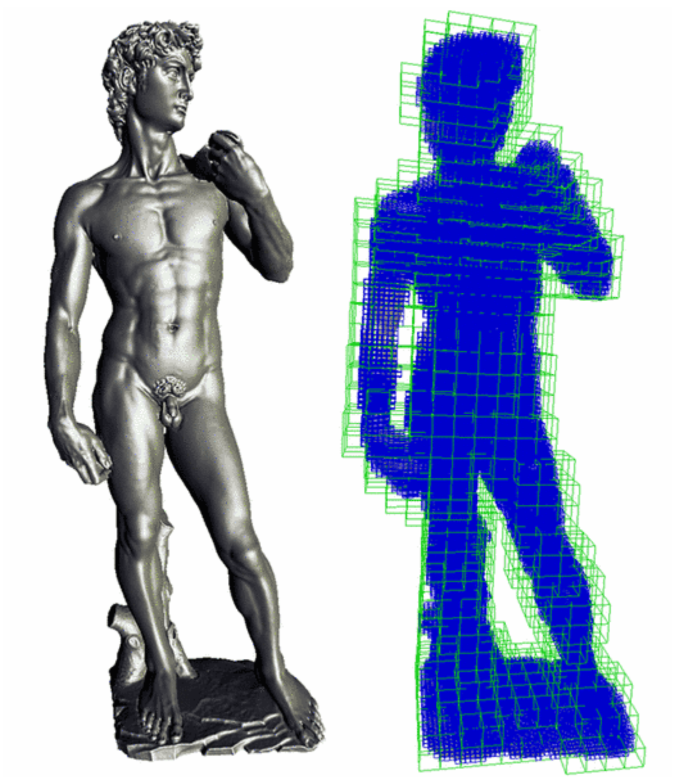


Figure 2.12: Rendering of a 576x352x1536 16-bit distance field, the green blocks show the caching structure, the blue ones the bounding geometry.

The HVR framework has also been integrated into a Virtual Reality environment [KHS$^+$06] and was extended for the use with virtual endoscopy [SHN$^+$06] with the possibility to deform the volume [SBH07].

Chapter 3

Rendering Industrial CT Data

In this chapter the modifications of the hardware volume renderer for industrial CT data are presented. In Section 3.1 the basics for the new two dimensional transfer functions needed for the new render modes are described. While this includes the details of the new panel a complete user-guide for the panel can be found in Appendix A.

In Section 3.2 the extension of the shaded-dvr- and isosurface-render-mode to support correct shading of clipped edges based on [WEE03] will be explained. Notice that this improvement does not rely on any kind of transfer function, though it can be used to improve image quality for all render modes using local illumination and was implemented for all of the 2D transfer function render modes.

The following Sections 3.3, 3.4 and 3.5, cover the newly implemented render modes using the 2D transfer functions. Each of these sections contains two subsections, one describing the necessary preprocessing and the other one the rendering part.

3.1 The 2D Transfer Functions

The first decision when creating a new user interface to build multidimensional transfer function is how many dimensions are needed and through this what kind of coordinate system is needed. As right from the planning stage of this work all render modes that were taken into account use no more than two dependent dimensions in the domain for each transfer function an interface based on cartesian coordinates like the one presented in [KKH01] (see Figure 2.6) was chosen. This is sufficient for all planned render modes and the users of the 1D panel do not need a lot of practice due to the similar setup of both panels.

However the 2D transfer function panel is not a complete new development for the VRVis hardware volume renderer. Some basic user interface components, though completely without functionality, were already present when this work was started. As these were mainly converted from the one dimensional panel the setup of the panels is very similar. As the design decisions that were made, did not conflict with the existing basis it was decided to build upon this work not only to save development time but also because the similarities between the two panels would make the migration for the users from the 1D to the 2D panel easier.

As the Graphical User Interface (GUI) of the existing application was built using QT 3.3 [Tro05] it was a case of Hobson's choice to use this for the implementation of the GUI for the two dimensional transfer functions as well. Although QT 4 comes with a complete new implementation of the Qcanvas drawing engine (Graphics View) [Tro07] which may have solved some issues, none of them were that important as to justify an upgrade of the whole volume renderer application to QT 4 at this time.

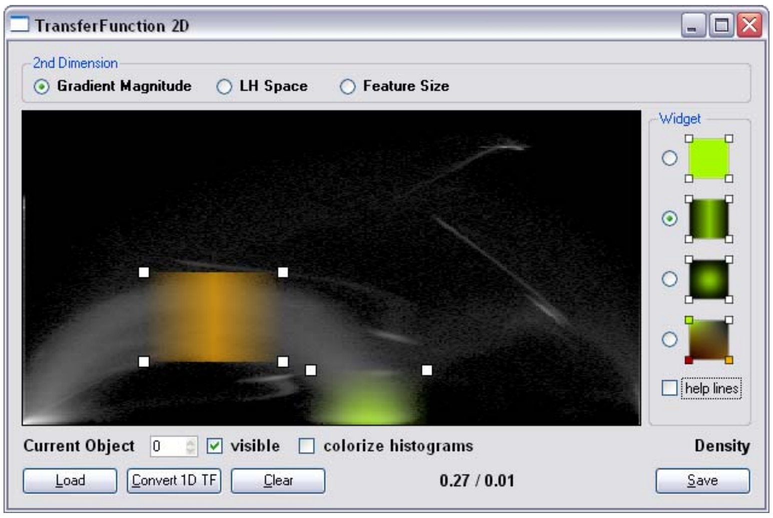


Figure 3.1: The graphical user interface to create 2D transfer functions.

Figure 3.1 shows the final implementation of the 2D transfer function panel. The main part is the 2D drawing canvas. Just as in the 1D panel the background is a pixmap of the histogram (compare Figure 2.5). Multiple histogram types can be displayed guiding the user in creating the transfer function according to the chosen render mode. The standard histogram has the density on the x-axis and the gradient magnitude on the y-axis. The histograms are in logarithmic scale. If needed, a colour encoding can be used instead of the standard grey values (compare Figure 2.8(a)). High accumulations are then in red, lower denominations are indicated by purple colour. To narrow the histogram a cap was used because usually the regions of very low density (mostly air) add up to very large numbers so that the rest is barely visible even in logarithmic scale often. However the histograms are computed (and cached) unscaled and scaling is done in the GUI part so that no data is lost and if needed the whole range can be used.

Where a continuous line was drawn to define the transfer function in the 1D panel here multiple seperate widgets can be placed on the canvas. The widgets can be moved around, their size can be modified by dragging the control points on the vertices, and colour and opacity can be set. In total four different widgets were implemented. They are all derivations of the QCanvasRectangle class. A free polygonal widget was planned but discarded due to problems with the Qcanvas class. Modifications regarding size and position of all these vertices are done the same way by either dragging the body of the item itself to move the widget or by dragging one of the vertices to change the size. The difference between the four widgets is their alpha and colour distribution. Three have one colour for the whole widget and different alpha distributions:

- same alpha for the whole widget
- linear ramps in x-direction from both sides to the middle
- a two dimensional Gauss-blob

The colour and opacity for the fourth widget can be defined differently for every vertex, between the vertices both values are bilinearly interpolated [Mül04, lecture 7, "Scan Konvertierung"]. All colour and opacity assignments are done on click with the getRgba() function of the QColorDialog class.

Just like for the 1D panel transfer functions can be loaded and saved. In addition a conversion of a 1D transfer function is possible (for the 2D modes where the standard density is included in the transfer function domain).

To avoid cryptic names for the different widgets the radio-button-menu to choose the widget on the right-hand side uses graphic icons as identifiers instead of text. A widget is created by choosing the type from this menu and clicking into the canvas. By clicking with the right mouse button on a widget it gets deleted from the list (in the internal representation every item has a pointer to the previous and to the next item making the deletion possible without searching). All widgets can be deleted at once with the clear button.

Of course, just like the 1D transfer functions their 2D equivalents can be saved for later use. While a saved one dimensional transfer function includes the definition of the transfer function as well as the lookup table the new *.tf2d files (which are basically human readable plain text files) only store the definition. A definition of a 2D transfer function consists of multiple ordered nodes, the ordering indicating the z-coordinate.

A node consists of a type indicating the form of the actual widget and its alpha and colour distribution, either one rgbα colour for the complete node or one colour for each vertex (if the type allows that) and finally four position coordinates the first two defining the position of the upper left vertex of the bounding box, the last two the size of the bounding box. Listing 3.1 shows an example *.tf2d file.

Listing 3.1: A 2D Transfer Function File.

```
transfer function definition_2D:

node: type: 2 1 color: 255 255 255 255
      position: 72.843307 20.256882 145.504272 143.889908
node: type: 2 3 color: 170 0 0 25 0 170 0 0 200 0 200 0 75 0 50 150
      position: 1.094017 83.963310 80.592590 142.972488
node: type: 2 1 color: 255 170 127 25
      position: 0.000000 44.036705 78.404556 71.045876

end TFD
```

Every action in the panel triggers a sync of the GUI-side transfer function definition to the definition on the render-side. Here the actual lookup table is created by filling a 256 times 256 times 4 byte-array with zeros and then going through the item list from bottom to top and setting the rgbα value with standard back to front alpha blending [EHK$^+$04, section 1.2.4]:

$$C_i' = C_i + (1 - A_i)C_{i+1}' \tag{3.1}$$

The array is then converted into a two dimensional texture and loaded onto the graphics board.

3.2 Illuminated Clipping Planes

Basic illumination in Direct Volume Rendering is usually done locally, which means only the orientation of the surface, the position of a point light source and some material properties are taken into account to compute the intensity [EHK+04, p. 66 – 71].The most popular model based on this concept is the Phong model [Pho75]. Lighting is evaluated as the sum of an *ambient*, a *diffuse* and a *specular* term:

$$I_{Phong} = I_{ambient} + I_{diffuse} + I_{specular}. \tag{3.2}$$

While the *ambient* term

$$I_{ambient} = k_a \tag{3.3}$$

is constant, the *diffuse* term

$$I_{diffuse} = I_p k_d \cos\phi \tag{3.4}$$

represents the light which is reflected equally in all directions, as well as the *specular* term

$$I_{specular} = I_p k_s \cos^n \alpha \tag{3.5}$$

which represents highlights on shiny surfaces, besides the intensity emitted by the lightsource I_p and the constants k_d and k_s depend on angles ϕ respectively α which both incorporate the surface normal $\vec{n}$. The surface normal of course is not present in Direct Volume Rendering, yet it can be substituted in most cases by the normalised gradient which is the first order derivative of the scalar field.

However, in one case the gradient is completely unusable. In homogeneous regions the gradient is not well defined as differences between neighbouring voxels are to small or actually zero. This is usually no problem as the goal is to light surfaces but when clipping into homogeneous material the clipping plane and with it the homogeneous area becomes the surface and so the lighting on the surface is not well defined (see Figure 3.2(a)). On such a surface it is impossible to differentiate actual indentations from the rest of the surrounding area.

This problem can be solved by using the normal of the clipping plane instead of the gradient in areas where the first hit of the raycasting ray is

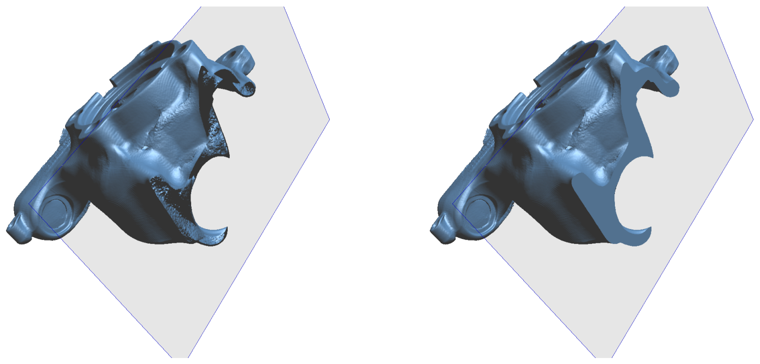

(a) Clipping Plane Illumination Off. (b) Clipping Plane Illumination On.

Figure 3.2: Clipping into homogeneous areas with isosurface rendering.

on the clipping plane as described in [WEE03].

The VRVis hardware volume renderer uses a very basic box clipping approach. Two planes for each dimension, orthogonal to the axis, define a cuboid. The inside of this box is rendered, while the outside is clipped. Such a cuboid can be described by just the minimum and the maximum value for each dimension defining two points in $\mathbb{R}^3$. The clipping is done on the bounding geometry and thus in the OpenGL part of the hardware volume renderer to increase performance. Therefore the existing shader programs must be extended by the coordinates of the two points describing the clipping box. This is done by simply passing two *uniform vec3* variables, each holding the coordinates of one point to the shader.

The shader program itself only has to do two additional steps. The first time the raycasting ray hits a samplepoint which actually has to be rendered (either determined by the isovalue in isosurface rendering or the transfer function in DVR mode) it has to be checked if this samplepoint lies on a clipping plane. If not, the gradient computation can be done as normal for this and all following samples on the ray. If the voxel actually is on a clipping plane, the associate clipping plane normal must be computed instead of the gradient for the first sample point. As shown in lines 14 – 18 of Listing 3.2 these computations can be done very efficiently.

Listing 3.2: Part of the GLSL fragment program for isosurface rendering with illuminated clipping planes.

```
// GLSL fragment shader
// raycast3_fragment_program_RC_FH_CLIP.glsl

...

uniform vec3 clip_min;  // Point with minimum value of each direction
uniform vec3 clip_max;  // Point with maximum value of each direction

void main()
{
        // ... prerequisites and cast till the first hit

        // check if a clipping plane was hit
        vec3 minfaces = 1.0 + sign( clip_min - samplepos.xyz );
        vec3 maxfaces = 1.0 + sign( samplepos.xyz - clip_max );

        // compute the surface normal (eventually normalise later)
        vec3 gradient = maxfaces - minfaces;

        // bisection and gradient computation only if not on clipping plane
        if ( all ( equal ( gradient, const_zero.xyz ) ) ) {

                // ... standard normal computation
        }

        // normalise gradient (either clipping plane or volume gradient)
        gradient = normalize( gradient );

        // ... perform lighting and output color
}
```

The vectors `minfaces` and `maxfaces` essentially are binary vectors, holding one if the corresponding clipping plane was hit for each dimension. Because the only time both clipping planes of one dimension are hit at the same time there wont be any rendering anyway the difference-vector of these two vectors holds all information needed to determine if a clipping plane was hit. It will be `( 0, 0, 0 )` if no plane was hit and some combination of `1` and `−1` otherwise. Even better, due to the use of the `sign()` operator and the addition of one to the original vectors, the difference is also the surface normal if one surface was hit (if the edge of two or all three was it still must be normalised).

While the performance has not changed noticeably in comparison to the unmodified shader programs the gain in image quality is huge (compare

Figure 3.2(a) and 3.2(b)).

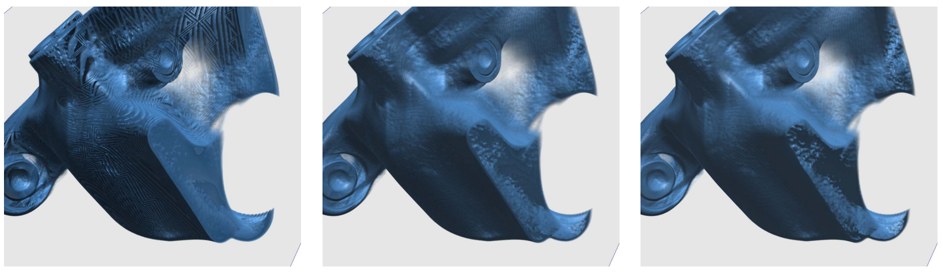

(a) 0.3x Sampling. (b) 1.0x Sampling. (c) 4.0x Sampling.

Figure 3.3: Clipping into homogeneous areas with shaded DVR, only shading of first hit modified.

Whereas this approach works really well with the isosurface rendering in the DVR mode the sampling rate influences the rendering quality. With a higher sampling rate the influence on the rendering of a single sample (and so of the first sample with the modified lighting) gets obviously less as more samples are traversed by the ray, making the ”skin” with the modified lighting less visible. In areas of translucent material the samples with unmodified lighting behind the ”skin” shine through (see Figure 3.3). Figure 3.4 shows a modified approach: an epsilon (which can be changed interactively by the user) defining the thickness of a layer behind the actual clipping plane was introduced. All samples inside this layer are shaded with the clipping plane normal instead of their respective gradient. The clipping plane is clearly recognisable as a surface here.

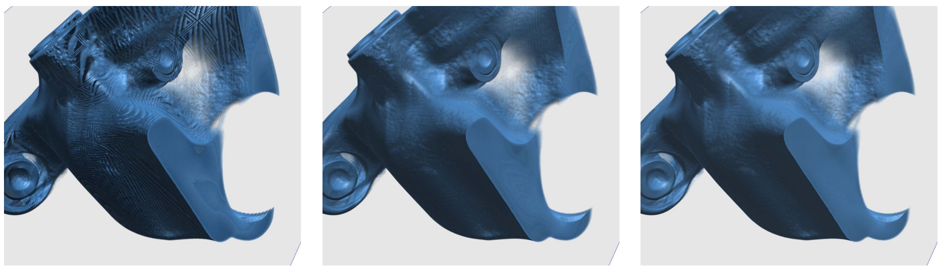

(a) 0.3x Sampling. (b) 1.0x Sampling. (c) 4.0x Sampling.

Figure 3.4: Clipping into homogeneous areas with shaded DVR and a thicker layer with modified shading.

3.3 Density / Gradient Magnitude-Based Render Mode

As shown in Figure 2.7 the gradient magnitude is an excellent property to identify material transitions. Therefore a two dimensional transfer function classifying a sample based on its density and gradient magnitude should be a very good choice to visualise material boundaries alongside homogeneous areas.

There is no need for any preprocessing for the density/gradient magnitude based rendering. The gradient magnitude can be computed on the fly in the fragment shader of modern graphics cards without losing interactivity as it is already done in the shaded render modes. However to ease the task of creating a suitable transfer function a histogram is created and for performance reasons the block cache must be adapted to the 2D transfer function. This is described in Subsection 3.3.1. The fragment program used for rendering is then presented in Subesction 3.3.2.

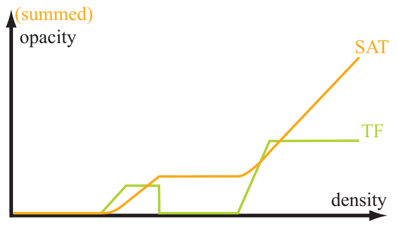
3.3.1 Preprocessing

After the volume is loaded first the histogram is created. To minimise memory consumption the histogram is computed directly from the volume data, meaning there is no intermediate step where a gradient-map for the whole volume is computed. To scale the histogram to the maximum gradient magnitude there must be two passes through the whole volume. The first pass only computes the maximum gradient magnitude, in the second pass the gradient magnitude is computed for every voxel, scaled with maximum gradient magnitude and then the histogram entry for the (density, gradient magnitude) tuple for this voxel is increased. This approach is necessary because some of the very large data sets make it impossible to create a complete gradient magnitude volume. A short preprocessing time is acceptable as it has to be done only once when the volume is loaded the first time and after that the histogram is stored in a cache file with the volume and can be loaded anytime later on with the volume.

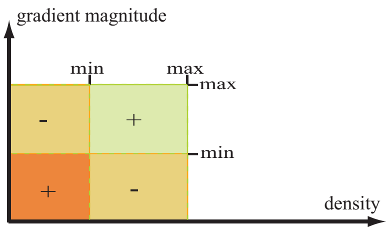
The gradient magnitude is computed on the CPU with central differences to lower computation time, as the histogram is of course only a guide to create the transfer function and the rendering quality is not directly influenced by the histogram quality. However, if needed a custom 3x3x3 filter stencil is implemented and can be used for example with a sobel operator or 4D linear regression.

The culling algorithm of the hardware volume renderer described in Section 2.4 for the one dimensional transfer function uses the minimum and maximum density of every block to decide whether a block has to be rendered or not. A (1D) summed area table [Cro84] is generated from the opacity values of the transfer function table. The summed area table is simply a table storing for each position *i* the sum of the opacity in position *i* and all its predecessors in the transfer function table. If the difference of the lookup of the minimum density of a block and the maximum density of the same block in the summed area table yields zero, the block can be discarded for rendering.

To adapt this concept to the two dimensional transfer function first minimum and maximum gradient magnitude values for each block have to be stored. This can be easily done while creating the histogram. The block structure is known at this point, so all that has to be done is to check which block the actual voxel belongs to and then check if the just computed gradient magnitude is a minimum or a maximum for this block. In the second step the summed area lookup table has to be extended to the second dimension. Here for every position (x, y) the summed area value is the sum of all values in the rectangle defined by the origin and the position (x, y). The decision if a block is needed or not with the minimum and maximum values can than be done with four lookups in the summed area table, one for each combination. The sum that has to be computed is then $f(max, max) - f(min, max) - f(max, min) + f(min, min)$ (Figure 3.5(b)).



(a) Transfer function (TF) and summed area table (SAT).



(b) 2D summed area table lookup.

Figure 3.5: (a) Opacity of a 1D transfer function (green) and the corresponding 1D summed area table (different opacity scale). (b) A 2D summed area table lookup. The "+" marked areas are added, the "-" marked substracted. Note, the lower right corner of all areas is the origin.

3.3.2 Rendering

As the second dimension (gradient magnitude) for the transfer function texture lookup is not precomputed it has to be computed on the fly in the fragment program. In fact this is already done in all render modes with local illumination, like the isosurface and shaded DVR modes. There the gradient is computed for lighting purposes. However this computation is done in the normal modes after the classification to improve rendering speeds, as samples which were classified with a very low opacity do not have to be shaded. Here obviously the computation has to be done prior to the classification. After that, additionally the length of the gradient has to be computed, as this is not needed for lighting and finally the length has to be scaled with the factor already used when creating the histogram to match with the transfer function.

The additional information needed in the fragment program is the scaling factor and in the non shaded render mode the voxel size is needed to compute the gradient. Both are present in the framework and can be passed to the shader program with two uniform variables. The gradient computation is done with central differences just like in the preprocessing, but with subvoxel precision with hardware native trilinear interpolation. This of course means that the histogram does not exactly represent what will actually be rendered, but the differences are not noticeable in the rendering.

After the computation the classification can be done using the density and gradient magnitude. Here the only difference to the 1D transfer function is a 2D texture lookup instead of 1D.
After that the shader is basically the same as in the render modes using a 1D transfer function.

Listing 3.3: 2D texture lookup in GLSL.

```glsl
// vector holding density and gradient magnitude
vec2 classifier = vec2( tex_density , gradientmagnitude );

//classification with the density/gradient magnitude vector
vec4 col_classified = texture2D( tf_texture_2d , classifier );
```

3.4 Low/High Value-Based Render Mode

The low/high render mode presented in [ŠBSG06] uses two density values to classify a sample. It is especially suited for the visualisation of material boundaries.

In an ideal volume without any noise or bias, consisting of two materials, A and B, all voxels belonging to material A would have the same density and so would all voxels belonging to material B. The gradient magnitude would take on a value greater zero only in those areas where two voxels of the two different materials are neighbours.
Low/high based classification exploits this in the following way. For every sample point inside a material the low and the high value both are the density of the material. If the sample belongs to a boundary the densities of the two materials sharing this boundary are taken as low and high value respectively. In the original approach these values were saved in order of the density value, thus the name low/high. In a more sophisticated approach [ŠBG06] which is presented here the material the sample belongs to is always the first value in the tuple.

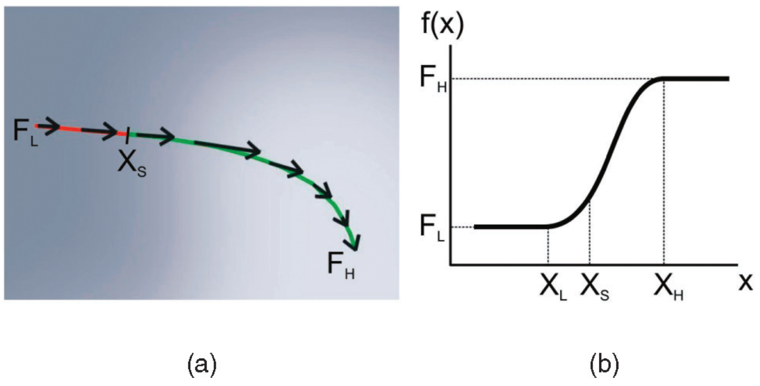
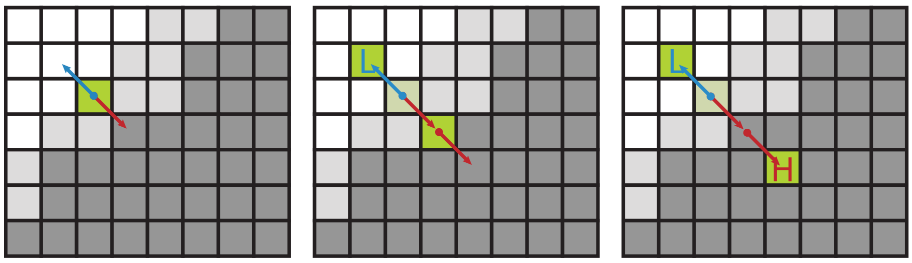


Figure 3.6: (a) Starting at position X_S, the path across the boundary by integrating the gradient field is generated. The green part of the path in the gradient direction leads to F_H. The red part generated in the opposite direction ends by finding F_L. (b) The intensity profile along the path. [ŠBSG06]

3.4.1 Preprocessing

To find the materials sharing a boundary in real world data sets, however, is a time consuming process, as a typical boundary usually consists of multiple voxels and multiple iterations have to be done until a homogeneous region is reached. Thus live computation of low/high values in the fragment shader is very slow with current graphics hardware. An implementation for testing purposes resulted in about three frames per second for a small 256x256x256 voxel data set. Therefore in the preprocessing step both values are precomputed for every voxel on the CPU and a multichannel (luminance-alpha) volume consisting of low and high values is passed to the fragment program.



(a) Datafield, starting voxel (marked green) with gradient (red).

(b) First iteration in both directions.

(c) Low and High value found.

Figure 3.7: The algorithm to compute Low High values for a voxel.

The procedure for the algorithm to build the low and high density volumes is shown in Figure 3.7. In a loop over the whole volume, for every voxel first the gradient is computed. For better performance this is done with central differences, meaning only the six direct neighbours for every voxel are taken in account. Then it is decided, whether the voxel belongs to a boundary or to a homogeneous area, based on the gradient magnitude. If the latter is the case, the algorithm immediately terminates using the current voxel density for both low and high value. If the voxel belongs to a boundary a step of the size of one voxel is made in the positive and the negative gradient direction. In reality this means that in most cases the discrete voxel-field is left here and some kind of interpolation must be done. In the illustration, a simple nearest neighbour approximation is used. However in the implementation the real subvoxel position is used and ev-

ery following step uses trilinear interpolation to compute the gradient and its successor position. This is relatively expensive to compute on the CPU. Now for the two positions again it is checked whether they belong to a homogeneous area or not. If not, another step (this time in direction of the gradient at the new position) is done for both samples. In contrast to the initial voxel only one step for each sample is taken. From the sample found in negative gradient direction it is stepped further in negative direction and vice versa for the sample found in positive gradient direction from the original voxel.
This is repeated until a homogeneous area is found in each direction. To speed up the process the algorithm also terminates after a maximum number of steps or of course when the next step would be outside of the volume. In both cases the values of the two positions with the lowest gradient magnitude on the path are used as low and high values.

Within the algorithm the histogram can be tracked making a second pass over the low/high volume unnecessary. Both the low/high volume as well as the histogram are then just like the density/gradient magnitude histogram written to the the harddisk making this procedure unnecessary the next time the volume is loaded.

3.4.2 Rendering

Rendering using pre-computed low/high values is straight forward. The only difference from render modes using one dimensional transfer functions is the transfer function lookup. The values for the lookup are fetched from the low/high volume instead of the density volume and the transfer function lookup itself again is a 2D lookup. In theory the low/high transfer function can be used for any of the DVR render modes. Practically some modulation to the values has to be done.
The blobs in the histogram carry no information besides the material/boundary information. The position inside the blobs is more or less random and one can not draw conclusions about the position of the voxel by the position in the blob, for example if it is in the centre or the edge of a boundary. De facto every boundary or homogeneous region can only be selected as an entire block.
A good modulation can be made with the gradient magnitude, by emphasising samples with a high gradient magnitude and reducing opacity for

samples with lower gradient magnitude. Therefore the original density volume is passed to the graphics board in addition to the low/high volume. Furthermore shading can also be done based on the gradient of the density volume.

3.5 Feature Size-Based Rendering

The idea of the feature size-based mode is to use the size of a homogeneous area a voxel belongs to as the second property for the transfer function. This would make it possible to easily classify small pores in bigger homogeneous areas, which is one of the main tasks for non destructive testing.

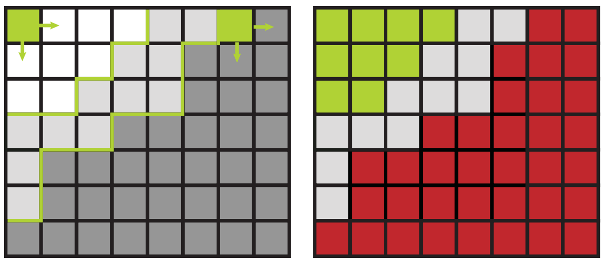
3.5.1 Preprocessing

Again, just like for the low/high mode live computation of the feature size can not be done at acceptable speeds on the graphics board. Therefore a second volume holding this information is built in the preprocessing step. The algorithm that was implemented uses region growing [AB94] to classify the features. As the main focus is to visualise pores a certain size is specified as a threshold and all areas bigger than this size are just marked *too big*. On the other hand all regions smaller than a certain threshold, which would be scaled to zero in the histogram are immediately set to zero, meaning *untagged*, as it can be assumed that these regions can mainly be attributed to noise in the density volume. To safely tag all important areas, a two pass approach is needed.

The algorithm uses the block structure of the volume to discard unused blocks and tag them as *too big* in a first step based on the minimum and maximum density values and the maximum gradient magnitude value, appended to the block structure. If the difference of maximum and minimum density or the maximum gradient magnitude are too low the block is discarded and all voxels are marked *too big*.

The first pass of the algorithm then goes over all remaining voxels and if a voxel is not already tagged and in a homogeneous area it is set as seedpoint (see Figure 3.8(a)). Then the region is grown based on density of the next voxel compared with the mean of all voxels already belonging to the region. After the first pass is done all voxels in homogeneous regions should be marked either with the size of the region or with the *too big* tag. All voxels left should belong to a boundary (grey squares in Figure 3.8(b)).

In the second pass, the border areas are grown. This is needed mainly because a lot of pores are so small that they lack a homogeneous core and thus are not identiefied in the first pass. All voxels with a gradient magnitude above a certain threshold are used as seed voxels (as long as they have

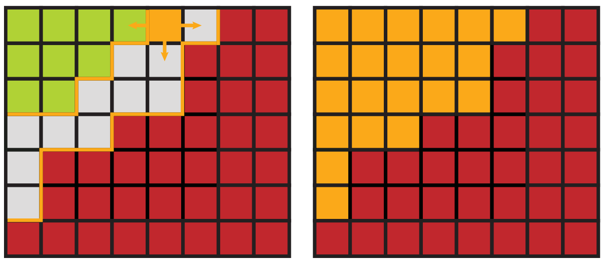


(a) Datafield, two example voxels. The borders of the area they will grow marked in green.

(b) Tagged grown areas. Red means too big, green means pore.

Figure 3.8: First pass of the feature size computation, growing homogeneous areas.

not been tagged in this pass before, the results of the first pass however are not taken into account for the seed voxel choice). The region is then grown on the basis of the gradient magnitude. All neighbouring voxels with a gradient magnitude above the same threshold are put into the region.
If this region grows into a homogeneous region which is tagged with a size (and not with the *too big* marker) it can safely be assumed that the bounding area just grown and and the homogeneous area together build a pore. The homogeneous area thus is flood-filled and border voxels are taken into account for further growing (Figure 3.9(a), 3.9(b)). If the size of the homogeneous core and the area together is larger than the maximum pore size the complete area will be marked with the *too big* tag.



(a) Example voxel and the area it would grow marked with orange borders.

(b) The final areas grown. Orange marking a tagged pore, red again an area too big.

Figure 3.9: Second pass of the feature size computation, growing material borders.

The histogram can not be built during the feature size volume generation and another pass over the two volumes is needed. Again the histogram as well as the feature size volume are stored on the harddisk for later use.

3.5.2 Rendering

As all the voxels tagged with the *too big* marker map to the same line in the histogram despite representing the majority of the voxels in the volume it would be very hard to create a suitable transfer function for the main part of the volume only with the 2D panel. So we decided to use the two dimensional transfer function only to classify the pores and use an additional one dimensional transfer function to classify all untagged voxels and voxels tagged with the *too big* tag.
Thus the feature size render mode uses both the one and two dimensional transfer functions in combination. First a lookup in the feature size channel of the volume yields the size of the area the current sample belongs to. If it is too big to be a pore the standard rendering based on the 1D transfer function is used. Depending on the render mode shading is applied. Thus without the 2D transfer function actually a rendering very similar to the standard render modes using the 1D transfer function would result. If the sample is tagged as part of a pore the density is looked up and with the density and the feature size a lookup in the 2D transfer function table is done. Again of course depending on the mode, shading is applied.
Note that interpolation between the feature size values makes no sense as every pore has only one feature size value and interpolation would only lead to the boundaries of the pores being classified completely wrong. Therefore the feature size volume is passed to the fragment program with the GL_NEAREST option. On the other hand this will make the rendering of the pores very blocky, so again the implemented fragment shaders all use the gradient magnitude based modulation also used for the low/high based rendering.
Usually a very simple 1D transfer function with an opacity gradient covering the regions of interest and only one or two colours is adequate to render the main part. Of course a more sophisticated 1D transfer function will lead to more impressive renderings. The 2D transfer function can also be created in a very simple way. For example a single widget covering the whole domain of the 2D transfer function would classify all pores. In

the histogram every line represents pores of a particular size (or pores of a range of sizes). So pores of different sizes can very easily be classified by placing rectangular boxes over the horizontal lines in the histogram.

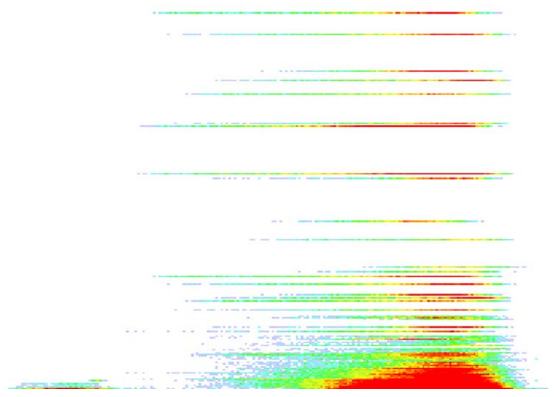


Figure 3.10: A typical feature size-based histogram. On the bottom the lines accumulate, as a lot of small pores are in the volume, single lines, each representing one larger pore are clearly visible in the upper two thirds of the histogram.

Chapter 4

Results

In this chapter the results of this work are presented. Section 4.1 gives a brief overview of the performance that can be achieved with the implemented render modes. In Section 4.2 example renderings for all render modes are presented and if applicable compared to the render modes they should replace.

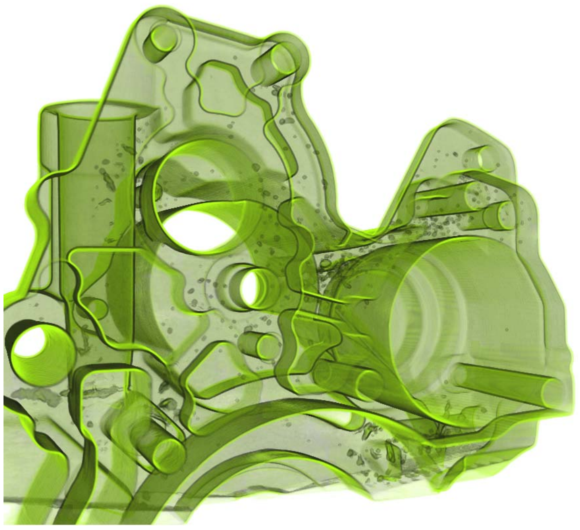


Figure 4.1: Rendering of a cast housing(667x465x512 voxels), with DVR and a two dimensional transfer function based on density/gradient magnitude. Due to the high transparency the frame rates are low, but still interactive at 3-5 frames/s on an ATI Radeon 1900XTX.

4.1 Performance

The following tables give a brief overview of the performance which can be achieved with the implemented render modes. The system that was used to do the performance tests has the following specifications:

- AMD Athlon 64 3800+, Single Core
- 2GB RAM
- ATI Radeon 1900XTX with 512MB graphics RAM
- MS Windows XP Pro SP2.

If not mentioned otherwise the image size was 512 times 512 pixels with a sampling rate of one sample per voxel.

The details of the data sets used in this chapter are as follows:

Dataset	Dimensions [voxel]	density depth [bit]	Size [MB]
Hinge Bearing	300 x 450 x 512	16	135
Weld Seam	300 x 600 x 900	16	379
Cast Housing	667 x 465 x 512	14	310

Table 4.1: Details of the used data sets.

Note that results for the different modes are not directly comparable as the transfer function itself has a heavy influence on the performance as optimisation methods like early ray termination and empty space skipping are highly dependent on the transfer function.

Illuminated Clipping Planes

At first sight the results for rendering with illuminated clipping planes (especially with isosurface rendering) may seem a bit surprising as the more complex shader yields a better performance than the standard isosurface rendering. At second glance this fact is logic as the expensive gradient computation can often be skipped and replaced with the very simple normal computation and the single *if*-branch needed for the decision whether to compute the normal or the gradient does not seem to have that big of an effect as one might expect (the dynamic branching of the ATI card seems quite efficient here).

Clipping Plane	Isosurface [fps]	DVR 1 [fps]	DVR 2 [fps]
Unshaded	34	16	8.5
Shaded	38	15	8

Table 4.2: Performance of different render modes with and without clipping plane illumination.

The test was done with the cast housing data set. Half of the volume (in y-direction) was clipped, so the effectively rendered volume measured about 80 Million voxels. The image size was increased to 1000 squared. For the DVR render mode a (1D) transfer function with medium overall opacity (DVR 1) and one with very low opacity (DVR 2) leading to an highly transparent rendering (like the one in Figure 4.1) were used. Due to virtually nonexistent early-ray-termination in the latter case the framerates of course drop significantly.

Density / Gradient Magnitude-Based Render Mode

Results for the rendering with the density / gradient magnitude based transfer function are pretty straightforward. The significant drop in frames when rendering with unshaded DVR and the 2D transfer function (DVR 2DTF GM) compared to the unshaded DVR with a one dimensional transfer function clearly is caused by the additional computation of gradients which has to be done in the 2D mode to perform the lookup in the transfer function table. This becomes even more apparent when comparing the shaded DVR modes. Here the gradient has to be computed for every sample for all three render modes.

The difference between the standard shaded DVR modes with one and two dimensional transfer functions can be explained by the fact that with the one dimensional transfer function the gradient can be computed after the transfer function lookup and thus can be skipped for voxels with too low opacity.

The gradient magnitude modulated shaded DVR (DVR sh 1DTF GM - modulated) is an approach using the gradient magnitude to weight the influence in the compositing, meaning that samples with a higher gradient magnitude are more important and thus rendered more opaque than samples with a lower gradient magnitude. This is an intermediate step between 1D density based TF DVR and 2D density/gradient magnitude based TF DVR, highlighting boundaries but giving the user no room to influence the modulation. Here the gradient magnitude also has to be computed for every sample point just as with the two dimensional transfer function. However due to the modulation the rendering results differ quite a bit and thus differences in the framerates can be explained.

Render mode	Cast Housing [fps]	Hinge Bearing [fps]
DVR 1DTF	37	30
DVR 2DTF GM	12	10
DVR sh 1DTF	18	15
DVR sh 2DTF GM	11	10
DVR sh 1DTF GM-modulated	17	11

Table 4.3: Rendering performance with the density/gradient magnitude-based 2D transfer function compared to rendering with 1D transfer function.

The tests were done with the cast housing and the hinge bearing data sets. The 2D transfer function was converted from the one dimensional one, meaning rendering with both transfer functions was identical. However in real world applications a more sophisticated 2D transfer function (even in the same density region) will often lead to culling away larger parts of the volume and thus to better performance.

Low/High Value-Based Render Mode

As described in Section 3.4 the rendering with the low/high based transfer function is virtually the same as with a 1D transfer function and gradient magnitude modulation. The only difference is that two volume lookups are necessary instead of one and the transfer function lookup is two dimensional.
Because of the fact, that the original density is not part of the domain of the low/high based transfer function, it is not possible to convert one dimensional transfer functions. Thus a direct comparison between 1D and 2D modes with the same rendering results is not possible and the two transfer functions need to be adjusted manually to produce similar renderings.

Render mode	Weld Seam [fps]	Hinge Bearing [fps]
DVR 1DTF	32.5	67
DVR 2DTF LH	18.3	15.6
DVR sh 1DTF GM-modulated	6.9	10
DVR sh 2DTF LH	6.6	12

Table 4.4: Rendering performance with the low/high value-based 2D transfer function compared to rendering with 1D transfer function (measured with a nVidia GeForce 8800).

Feature Size-Based Rendering

Just like the low/high transfer function based rendering the performance of the rendering with the feature size transfer function should not suffer a significant drawback compared to the rendering with the 1D transfer function. The performance test confirms this with one exception. Framerates when rendering the cast housing with unshaded DVR dropped by one third. The fragment program for unshaded DVR is very simple and it seems that the additional if clause to test which transfer function to use seems to take heavy effect under certain circumstances. Shaded and unshaded modes are not directly comparable, as the unshaded mode lacks the gradient magnitude modulation, which often leads to highly transparent renderings, which lack early ray termination optimisation.

Render mode	Cast Housing [fps]	Hinge Bearing [fps]
DVR 1DTF	35	18
DVR 2DTF FS	22	17
DVR sh 1DTF GM-modulated	7	15
DVR sh 2DTF FS	6	15

Table 4.5: Rendering performance with the feature size-based 2D transfer function compared to rendering with 1D transfer function.

The preprocessing for the feature size volume takes quite some time, but as the volume is cached and the volume generation is done only once a short preprocessing time is acceptable. Preprocessing times for the cast housing (and a cutout measuring 200x200x100, produced for testing purposes) are as follows. The list for the potential voxels for the region can grow very large when enlarging the maximum pore size making the the memory access quite slow and thus leading to the performance loss despite the fact that independent of the size always the same amount of voxels are touched.

Max pore size [voxel]	3072	10240
time (whole volume) [s]	437	928
time (cutout) [s]	3	8

Table 4.6: Preprocessing time for feature size volume creation with different maximum inclusion sizes.

4.2 Advanced Rendering

Illuminated Clipping Planes

Figures 4.2, 4.3 and 4.4 show the isosurface and shaded DVR (1D TF) rendering modes with shaded clipping planes in comparison with their counterpart with lighting on the clipping planes based on standard gradient computation.

Figure 4.2 shows a rendering of a hinge bearing with isosurface rendering. In Figure 4.2(a) there is no defined surface visible. The lighting on the clipped surface is completely scrambled and the rendering on the clipping gives no usable information. Figure 4.2(b) shows the same rendering with clipping plane illumination turned on. The surface is clearly visible and it can easily be seen that there are no pores of material with a density lower than the chosen isosurface value in this region.

In Figures 4.3 and 4.4, showing a weld seam and a part of a cast housing the standard clipping planes are again not useful for any kind of material analysis. Even worse in Figure 4.3(a) the circle artefacts around the spin axis of the item in the CT-scanner are strengthened. On the other hand the indentation on the surface classified with zero opacity in Figure 4.3 and in the colour-range from yellow to green in Figure 4.4 can be easily distinguished from the rest of the surface in the images of the advanced render mode with clipping plane illumination. Even the scanning artefacts in Figure 4.3 are a little smoothened.

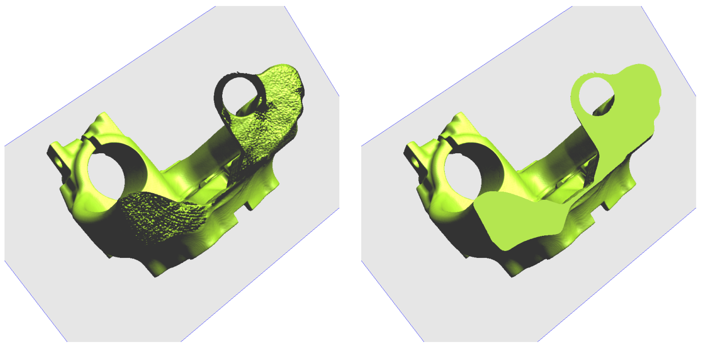
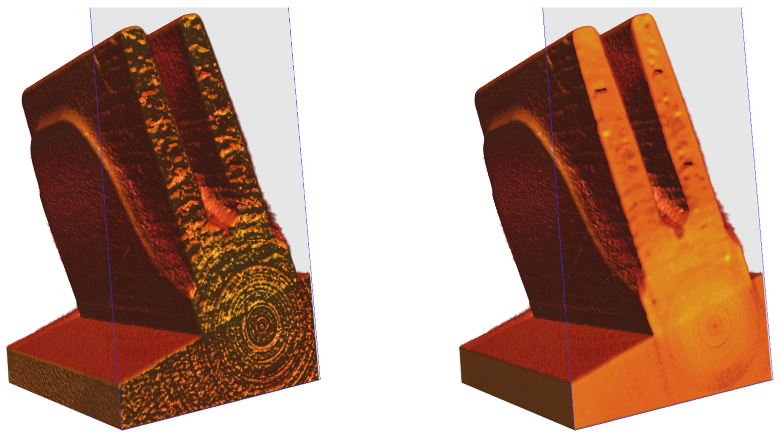
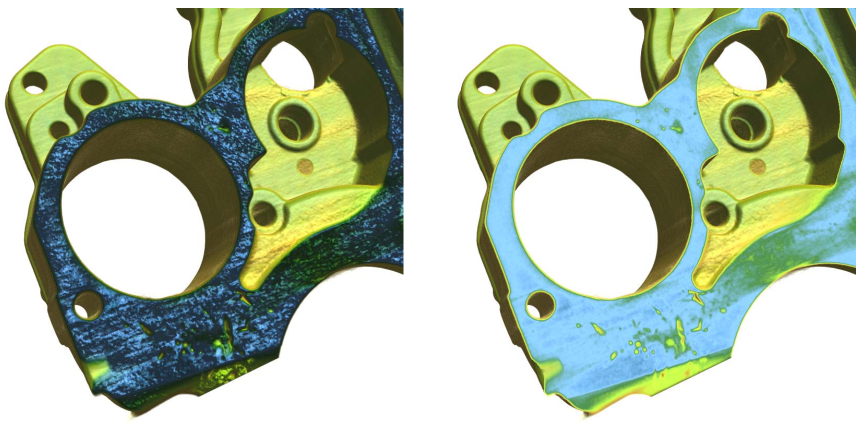

(a) Unshaded clip plane. (b) shaded clip plane.

Figure 4.2: Isosurface rendering of a hinge bearing. Clipping plane at around 70% in z-direction. Isosurface at density 0.32 (in [0..1] range).



(a) Unshaded clip plane. (b) shaded clip plane.

Figure 4.3: Shaded DVR rendering (1D transfer function) of a weld seam. Clipping plane at about half the length of the z extent.

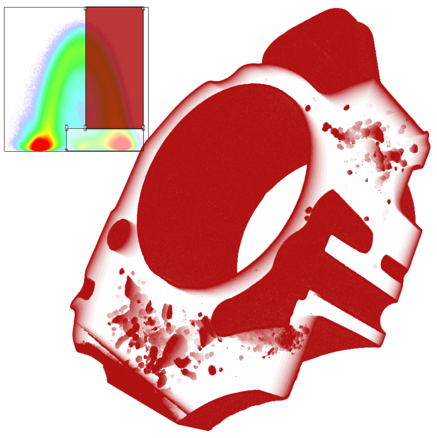


(a) Unshaded clip plane. (b) shaded clip plane.

Figure 4.4: Shaded DVR rendering (1D transfer function) of a cast housing. Clipping plane in the middle of the y extent.

Density / Gradient Magnitude-Based Render Mode

Figure 4.5 shows DVR renderings with a 2D transfer function based on density and gradient magnitude. Figure 4.5(a) incorporates unshaded DVR, Figure 4.5(b) the shaded equivalent. The transfer function in the upper right of each image shows that the white as well as the red widgets in the rendering mostly share the same density region (x-axis of the transfer function) but differ in the gradient magnitude range. The red widget covers the side of the air - housing transition belonging to the cast housing, the white widget covers the homogeneous part of the housing. Such a differentiation would not be possible with a standard 1D transfer function. Here the boundary of the housing as well as the pores can be clearly visualised alongside the homogeneous part of the housing.


(a) DVR.

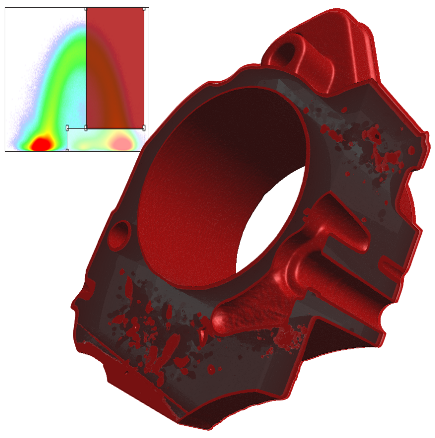

(b) Shaded DVR (illuminated clipping planes).

Figure 4.5: DVR rendering with a 2D transfer function based on density and gradient magnitude.

Figure 4.6 shows the gradient magnitude based transfer function exploited to visualise pores in a weld seam. Again this is shaded DVR mode. The material of the welding itself is rendered with very low opacity, the pores are rendered nearly opaque. The transfer function (small image) again shows the partial overlapping of the pores and material densities. This rendering however shows some problems with the density/gradient magnitude based transfer functions. The pores as well as the material surface share the same arch in the histogram, as the included material as well as the outside material have the same density (here it is air in both cases). Thus only a very small part of the arch can be covered with a widget as otherwise the material surface would be classified as well. A more sophisticated result can however be achieved with the feature-size based transfer function.

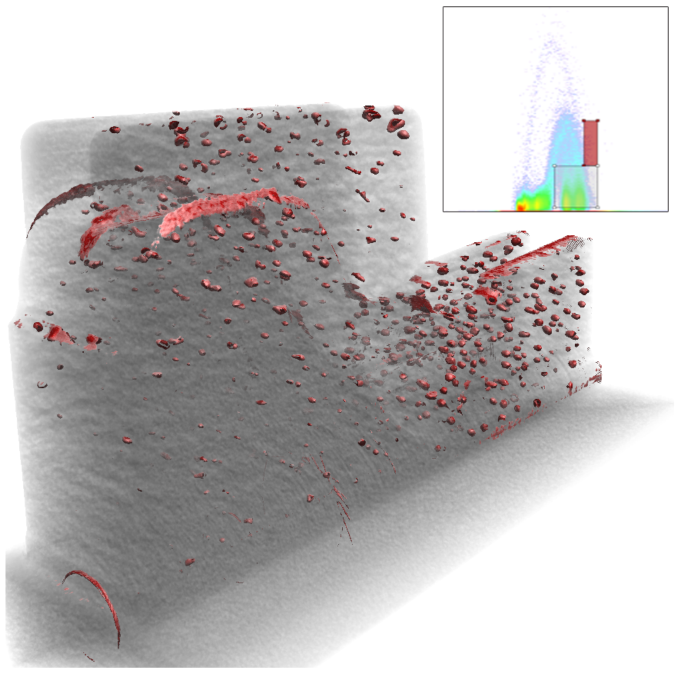


Figure 4.6: Visualisation of pores in the weld seam data set. DVR rendering with a 2D transfer function based on density and gradient magnitude.

Low/High Value-Based Render Mode

Figure 4.7 shows a rendering of the weld seam dataset with a low/high based transfer function. The rendering is very similar to the rendering made with a density/gradient magnitude based transfer function shown in Figure 4.6. However image quality of the latter one is a bit better and the low/high based rendering does not offer any advantages over the density/gradient magnitude based rendering regarding the rendering itself.

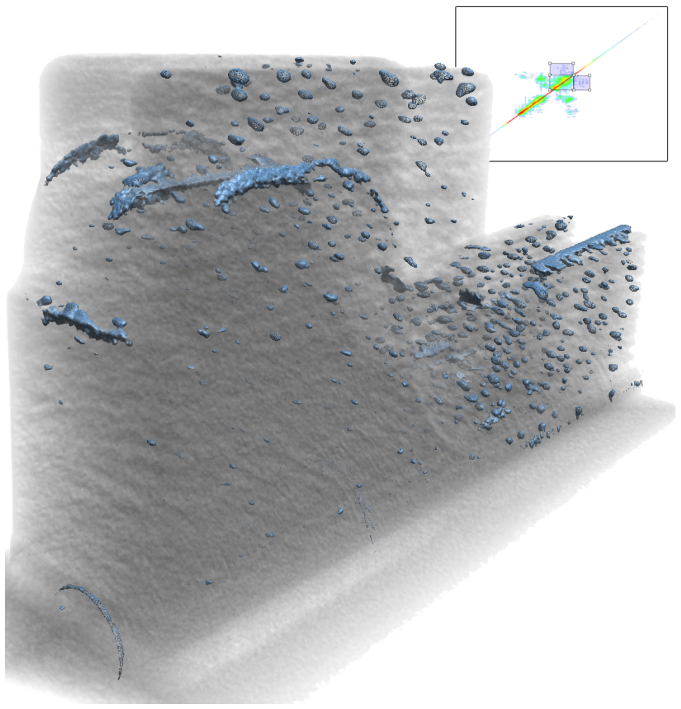


Figure 4.7: Visualisation of pores in the weld seam data set. DVR rendering with a 2D transfer function based on low/high values.

The main advantage of the low/high value transfer function can be seen easily when comparing the transfer functions with their histograms in detail, shown in Figure 4.8. The creation of the low/high value transfer function is much more intuitive than for the density/gradient magnitude based mode. The red box in Figure 4.8(a) covers only a very small part of the arch and the spot can only be found with cumbersome trial and error. Even though the low/high value histogram in Figure 4.8(b) is not as typical as the one in Figure 2.8(c) the indicated blobs mapping the pore boundaries can clearly be seen under the blue boxes and thus the creation of the transfer function is much easier compared to the density/gradient magnitude transfer function.

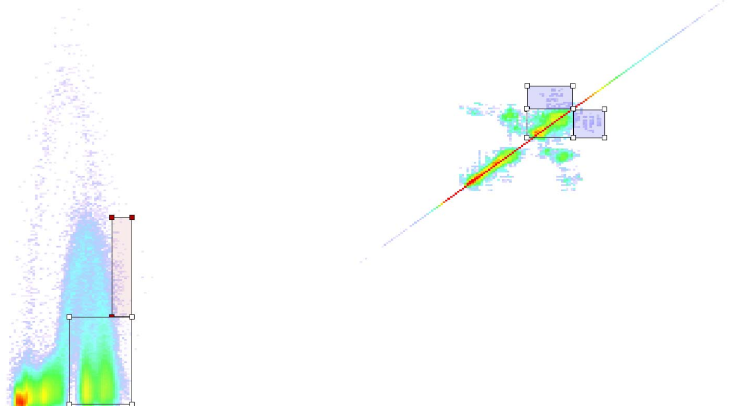

(a) GM 2D transfer function in detail. (b) Low/high 2D transfer function in detail.

Figure 4.8: 2D transfer functions for the weld seam data set.

Feature Size based Rendering

Figure 4.9 shows a rendering with a feature size based transfer function. Unshaded DVR was used with a simple 1D transfer function for the housing itself and a two dimensional transfer function with four simple boxes classifying four size ranges. The visualisation of the pores is a huge improvement over the renderings with the other transfer functions. The depth perception with unshaded DVR naturally is not very good, however the position of the pores can still be identified quite well.

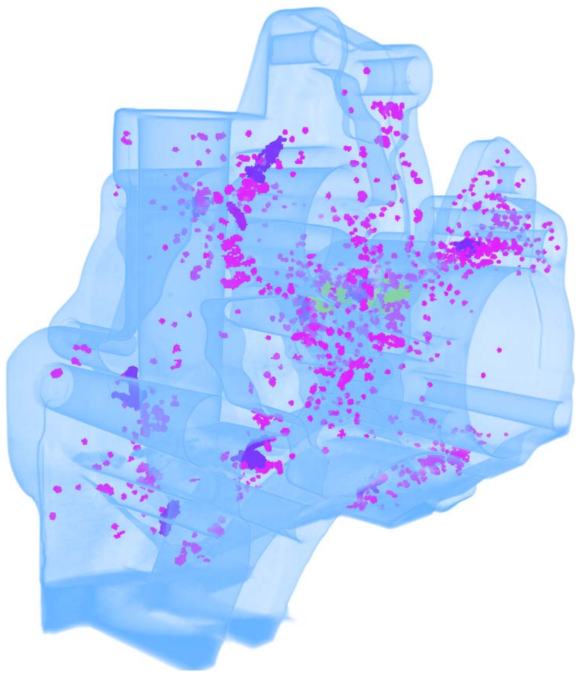
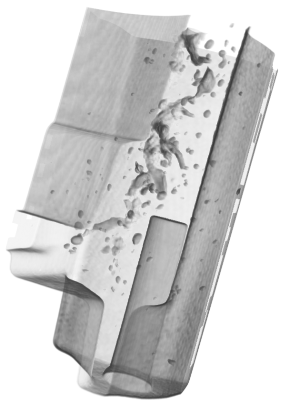
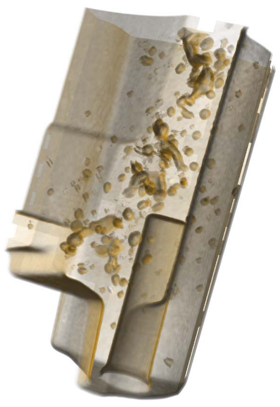
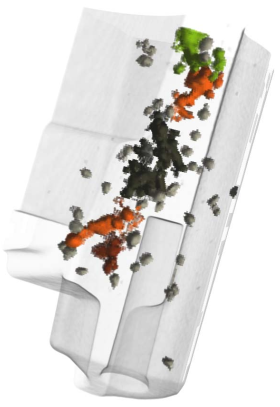


Figure 4.9: Unshaded DVR rendering of the cast housing with a feature size 2D transfer function.

Figure 4.10 shows a comparison between gradient magnitude modulated shaded DVR with a one dimensional transfer function, shaded DVR with a density/gradient magnitude based two dimensional transfer function, and finally shaded DVR rendering with a two dimensional transfer function based on density and feature size. The pores in the casting can be visualised quite well with all three modes, but with the first two the pores can not be classified different to the casting boundary. The step to the feature size transfer function is huge. The pores are not only clearly visible and can easily be distinguished from the boundary, pores of different sizes can also be classified differently. As visible in the transfer function below the 1D as well as the 2D transfer functions are very simple and the 2D histogram is a great guide to create the two dimensional transfer function.


(a) Gradient magnitude modulated shaded DVR rendering with a 1D transfer function.


(b) Shaded DVR rendering with a 2D transfer function based on density and gradient magnitude.


(c) Shaded DVR rendering with a 2D transfer function based on density and feature size.

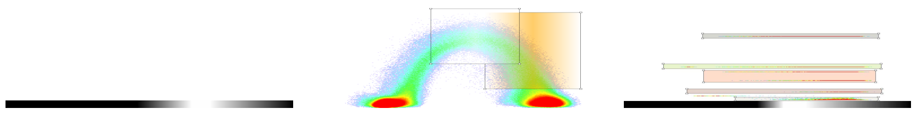


Figure 4.10: Renderings of a cutout of the cast housing dataset.

Chapter 5

Conclusions and Future Work

In this thesis two extensions were made to the hardware volume renderer with main focus on rendering of industrial CT data. The first one, was the correct illumination of clipping planes. The second one, which is the main part of this work, was the implementation of two dimensional transfer functions, including a graphical user interface and the adaptation of multiple render modes to take advantage of the different transfer functions.

5.1 Conclusions

The clipping plane illumination has a big effect on rendering results with gradient based shading. Clipping surfaces are clearly recognisable and indentations on the surface are clearly visible. Slice by slice ablation known from destructive testing methods can be simulated with isosurface rendering.
The clipping plane illumination virtually has no drawbacks, and memory consumption does not differ significantly as only seven more float variables are needed and performance differences are negligible. Under certain circumstances this mode is even faster than standard modes without illuminated clipping planes.
Concluding it can be said that the render modes without clipping plane illumination can safely be replaced by their counterparts incorporating this extension.

The three different two dimensional transfer function modes all work very well in their designated areas.
One thing that all three modes have in common is the fact that the histograms are usually a very good guide for creating the transfer function, which is typically not the case with one dimensional transfer functions. Cumbersome trial and error can often be avoided or minimised when using the 2D transfer functions. Even if the histogram is not as clear and trial and error is necessary this is a lot easier because the widgets placed on the drawing plane are independent, which is not the case with the one dimensional transfer function, where all nodes are connected, which can make the trial and error process with the 1D transfer function frustrating, because often one accidentally moves an existing node and the state before inserting a new node is usually very hard to restore.
We did not have the time to do extensive user testing with the new 2D panel, and usually users who are already familiar with the existing 1D panel can be sceptic, but the creation of sophisticated transfer functions is a lot easier with the new 2D panel, especially if the user is new to both panels.

The density/gradient magnitude based transfer function is very helpful to visualise material boundaries as well as pores (respectively the boundary of the pores). The same applies to the low/high based transfer function. However both approaches have different advantages and drawbacks. The main advantage of the low/high transfer functions are the easily interpretable histograms. Material boundaries can be identified very easily in the histogram. Rendering quality with the density/gradient magnitude transfer function on the other hand is a bit better. It is also not possible to select only partial boundaries with the low/high based transfer functions, as the local position inside a blob is merely random in the histogram and has no relation to the position of the boundary.
To sum this up both modes are excellent to visualise boundaries and have big advantages over simple 1D density based transfer functions. The density/gradient magnitude transfer function produces better renderings, while low/high transfer functions are a bit easier to create.
While the density/gradient magnitude based render modes are a bit slower due to the on-the-fly computation of the gradient magnitude, the low/high based modes need three times the memory, because of the additional low

and high value volumes.
An alternative to the real low/high values (which were also implemented during this thesis) are very basic low and high values which would be gathered by fetching only one step in both positive and negative gradient direction from the original sample. Using these values however removes one of the main advantages of real low/high values, namely the easily interpretable histograms. Also classification based on these values is problematic because noise can have a heavy influence in this case. The combination of a one dimensional transfer function for classification and a two dimensional transfer function based on these "fake" low/high values for lighting as presented in Subsection 2.3.2 could be a good application but after implementation and some testing we came to the conclusion that the rendering results do not fit our needs.

The third mode, based on density and the feature size is a very specialised solution to visualise pores. The kind of classification done with this mode can not be reached with any other mode, neither with one or two dimensional transfer functions. Rendering results very good. Pores can be clearly seen in the histogram and the creation of the transfer function is very intuitive. The pore tagging with the region growing algorithm works well, however, the gradient magnitude threshold and the maximum pores size sometimes have to be adjusted depending on the volume. The only negative point for this mode, especially for the very large data sets is the fact that the second volume containing the feature size is needed, doubling the memory consumption.

5.2 Future Work

A lot of improvements and extensions, especially for the two dimensional transfer functions presented in this work can be done.

To do detailed adjustments in the 2D transfer function panel a zoom function in the GUI would be of great use. Even if very good 2D transfer functions can be created with only a few widgets which do not even need exact placement for a lot of data sets, sometimes features are mapped to very small regions in the histogram and/or very close to other features. In such cases it would be useful to be able to zoom into such a region in the panel and move the widgets to exacter positions.

On the rendering side for now only the standard DVR and shaded DVR modes were adapted to the two dimensional transfer functions. The HVR framework however features a lot more render modes. Especially the gradient independent shaded DVR modes in combination with the two dimensional transfer functions could be a great tool for rendering industrial data. On the other hand the medical focused render modes could also take great advantage of higher dimensional transfer functions as well.

Of course other domains for the two dimensional transfer functions are also conceivable. For example, for multi volume rendering like PET-CT, instead of just blending the volumes it would be possible to create a multi-dimensional transfer function over all volumes at once.

Furthermore, combinations of the two dimensional transfer functions could be interesting. Using a full two dimensional density/gradient magnitude based transfer function instead of the gradient magnitude modulated 1D in combination with the feature size transfer function could further increase rendering quality for this mode.

But with all these extensions one should not forget the user. Good transfer function design was always a very hard task, and even though an increase of usability was one of the goals of this thesis which was (at least partly) reached, still is. More complex possibilities always require more complex user interaction, which could be counterproductive with respect to user friendliness.

Acknowledgements

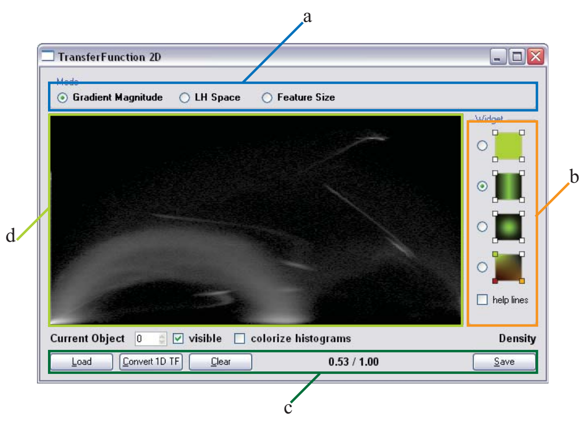
This work has been carried out at the Medvis group of the VRVis Research Center for Virtual Reality and Visualization in Vienna (`http://medvis.vrvis.at`). Most of the data sets are courtesy of the Österreichisches Gießerei Institut.
First of all, I would like to express my appreciation to those who contributed in the development of the volume renderer, I was allowed to work with.
I would like to express my sincere gratitude and appreciation to all who made this thesis possible. Foremost to my adviser Markus Hadwiger and to Katja Bühler. I thank the VRVis for providing me a work place and a computer and the whole staff for a pleasant working atmosphere throughout my time here. Further, I thank Stefan Müller and Matthias Biedermann for arising my interest in computer graphics and the motivation over the last years in my study.
Finally i want to thank my parents for their help in financing this semester abroad.

Appendix A

User Guide to the 2D Transfer Functions

The new two dimensional transfer functions are designed specifically to fulfil the needs of industrial CT rendering and non destructive testing. To just start working Section A.2 gives a short overview of a typical situation using the two dimensional transfer functions. A typical workflow is described using the main functions of the transfer function panel. A detailed description of all features of the two dimensional transfer function panel is given in Section A.3. The drawing widgets are described in full detail in Section A.4. And finally a list of hotkeys is given in Section A.5.

A.1 Nomenclature



a Mode Selector Menu

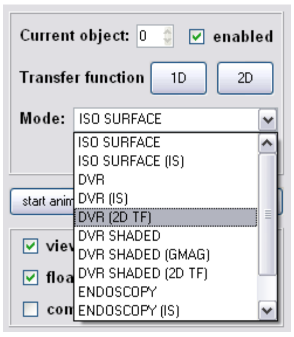
b Widget Selector Menu

c Load, Save, Clear and Convert Buttons

d The Drawing Canvas

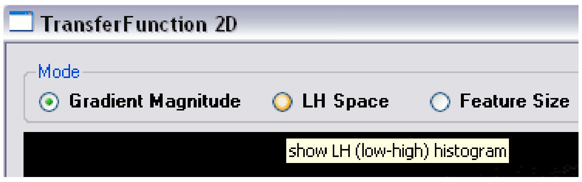
A.2 Workflow

Step 1: **Choosing 2D Transfer Function-Based Rendering and Opening the 2D Panel**
After loading a volume the render mode for use with the two dimensional transfer functions can be chosen by simply clicking the drop down menu on the left side of the volume renderer main window. In the drop down menu choose either *DVR (2D TF)* or *DVR SHADED (2D TF)* whether unshaded or shaded rendering is required. By default the density/gradient magnitude-based 2D render mode is now used. The render mode automatically adjusts to the histogram that is used in the transfer function panel. At this point the render window will not show a volume as no standard transfer function is defined when starting. After that a click on the button *2D* will open the 2D transfer function panel.



Step 2: **Changing the 2D Transfer Function Render Mode**
Atop the 2D transfer function window a set of radio buttons entitled *Mode* is used to switch between the guiding histograms and the corresponding render modes. The standard *Gradient Magnitude* uses the density and gradient magnitude histogram and rendering modes and is an excellent choice

to render solid material as well as material boundaries with smooth transitions. The *LH Values* entitled button switches to the Mirrored Low High histogram and render mode especially suited to render material boundaries. Finally the histogram and render mode based on the feature size is selected with the *Feature Size* radio button. This mode should be chosen when pores in a material shall be visualised.



Step 3: Creating the Actual Transfer Function

When the render mode is set the transfer function can be created (of course the render mode can still be changed after creating a transfer function, however a transfer function created based on one render mode will usually not fit the other modes). This is done by picking a widget from the menu on the right-hand side of the transfer function window. Now a click with the left mouse button into the canvas windows will place a widget of the chosen type on this position. Dragging the vertices or the edges of the widget will change the size, clicking onto the body of the widget will open a dialogue to chose the colour. The transparency of the widget can be set in this dialogue also by changing *Alpha Channel* value in the lower right corner. Place multiple of these widgets onto the drawing canvas to define the transfer function. Watch the changes in the rendering while doing so.

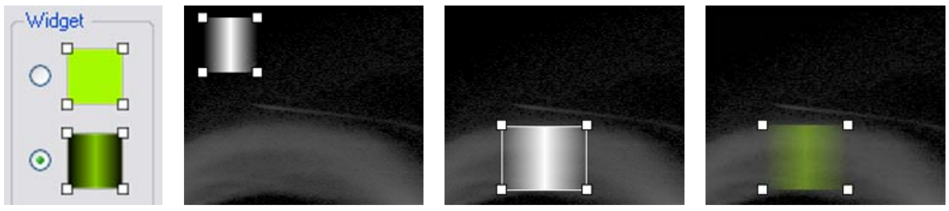
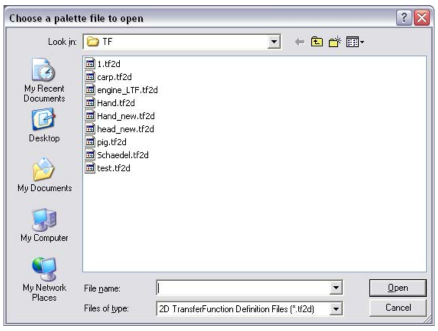


Figure A.1: Pick a widget, place it, move, resize and set colour.

A.3 The Panel

Part 1: **Load, Save, Convert, Clear**
Just as 1D transfer functions, 2D transfer functions can be saved and loaded. Just click the *Save* respectively *Load* button on the bottom of the window to access the save and load dialogues. The standard folder will be the *TF* folder inside the apps directory. When choosing a different folder this will be remembered as long as the application is running. Natively two dimensional transfer functions are saved in **.tf2d*-files. However, especially useful when using the gradient magnitude based mode, a 1D transfer function definition can be loaded directly into the 2D panel by clicking load and choosing *1D Transfer Function Definition* from the file type drop down menu.
ATTENTION: Loading a 1D definition into the 2D panel will also overwrite the current 1D transfer function in the 1D transfer function panel. If the current 1D transfer function shall be converted just click the *Convert 1D TF*-button. The resulting two dimensional transfer function will be the exact representation of the one dimensional transfer function for the gradient magnitude-based and the feature-size-based render modes, however when using the Low-High render mode a conversion will not lead to useful results.
If something went completely wrong and the created 2D transfer function is completely messed up its not required to delete all items by hand, but the whole definition can be cleared with one click on the *Clear*-button.

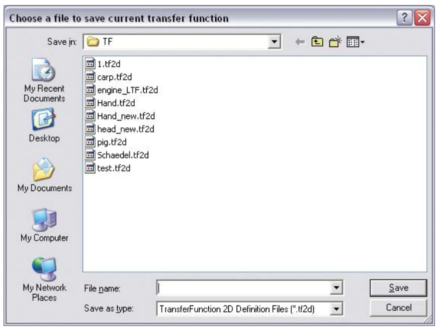

Choose a palette file to open

Look in: TF

1.tf2d
carp.tf2d
engine_LTF.tf2d
Hand.tf2d
Hand_new.tf2d
head_new.tf2d
pig.tf2d
Schaedel.tf2d
test.tf2d

File name:
Files of type: 2D TransferFunction Definition Files (*.tf2d)

(a) Load


Choose a file to save current transfer function

Save in: TF

1.tf2d
carp.tf2d
engine_LTF.tf2d
Hand.tf2d
Hand_new.tf2d
head_new.tf2d
pig.tf2d
Schaedel.tf2d
test.tf2d

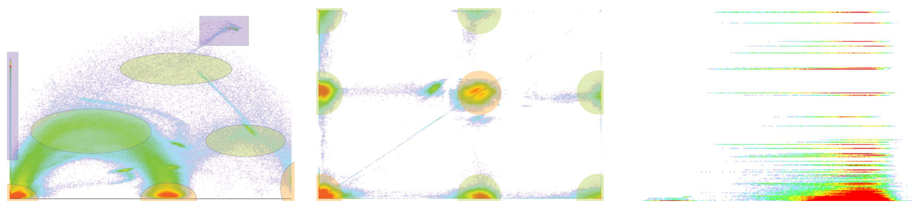
File name:
Save as type: TransferFunction 2D Definition Files (*.tf2d)

(b) Save

Figure A.2: Standard windows file dialogues.

Part 2: Place, Move, Colorise, Delete

The actual creation of a transfer function is very simple. Just choose one of the widgets from the menu on the right-hand side by clicking the corresponding radio button. A Click with the left mouse button into the drawing canvas will then place a widget of that type with the standard size, in white colour and full opacity on that position. By left-clicking, holding and then dragging a widget can then be moved to its desired position. Widgets ”snap” together, meaning if certain areas of one widget get into a region of another widget it will they will automatically be set to the same position. How the different widgets exactly snap to each other is described in the widgets Section A.4.
To fit its size either drag one of the vertices to modify the height and width at once or grab an edge to manipulate only the width or the height. Two snapped widgets can be resized simultaneously by grabbing the overlapping edge. Click the *Helplines*-checkbox under the widget menu to show help lines on the edge of each widget.
Set the colour and opacity of the widgets by clicking onto the widget. Enter the desired opacity in range of 0 to 255 in the box entitled *Alpha Channel* on the lower right of the colour chooser pop up. The opacity of the last used widget can also always be changed by rolling the mouse wheel. Snapping and simultaneous resizing can be turned of by holding the *Shift*-key while clicking and dragging.
If a widget is not needed anymore it can easily be removed from the canvas by clicking on it with the right mouse button.



(a) Density / Gradient Magnitude Based Histogram (b) LH Histogram (c) Feature Size Histogram

Figure A.3: In GM and LH histograms place widgets on the orange marked areas to visualise solid material regions, the green marked regions represent the material boundaries. The purple regions usually are maverick values on i.e. on the border of the data set.

Part 3: **Histogram**
Use the histogram as a guide to regions of interest. Higher quantities of volume elements with similar properties are represented in the histogram by brighter areas. If areas of lower quantities are hard to recognise use the *Colorize Histogram* checkbox on the bottom of the window to use a colour code instead of intensities to visualise quantities. The background will switch from black to white, the lowest quantities will be shown in purple, followed by blue, cyan, green, yellow, orange and finally the highest denomination will be shown in red.
Three different kinds of histograms can be chosen with the radio button menu on the top of the window. The choice of the histogram will also affect the render mode so that transfer function and rendering will fit together.

A.4 The Widgets

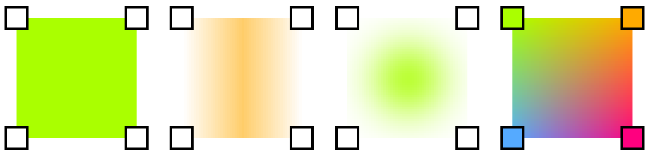

(a) Solid (b) Peak (c) Gauss (d) Free

Figure A.4: The four widgets.

Part 1: **Solid Colour Widget**
This widget has a single colour and opacity for its complete area. When created it is white and fully opaque, a click on the body opens the colour dialogue. As all other widgets it is moved by dragging the body and resized by dragging either a vertex or an edge. This widgets snaps with its edges on all sides.

The solid colour widget is very useful to classify large ranges of values which belong to the same material without the need of a smooth transition on the borders of this material on the one hand and on the other hand when very small regions in the histogram shall be covered with sharp edges. Use it to classify the ranges that shall be rendered opaque and spread up to a lot of values (i.e. when using gradient magnitude mode often the material with the highest density in the area on the right-hand side of the histogram). This widget for the feature size-based rendering. Place it on a horizontal line to cover all pores of the particular size.

Part 2: Colour Peak / Ramp Widget
This widget, just like the solid colour widget has a single colour but the alpha defined in the colour dialogue defines the maximum alpha value in the middle of the widget (in x-direction). From there the alpha is defined by linear ramps towards zero in both directions. Colour, size and position are defined just like for the solid colour widget. This widget also snaps to its bottom and top edge, but vertical snapping is done to the middle, where the alpha ramps meet.

The peak is very useful to define smooth transitions between two materials, but not very efficient to classify very small value ranges. Use it on the arches of the gradient magnitude histogram to create very smooth renderings of material borders.

Part 3: Gauss Blob Widget
The gauss blob also has only one colour and just like the peak widget the alpha value defines the maximum opacity, however here the alpha value is based on a two dimensional gaussian distribution. Again colour, size and position are defined just like for the solid colour widget but this widget does not snap at all to other widgets, as it does not have clear borders. Different to the other widgets this one can be moved over the borders of the histogram till its midpoint.

This widget is especially suitable for the low high histogram, place it on a blob and fit its size to dismiss the outliers near it. It is also useful when using the gradient magnitude histogram to render homogeneous material which usually also is represented by a half blob on the bottom of the histogram. Shove the widget over the histogram borders to fit it exactly to the histogram.

Part 4: Free Colour Widget
Snapping, moving and resizing of this widget works just like with the solid colour widget. However, for this widget the colour and opacity can be set individually for every vertex by clicking the marker on the vertex, between

the vertices the colour gets interpolated. It is possible to change the colour for all four vertices just as with the other widgets and in addition the colour of two vertices can be set by clicking the edge between the two vertices. Just as for resizing the help-lines might be useful to hit the edge.

Most likely the complexity of the free colour widget will be needed only in very rare cases. However, it will be used whenever a one dimensional transfer function is converted. Use this widget whenever a single opacity ramp or a smooth colour transition is needed.

A.5 Hotkeys

- Left Mouse Button: Place a widget / set colour, opacity
- Right Mouse Button: Remove a widget
- Mouse Wheel: Modify the opacity of the last used widget
- Alt + L: **L**oad a transfer function definition
- Alt + S: **S**ave a transfer function definition
- Alt + E: Cl**e**ar a transfer function definition
- Alt + C: **C**onvert a transfer function definition
- Alt + 1: Switch to Gradient Magnitude mode
- Alt + 2: Switch to Low High mode
- Alt + 3: Switch to Feature Size mode
- Alt + O: C**o**lorise histograms
- Ctrl + 1: Use the Solid widget
- Ctrl + 2: Use the Peak widget
- Ctrl + 3: Use the Gauss widget
- Ctrl + 4: Use the Free widget
- Ctrl + H: Show the widget **H**elplines
- Shift: Hold to disable Snapping

Bibliography

[AB94] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6):641–647, 1994.

[AMD05] Radeon x1800 graphics technology, 2005. `http://ati.amd.com/products/radeonx1800/index.html` March, 6th, 2007.

[BCL$^+$06] Alex Laier Bordignon, Rener Castro, Hélio Lopes, Thomas Lewiner, and Geovan Tavares. Exploratory visualization based on multidimensional transfer functions and star coordinates. *sibgrapi*, pages 273–280, 2006.

[BIN07] About NDT, 2007. `http://www.bindt.org` March, 6th, 2007.

[Bli82] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH '82*, pages 21–29, 1982.

[Cro84] Franklin C. Crow. Summed-area tables for texture mapping. In *SIGGRAPH '84*, pages 207–212, 1984.

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *SIGGRAPH '88*, pages 65–74, 1988.

[EHK$^+$04] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Aaron E. Lefohn, Christof Rezk Salama, and Daniel Weiskopf. Real-time volume graphics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, 2004.

[EHK^{+}06] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, and Daniel Weiskopf. *Real-time volume graphics*. A K Peters, Wellesley, Mass., 2006.

[HBH03] Markus Hadwiger, Christoph Berger, and Helwig Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of IEEE Visualization 2003*, page 40, 2003.

[HKSB06] Markus Hadwiger, Andrea Kratz, Christian Sigg, and Katja Bühler. Gpu-accelerated deep shadow maps for direct volume rendering. In *Graphics Hardware 2006*, pages 49–52, 2006.

[HM03] Runzhen Huang and Kwan-Liu Ma. Rgvis: Region growing based techniques for volume visualization. In *PG '03*, pages 355–363, 2003.

[HMMW03] Runzhen Huang, Kwan-Liu Ma, Patrick McCormic, and William Ward. Visualizing industrial ct volume data for non-destructive testing applications. In *IEEE Visualization '03*, pages 547–554, 2003.

[HP06] Joachim Hornegger and Dietrich Paulus. Medizinische bildverarbeitung 1, 2005/06. Not public available online. Contact: `http://www.uni-koblenz.de/FB4/People/Person/paulus`.

[Kan01] Eser Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *KDD '01*, pages 107–116, 2001.

[KHS^{+}06] Andrea Kratz, Markus Hadwiger, Rainer Splechtna, Anton L. Fuhrmann, and Katja Bühler. Gpu-based high-quality hardware volume rendering for virtual environments. In *International Workshop on Augmented Environments for Medical Imaging and Computer Aided Surgery*, 2006.

[KKH01] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. *IEEE Visualisation '01*, pages 255–262, 2001.

[Kla06] Oliver Klar. Interactive gpu based segmentation of large medical volume data with level sets. Master's thesis, Universität Koblenz, VRVis Research Center, 2006.

[KW03] J. Krüger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *IEEE Visualization 2003*, page 38, 2003.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, pages 163–169, 1987.

[Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.

[LM04] Eric B. Lum and Kwan-Liu Ma. Lighting transfer functions using gradient aligned sampling. In *IEEE Visualization '04*, pages 289–296, 2004.

[Mül04] Stefan Müller. Computergraphic 1, lecture materials. online, 2004. `http://www.uni-koblenz.de/FB4/Institutes/ICV/AGMueller/Teaching/SS04/CG1/Materialien`, March, 6th, 2007.

[nVI04] Geforce 6 series technical briefs, 2004. `http://developer.nvidia.com/object/geforce_6_series_tech_briefs.html` March, 6th, 2007.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[ŠBG06] Petr Šereda, Anna Vilanova Bartroli, and Frans A. Gerritsen. Mirrored lh histograms for the visualization of boundaries. In *VMV*, pages 237–244, 2006.

[SBH07] Florian Schulze, Katja Bühler, and Markus Hadwiger. Interactive deformation and visualization of large volume datasets. In *Grapp; 2nd International Conference on Computer Graphics Theory and Applications*, 2007.

[ŠBSG06] Petr Šereda, Anna Vilanova Bartroli, Iwo W.O. Serlie, and Frans A. Gerritsen. Visualization of boundaries in volumetric data sets using lh histograms. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):208–218, 2006.

[Sch05] Henning Scharsach. Advanced gpu raycasting. In *CESCG Proceedings*, pages 69–76, 2005.

[SHN$^+$06] Henning Scharsach, Markus Hadwiger, André Neubauer, Stefan Wolfsberger, and Katja Bühler. Perspective isosurface and direct volume rendering for virtual endoscopy applications. In *EUROVIS - Eurographics /IEEE VGTC Symposium on Visualization*, pages 315–322, 2006.

[Sie06] Excellence in ct, somatom definition, 2006. Available online at `http://www.medical.siemens.com/siemens/en_INT/gg_ct_FBAs/files/brochures/ct_somatom_definition.pdf` on February 7th 2007.

[Tro05] *Qt Reference Documentation*, 2005. `http://doc.trolltech.com/3.3/`, March, 6th, 2007.

[Tro07] *What's New in Qt 4.2*, 2007. `http://doc.trolltech.com/4.2/qt4-2-intro.html` March, 6th, 2007.

[WEE03] Daniel Weiskopf, Klaus Engel, and Thomas Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics*, 09(3):298–312, 2003.