Visualisation of Two-Dimensional Volumes

vorgelegt von
Sebastian Löbbert, Bachelor of Science
aus Dernbach/Westerwaldkreis

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuß:

Vorsitzender:	Prof. Dr. Hellwich
Berichter:	Prof. Dr. Lemke
Berichter:	Prof. Dr. Santos (Universidad Politécnica de Madrid)

Tag der wissenschaftlichen Aussprache: 14. Juli 2004

Berlin 2004
D 83

Zusammenfassung

In dieser Arbeit wird ein neues Verfahren zur Visualisierung zweidimensionaler Volumen vorgestellt. Der Begriff multidimensionales Volumen wird dabei definiert als eine Menge von räumlich dreidimensionalen Datensätzen, die jeder eine andere Eigenschaft (eine physikalische Qualität, z.B. Dichte oder Temperatur) desselben Objekts beschreiben. Zweidimensionale Volumen beschreiben also zwei verschiedene Eigenschaften eines Objekts. Sie entstehen z.B. in biomedizinischen Anwendungen, wenn gleichzeitig funktionale und anatomische Datensätze untersucht werden.

Zunächst wird der Stand der Technik in der Visualisierung zweidimensionaler Volumen dargelegt. Dabei sind besonders die folgenden Schwächen bestehender Verfahren erkennbar:

- Schlechte räumliche Darstellung und schlechte Lokalisierbarkeit von Ausprägungen (bemerkenswerte Quantitäten einer Eigenschaft an einer Stelle).
- Beschränkung auf Datensätze aus speziellen Quellen oder spezielle Kombinationen von Datensätzen.
- Prinzipbedingte Beschränkung einer Eigenschaft auf wenige kleine Regionen innerhalb der anderen Eigenschaft.

Basierend auf diesen Defiziten werden die Anforderungen für ein besseres Visualisierungsverfahren herausgearbeitet, anhand derer ein neues Verfahren, *dependent rendering* genannt, entwickelt wird. Das Verfahren basiert auf der Annahme, dass bei der Visualisierung mehrerer Eigenschaften immer eine Eigenschaft als Referenz zur Lokalisierung dienen kann. Abhängig von der ersten kann eine weitere Eigenschaft visualisiert werden.

Es werden drei Implementierungen des Verfahrens vorgestellt, die ersten beiden sind Prototypen, die dritte eine spezialisierte Anwendung für eine biomedizinische Visualisierungsplattform. Die Implementierungen veranschaulichen, dass sich das vorgestellte Verfahren gegenüber bestehenden Ansätzen besonders durch folgende Punkte auszeichnet:

- Gute Lokalisierbarkeit von Ausprägungen bei gleichzeitiger guter räumlicher Darstellung des Objekts (z.B.: “Ist es auf der Oberfläche heiss oder innerhalb des Objekts?”).
- Gleiche räumliche Ausdehnung beider Datensätze möglich.
- Genereller Ansatz: Keine Beschränkung auf Datensätze aus speziellen Quellen oder auf spezielle Kombinationen von Datensätzen.

Das vorgestellte Verfahren stellt daher einen bedeutenden Fortschritt in der Technik der Visualisierung zweier Eigenschaften eines Objekts dar.

Abstract

In this thesis, a new technique for the visualisation of two-dimensional volumes is presented. The term multi-dimensional volume is defined as a set of spatially three-dimensional data sets, each of them describing another property (a physical quality, e.g. density or temperature) of the same object. Thus, two-dimensional volumes describe two different properties of an object. They are used e.g. in biomedical imaging, where anatomical and functional data are examined jointly.

First, the state of the art in the visualisation of two-dimensional volumes is presented. In the course of this, the following deficiencies of existing approaches become apparent:

- Unsatisfactory 3D impression (it is difficult to mentally reconstruct the spatially three-dimensional object from the rendering) and difficult localisation of features (i.e. remarkable characteristics in the quantity of a property at a given location).
- Restriction to data sets from particular origins or particular combinations of data sets.
- By design, one property is restricted to only a few small regions inside the other property.

Starting from these deficiencies, the requirements for a visualisation technique that overcomes these limitations are elaborated. These are then used to develop a new technique, called *dependent rendering*, which is based on the assumption that, when visualising two properties of an object, there is alway one property that can serve as a spatial reference for the other. The other property is then visualised in dependency on this reference.

Three implementations of the technique are presented, the first two are prototypes, the third one is a specialised application for a biomedical visualisation platform. The implementations show that, compared to existing approaches, the presented technique especially stands out because of the following features:

- Precise localisation of features combined with good 3D impression of the object (e.g. “Is it hot on the surface or only inside the object?”).
- Both data sets can be extended over the same region.
- General approach: No restriction to data sets from particular origins or particular combinations of data sets.

The presented technique therefore represents an important advancement in the joint visualisation of two properties of an object.

Acknowledgements

This research was carried out during my time as an external doctoral student at the department of Computer Graphics and Computer Assisted Medicine of Professor Dr. H.U. Lemke at the Technische Universität Berlin and supervised by Dr.-Ing. Steffen Märkle. To him, I am especially grateful for the many discussions and valuable suggestions. During this time, I was employed as a part-time software developer at Infopark AG, Berlin, which allowed me to conduct the research independent of financial aids.

A significant part of the work described in this thesis (the research and implementation described in section 5.2) was carried out during my 6-month stay as a Marie Curie fellow at the biomedical imaging group of Professor Andrés Santos at the Universidad Politécnica de Madrid. This fellowship was financed by the European Union. I thank Professor Andrés Santos for inviting me to Madrid and supervising my work from this time onwards. I also wish to thank Javier Pascau and Manuel Desco from the Hospital General Universitario Gregorio Marañón in Madrid for discussions on what visualisation features are important for their work with biomedical data.

Special thanks go to Ursula Passing for continuous proof-reading and many discussions on how to structure scientific work and how to explain complex matters understandably. It is also her achievement that I was able to enjoy this demanding time.

Parts of the work presented in this thesis have already been presented on international conferences: [LM02b] (see section 5.1.2), [LM02a] (see section 6.1) and [LPDS03] (see section 5.2).

Contents

List of Figures

List of Tables

Abbreviations

2D	two-dimensional (in space)
3D	three-dimensional (in space)
aka	also known as
AUC	Area Under the Curve
CIE	Commission Internationale de L'Éclairage
CT	Computed Tomography
DVR	Direct Volume Rendering
EEG	ElectroEncephaloGram
FFT	Fast Fourier Transform
fMRI	functional Magnetic Resonance Imaging
FVR	Frequency domain Volume Rendering
GUI	Graphical User Interface
HGUGM	Hospital General Universitario Gregorio Marañón (Madrid)
HSV	Hue-Saturation-Value
IDL	Interactive Data Language by Research Systems Inc.
LUT	Look-Up Table
MEG	MagnetoEncephaloGram
MIP	Maximum Intensity Projection
MRI	Magnetic Resonance Imaging
NMR	Nuclear Magnetic Resonance
PET	Positron Emission Tomography
QFD	Quality Function Deployment
RGB	Red-Green-Blue
RGBA	Red-Green-Blue-Alpha
ROC	Receiver Operator Characteristic
SPECT	Single Photon Emission Computed Tomography
UML	Unified Modeling Language

Chapter 1

Introduction

1.1 Research question and motivation

Volume visualisation is a method for the visualisation of spatially three-dimensional data sets of arbitrary objects with applications in medicine, engineering, science and other fields. Its purpose is to "extract meaningful information from volumetric datasets through the use of interactive graphics and imaging" [KCY93, p. 51].

Unfortunately, many properties of an object, like for example the temperature distribution, cannot be visualised meaningfully without using a second property, like the matter distribution, as a spatial reference.

While visualising a single property of an object, especially the matter distribution, using volume rendering is well researched, little is known about how to visualise several properties, like the matter and the temperature distribution, at the same time.

The ability to visualise two or more properties of an object is crucial for correctly interpreting data sets, for example in order to precisely locate the epileptic sources in epilepsy patients' brains. For this purpose, two data sets, one for the anatomical structure of the brain and one for the electrical signals inside the brain, must be jointly visualised.

There are some approaches to jointly visualise two data sets, but they have significant shortcomings, for example many techniques require the second property to be restricted to few regions that cover only a small fraction of the extension of the first property.

The goal of the work presented in this thesis is to develop a method which is suitable for visualising two properties of one object in a single projection, enabling an observer to recognise relations between the two properties. This method shall be as generally applicable as possible. Therefore is should not be restricted to a special application domain (like biomedical imaging), but be applicable to data sets representing arbitrary properties.

1.2 Outline of the thesis

After presenting some background information on volume rendering in general, so-called multi-dimensional volumes, i.e. data sets that represent several properties of an object, are introduced. Then the term "multi-dimensional volume" is defined formally. Both the origin and possible uses of multi-dimensional volumes are presented. Common methods for visualising multi-dimensional volumes are introduced and their advantages and shortcomings are discussed. Based on this discussion, requirements for visualising multi-dimensional volumes are derived.

From these requirements, a new method to visualise multidimensional volumes is developed, named "dependent rendering". This approach is explained in depth.

The algorithm is then evaluated by three implementations: A general purpose prototype implementation, a second prototype that overcomes deficiencies in the first prototype and a specialised implementation for biomedical imaging.

Finally, some conclusions are drawn and an outlook to further research is given.

Chapter 2

The background: Volume rendering

In this section, first a short introduction to volumetric data and where it originates from is given. In this context, the concept of a *voxel* is introduced. Then the purpose of volume visualisation is outlined and some of the most common techniques are introduced with emphasis on the volume rendering approach. The techniques described in this chapter are the foundation for the algorithm and the implementations used for the work described in this thesis.

2.1 Volumetric data sets

Volumetric data sets describe a property of a spatially three-dimensional object by sampling this property's underlying function on a discrete three-dimensional grid. These data sets usually stem from simulation or measurement, examples for the latter being CT (Computed Tomography) or MRI (Magnetic Resonance Imaging, also known as NMR, Nuclear Magnetic Resonance).

After the raw data have been acquired, they are usually transformed such that they fit on a rectangular grid. In the context of this thesis, such data sets are also called a *volume*. To simplify the discussion, it is assumed throughout this chapter that the property described by the data set can be mapped to the matter distribution of the object. E.g. using CT, the electron density inside the object is measured, which can at least partially be mapped to the materials the object consists of.

2.2 Voxel representation

Each sampling point on the data set grid contains a discrete scalar value. This value is called a *voxel* (short for *volume element*) and can be interpreted in two different ways (see [LCN98, pp. 15–16]):

- A voxel is a size-less point in space representing the scalar value of the described property's underlying function at this sample point.

- A voxel is a small cube of constant value with its edges' length given by the sampling distance. The voxel's value represents the average value of the underlying function in this cube.

Throughout this chapter, the first of these interpretations is used. It allows to interpolate the values of samples not falling on the grid points from its neighbours. Besides the underlying functions' value, further attributes like materials or flow directions can be associated with a voxel.

2.3 Volume visualisation

The goal of volume visualisation is to produce a two-dimensional rendering of a three-dimensional object conveying as much information as possible on the inner parts and on the three-dimensional structure of the object. An overview of different volume visualisation techniques is given in [Yag93], while [Kau91] contains a selection of fundamental articles on volume visualisation. In the next section, some common techniques for volume visualisation are presented. Thereafter, one of these techniques, namely volume rendering, is described in more detail.

2.3.1 Visualisation techniques for volumetric data sets

The general setup valid for all techniques described here is depicted in figure 2.1: Two coordinate systems are set up: The first one is the *image space* coordinate

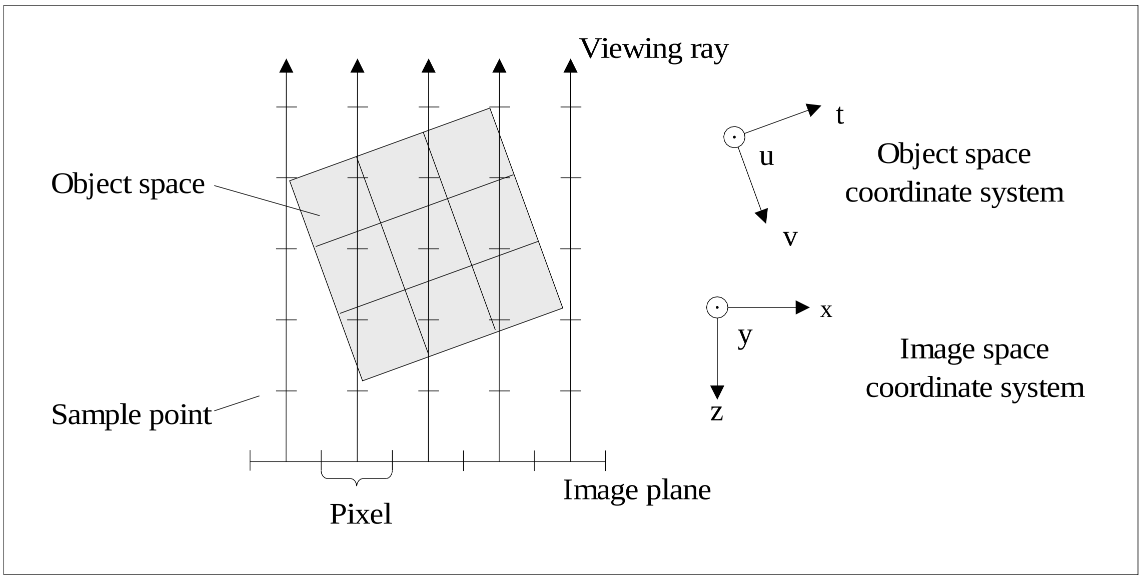


Figure 2.1: Basic setup of coordinate systems and viewing rays for volume rendering (The y-axis and u-axis in the coordinate systems is perpendicular to the paper and points to the reader: right-handed coordinate system)

system of the resulting projection image (coordinates x,y and z), and the second one the *object space* coordinate system of the volume data set representing the object (coordinates t,u and v). It is common to denote the two axes of the projection image as x and y axis while the axis perpendicular to the image plane is the z axis, which is parallel to the viewing vector. To view the volume from different directions, the object is rotated and, if necessary, translated and scaled. A transformation matrix allows to transform one coordinate system's coordinates into the other's coordinates.

Volume visualisation techniques can be classified as two groups, image space and object space based techniques. For object space based techniques, also called forward mapping techniques, the volume is traversed and for each of the voxels the contribution to the projection image is calculated. When using image space based techniques, also called inverse mapping techniques, for each pixel in the projection image the appropriate voxels' contributions are combined to form the pixel value.

A common volume visualisation technique is the *ray casting* approach, which is a image space based technique. In ray casting, for each pixel in the projection image, a viewing ray is sent into the volume and the voxels along this ray are combined according to a certain composition function. In the following, common volume visualisation techniques are presented.

Surface rendering An *iso-surface* (i.e. a surface along which the voxel value is constant, an iso-surface can, for example, describe the boundary of a region made up from a specific material) of the object is extracted from the data set and then visualised. A common algorithm for extracting surfaces from volumetric data is the *marching-cube* algorithm described in [LC87]. Although being computationally inexpensive once the surface has been extracted, this technique has the decisive drawback of giving no information of what is inside the object (see [KCY93]), which is what the observer is interested in, therefore this method is not further discussed here. It is possible to extract the surfaces of several regions of the object, and render these surfaces semi-transparently, such that surfaces that are inside the object shine through those that are near to the border of the object.

Maximum intensity projections MIP project the three-dimensional data set on a two-dimensional plane by taking into account only the maximal voxel value along each viewing ray. Although containing information about the interior of an object, the resulting images are visually flat and contain no depth information (see [MHG00]), therefore it is not further discussed here.

Average or X-ray composition These techniques calculate the average over all voxels along each viewing ray to form the projection image. The resulting projection image looks quite similar to conventional X-ray images, it contains information of the object's inner parts and has, at least for the trained user, some depth information.

Volume rendering using the *over* operator This technique tries to discretise the *ray casting integral* introduced in section 2.3.2. There are several variations of this very common approach. For fundamental articles on volume rendering, see e.g. [DCH88], [Wes89], [Lev88]. A book on volume rendering covering many aspects is [LCN98]. Volume rendering is also the approach which is used for the work presented in this thesis and will therefore be further described in the next section. The term volume rendering is often used to implicitly denote this technique.

2.3.2 Volume rendering

The steps performed during volume rendering can be ordered in the *volume rendering pipeline* which is briefly introduced in this section.

The volume rendering pipeline (see figure 2.2) discussed here was first introduced by Levoy (see [Lev88],[Lev90]). Although actual implementations do not fully adhere to this pipeline, it is well worth studying because it clearly shows the principles. While the data are prepared, they are brought on a rectangular grid. Thereafter

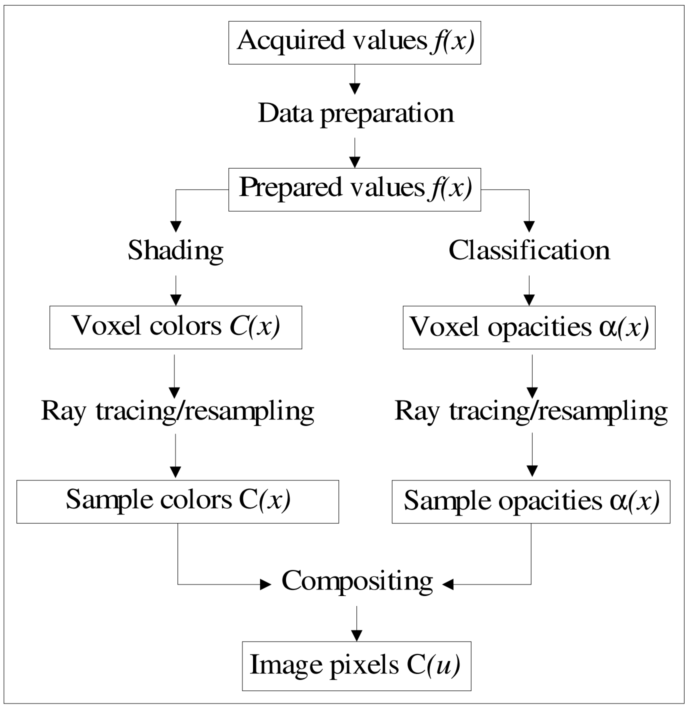


Figure 2.2: Volume rendering pipeline
(Adapted from [Lev88])

two processes are performed independently of one another:

In the *shading* process, illustrated in the left branch in figure 2.2, the volume is shaded. In volume rendering, each voxel is considered as a primitive object with its own surface. The direction of the surface is determined by calculating the local gradient of the voxel values. Each voxel is shaded by taking its surface direction, the viewing vector and virtual light sources and applying a shading operation, like e.g. Gouraud [Gou71] or Phong [Pho75] shading (see below). The resulting grey values or colours can be stored in a separate data set for later retrieval.

In the *classification* process (the right-hand branch), the volume is classified, i.e. to each voxel an opacity and a colour is assigned according to its value and some classification function. By changing the classification function, it is possible to deliberately hide or highlight certain materials. As with shading, the resulting opacities can be stored in a separate data set for later use.

Finally, for each pixel in the projection image, a viewing ray is cast into the data set, and for each sample point on this viewing ray, the colours and opacities from shading and classification are combined. These sample values are then composed to yield the pixel value.

This simple volume rendering pipeline has been refined in many ways (see e.g. [MWG98]), but the basic steps remain the same: data preparation, shading and classification, composition.

Several techniques exist for the composition step. The simplest ones are MIP, which just takes the highest sample value on the viewing ray, and average or X-ray composition (see above), which sums up all sample values on a viewing ray and finally divides this sum by the number of samples on the ray. Unfortunately, the images generated by these composition techniques contain no or only little depth information. Therefore it is very hard or even impossible to receive information about the three-dimensional structure of the object from these images..

A composition operator that overcomes these shortcomings is the *over* operator developed by [PD84]. In the following, a short introduction to an optical model for the interaction between light and matter is given and it is shown how this model can be applied for volume rendering. A survey of different optical models for the interaction of light and matter is given in [Max95]. For a sketch illustrating the model, see figure 2.3. When a beam of light hits a piece of matter, parts of the

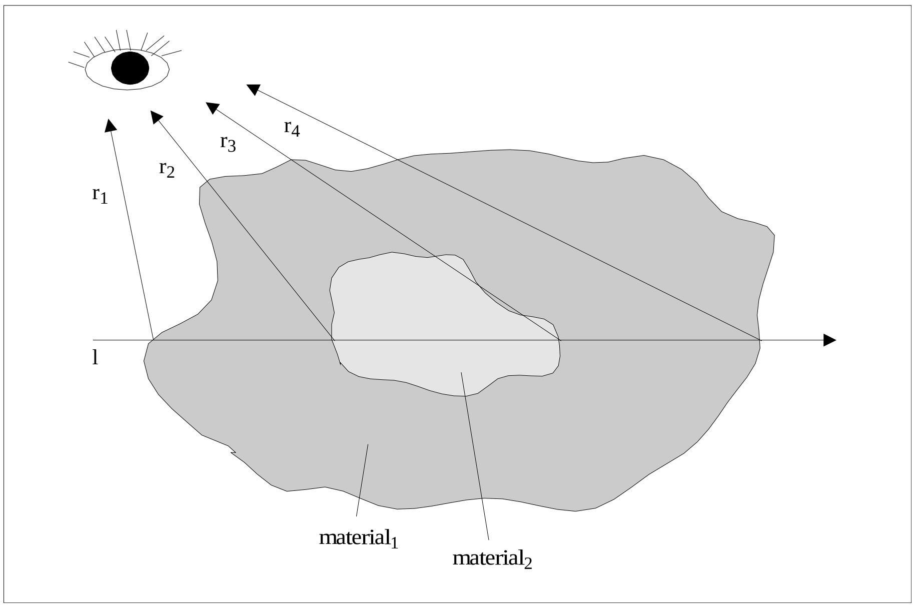


Figure 2.3: Optical model for volume rendering

light are scattered or reflected from the matter's surface, while other parts enter the matter. The scattered parts can be detected by an observer. In figure 2.3, only light reflected in direction to the observer, denoted by r_1 through r_4, is shown, but in reality the light is scattered in all directions. The parts that enter the piece of matter travel on and are absorbed by the matter. Eventually the remaining light hits a surface inside the matter (e.g. a material boundary) and again parts of the light are scattered and parts pass through the surface and travel on.

The final colour and intensity I of a viewing ray which extends from a point a to another point b is given by the *ray casting integral* (see [LCN98, pp. 122-128]):

$$I(a,b) = \int_a^b g(s)\ \exp^{-\int_a^s \tau(x)\mathrm{d}x}\ \mathrm{d}s \tag{2.1}$$

In the above equation, ds is the direction of the viewing ray. The *source term* $g(s)$ is a surrogate for an illumination model such as the Phong or the Gouraud illumination model (see [Pho75] or [Gou71], respectively), which describes how light reflected from surfaces together with the ambient light contributes to the viewing ray. The term $\exp^{-\int_a^s \tau(x)\mathrm{d}x}$ describes the fraction of the initial light still available at a distance s from the starting point a. The extinction coefficient $\tau(x)$ describes how much the light intensity decreases per unit length due to extinction and scattering.

For each (infinitesimally small) point on the viewing ray, the ray casting integral calculates how much light is still available, and applies an illumination model at this point using the remaining light as input. The ray casting integral can be discretised

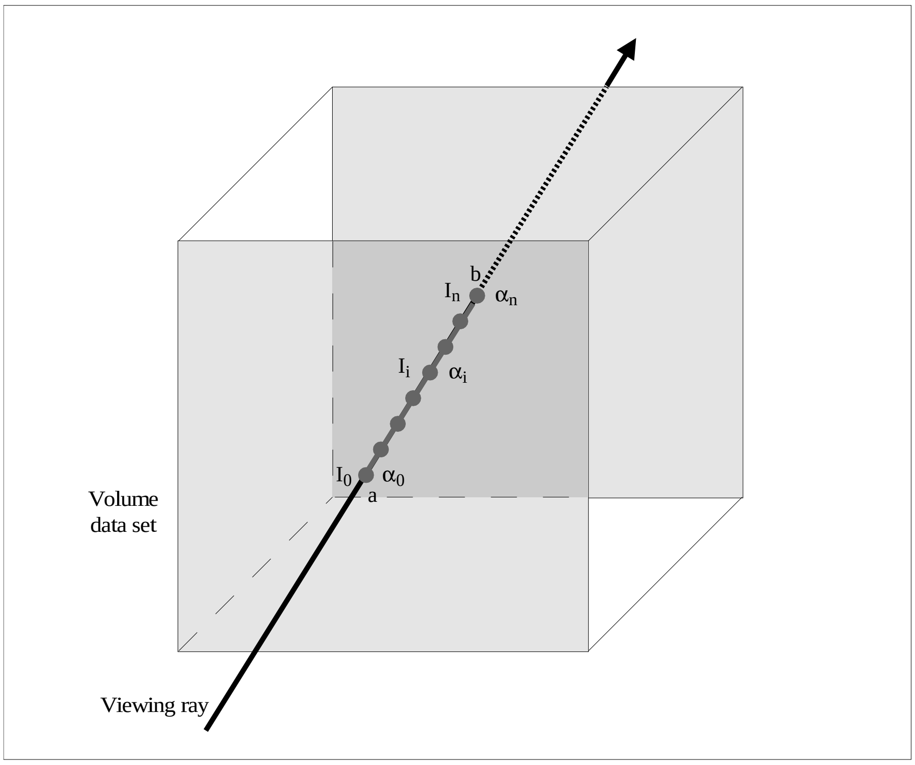


Figure 2.4: Sketch of a viewing ray traversing a volume
(Adapted from [Lev88])

to the following sum:

$$I(a,b) = \sum_{i=0}^{n} I_i \prod_{j=0}^{i-1}(1-\alpha_j) \tag{2.2}$$

$$= I_0 + I_1(1-\alpha_0) + I_2(1-\alpha_0)(1-\alpha_1) + \ldots + I_n(1-\alpha_0)\ldots(1-\alpha_{n-1})$$
$$= I_0 \text{ over } I_1 \text{ over } I_2 \text{ over} \ldots \text{over } I_n$$

As before, $I(a,b)$ denotes the final intensity of a viewing ray going from point a to point b, a sketch of the setup is shown in figure 2.4. The viewing ray is divided in n parts of equal length, each containing one sample point marked with the indices i and j. The absorption at the sample point with index i is described by the sampling point's opacity α_i. For each sample point, both the intensity I_i and the opacity is calculated. The opacity is calculated by the product of the opacities of all sample points encountered so far on the ray. Originally, the *over* operator introduced in [PD84] had been developed to calculate the resulting picture when combining a foreground and a background picture: The over operator combines two pictures A and B such that A over B places the foreground picture A in front of the background picture B.

The intensity I_i at a sample point with index i on the viewing ray is usually defined as the product of the opacity α_i and the colour C_i at this point:

$$I_i = C_i \cdot \alpha_i. \tag{2.3}$$

The sample point's colour C_i and opacity α_i result from the shading and classification operation.

In the following, the basics of the Phong illumination model (see [Pho75]), which is used for implementations described in this thesis, is shortly described. The aim of an illumination model is to simulate the reflection of light on surfaces and to calculate the colour and the intensity of the light perceived by an observer. The Phong illumination model describes reflected light by three categories: ambient light, diffuse reflection and specular reflection. These three categories are combined to yield the colour of a point on the surface. Each surface point is associated with several properties that depend on the materials at this point.

Ambient light describes a non-directional light source that is constantly distributed over the scene and does depend on neither the observer's viewing point nor on the angle between the light's direction and the surface normal. When only ambient light of the colour C_a shines on a surface point with an ambient reflection coefficient k_a and a diffuse colour C_d, this point's colour C is

$$C = C_a k_a C_d. \tag{2.4}$$

Diffuse reflection characterises how light coming from a point light source with colour C_p that radiates uniformly in all directions is diffusely reflected in all directions. This is illustrated in figure 2.5.

The colour of a surface point depends both on the distance between the light source and the surface point and on the angle θ between the light vector $\vec{L}$ and the surface normal vector $\vec{n}$ at this point, but not on the position of the observer. Often it is assumed that the light source is infinitely far away and the distance between the light source and the surface point is not taken into account. Such light sources

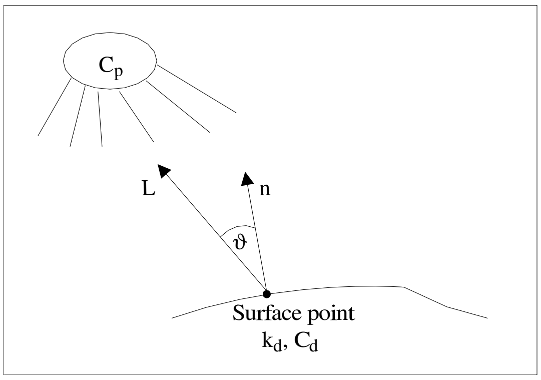


Figure 2.5: Phong illumination model: diffuse reflection

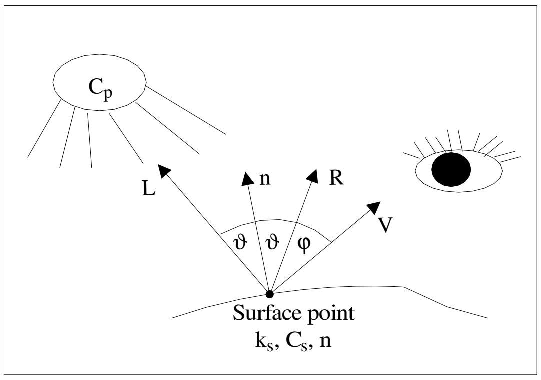


Figure 2.6: Phong illumination model: specular reflection

are called *directional lights*. In this case, the diffuse reflection's contribution to the point's colour is calculated by

$$C = C_p k_d C_d \cos\theta, \tag{2.5}$$

where k_d is the point's diffuse reflection coefficient, which is a property of the material at this point.

Specular reflection simulates the highlights which are caused by light reflected in a certain direction dependent on the angle between the incident light and the surface normal, as shown in figure 2.6. The specular reflection's contribution to the point's colour is calculated by

$$C = C_p k_s C_s (\cos\phi)^n. \tag{2.6}$$

Again, C_p is the colour of the point light source. C_s is the point's specular colour and k_s is the point's specular coefficient, both are properties of the material at this point. The *specular exponent* n simulates the fact, that specular highlights can only be seen when the observer's viewing direction and the direction of the reflected light are nearly the same.

Combining these three contributions to the final colour of the surface point yields:

$$C = C_a k_a C_d \; + \; C_p k_d C_d \cos\theta \; + \; C_p k_s C_s (\cos\phi)^n. \tag{2.7}$$

Chapter 3

Multi-dimensional volumes

In this chapter, first the terms *property*, *feature* and *multi-dimensional volume* are defined for this thesis. Then, the term *multi-dimensional volume* is differentiated from the term *multi-modal volume*, which is sometimes confusingly used in the context of multi-dimensional volumes. Thereafter, sources and applications of multi-dimensional volumes are presented. Finally, the state of the art in visualising such volumes is presented and discussed.

3.1 Definition of terms

3.1.1 Property and feature

The terms *property* and *feature* are used to denote two different things in this thesis: The term *property* denotes a physical quality of an object, e.g. its density or temperature. The term *feature* is used to describe a remarkable characteristic in the quantity of a property at a given position or region, e.g. a region of high density or a hot spot. A position or region where the relation between two properties is remarkable is also called a *feature*, e.g. a region where both density and temperature are high.

3.1.2 Multi-dimensional volume

In the context of this thesis, the term *multi-dimensional volume* is used for a data set that describes not only one but multiple properties of an object. Therefore, the term multi-dimensional refers not to the spatial dimensions but to object property dimensions. In contrast, in this thesis, the abbreviations *2D* and *3D* (two- and three-dimensional in space, respectively) always refer to the spatial dimensions of an object.

In Cartesian coordinates (x, y, z), a scalar property p (like density, temperature or pressure) of an object can be described by a function $p = f(x, y, z)$. The volumetric data set of this property consists of a set S of samples (x, y, z, p) representing either the value p of this property at the sampling point (x, y, z) or the mean value of this property in the volume element between the eight sampling points (x, y, z), $(x, y, z+1)$, $(x, y+1, z)$, $(x, y+1, z+1)$, $(x+1, y, z)$, $(x+1, y, z+1)$, $(x+1, y+1, z)$

and $(x+1, y+1, z+1)$, corresponding to the two possible definitions of the term *voxel* in section 2.2.

The term "n-dimensional volume" denotes a volumetric data-set consisting of a set S of samples $(x, y, z, p_1 \ldots p_n)$ representing not only one scalar property p_1 but n properties $p_1 \ldots p_n$, which are described by n functions $p_i = f_i(x, y, z),\ i = [1 \ldots n]$.

Although it is possible to think of non-scalar properties of an object (like the velocity of a fluid, which clearly is a vector), these are not discussed in this thesis.

3.1.3 Multi-dimensional vs. multi-modal volumes

The frequently used term *multi-modal* volume data is closely related to the term multi-dimensional volume data, but they denote two different things. Two or more data sets that describe the same object but are acquired using different devices (or *modalities*) are called multi-modal. In the notation of this thesis, two or more data sets describing different properties of the same object are called multi-dimensional.

It is perfectly imaginable (and in fact quite common) to have a single-dimensional but multi-modal data set, e.g. a CT and an MRI data set of one object (thus multi-modal), both describing the matter distribution of the object (thus only one property dimension).

3.2 Origins and applications of multi-dimensional volumes

There are two main origins of multi-dimensional data sets: simulation and measurement.

A typical example for simulation is a fluid simulation yielding data sets containing information about pressure and velocity of particles. An advantage of simulation is that it can be set up such that the different properties' grids exactly match. The registration precision is then only limited by the simulation's error.

A good example for measurement is medical imaging. Two common imaging facilities are CT and MRI. CT data sets represent the object's electron density (see e.g. [Vog97], p. 641) which can be mapped to a matter distribution. MRI data sets usually represent the object's water distribution (the nuclear magnetic resonance signal from humans mainly comes from water protons, see e.g. [CMP99]), which can also be mapped to a matter distribution. Both techniques deliver structural information of the object. Additionally, by using *functional* MRI (fMRI), it is possible to obtain functional information about an organism, like for example cerebral activity. Other means of taking functional information are, for example, *electroencephalography* (EEG), *magneto-encephalography* (MEG), *positron emission tomography* (PET) and *single photon emission computed tomography* (SPECT). Furthermore, both structural and functional information can be gathered using *ultra sound* (US).

Compared to data from simulations, data sets of physically measured data are more difficult to handle as the data sets for two different properties usually have to be mapped to a rectangular grid (*interpolation*). Afterwards, they must be mapped to a common grid (*registration*), which is a difficult task that is prone to errors (see [SLPB96]). Additionally, the resolutions of the data sets representing the different properties of an object often differ, for example CT and MRI data sets usually have a much higher resolution than PET data sets of the same object.

An example for the use of multi-dimensional data in medicine is the research and diagnosis of epilepsy (see [RDC96]). The physician is interested in knowing exactly at which location in the brain the electrical currents are pathologically high and where the current's source is. In this example, one property dimension is the matter distribution of the brain acquired by CT and/ or MRI. The other property dimension is the electrical current distribution acquired by EEG or MEG.

For some applications, two data sets of the same object are combined to form a new data set (see e.g. [NBC$^+$00]), which is then visualised. This process is called *fusion*. A common problem in medical imaging is the fusion of high quality static data acquired before an intervention with low quality data acquired during intervention, for example in image guided neurosurgery (see e.g. [WTT$^+$02], [GNK$^+$01], [JFS$^+$00]) or in radiotherapy (see e.g. [RKC$^+$01]).

Additionaly, some applications make use of image sequences: For example, to research the relation between coronary artery disease and myocardial dysfunction, it is of high interest to visualise information about both the myocordial function (obtained from MRI) and the coronary vasulature (obtained from computed tomographic angiography data) as a sequence over the time of a heart beat (see [SPHW02]).

Another, more general example is the heat distribution in an object where the observer is interested in knowing what the temperature at a given location is. Again, one property dimension is given by the matter distribution, the other is given by the heat distribution.

3.3 State of the art

This section describes the current state of the art in the visualisation of multi-dimensional volumes. The techniques presented describe available methods, not particular programs. The critical review provides the basis for the development of the dependent rendering algorithm in chapter 4.

The methods commonly used to display multi-dimensional volumes are presented by starting with the simple and proceeding to the more elaborate approaches. The visualisation techniques can roughly be classified into three groups: Pure 2D techniques (see section 3.3.1), hybrid techniques combining 2D and 3D elements (see section 3.3.2) and 3D techniques (see section 3.3.3).

While discussing the approaches, special attention is given to the following criteria:

- **Localisation of features:** How precisely can features found in the renderings be located, both relative to a coordinate system and relative to landmark features in both properties?
- **3D impression:** Is it easy for the user to mentally reconstruct the 3D structure of the object from the renderings?
- **Arbitrary properties:** Is the technique restricted to an application domain or can it be used for data sets representing other properties?
- **Exploration:** Is it possible to explore data sets of unknown objects or is a-priori knowledge of the structure of the object required?

The section concludes with an evaluation of the presented techniques.

The need for displaying multi-dimensional volumes is high in the field of biomedical imaging, especially in the functional imaging area where both anatomical and functional data of humans are measured and visualised. Therefore the examples for the techniques described in this section all come from biomedical imaging applications.

It is usually necessary to visualise multiple properties jointly, where one property (usually the density) serves as spatial reference for the others. This is explained in more depth in section 4.1. For all examples in this section, the first property is given by the anatomy of the human brain, acquired by either CT or MRI. The second property is some kind of functional data.

3.3.1 2D approaches

3.3.1.1 2D Image gallery

The simplest approach in visualising multiple properties is using a *2D image gallery*, which is a static set of 2D slices of the data set where interesting features are highlighted. In the top two rows of figure 3.1, two different variants of this technique are shown: In the upper row, the slices from the anatomical data set are fused with the corresponding slice from the functional data set. The anatomical data is displayed in grey levels and the functional data in shades of red.

There are several techniques for fusing 2D images of anatomical and functional data, here a pixel interleaving method as described in [RSA+94] is used. Displaying just parallel slices of the data sets is very similar to the conventional presentation of X-ray images on a light board.

In the second row, Talairach coordinates (see [TT88]) are displayed instead of the anatomical data, and a maximum intensity projection of the functional data is overlaid. By integrating the functional information with Talairach coordinates, the user can localise features in one data set and transfer this localisation to other brains in terms of the Talairach coordinates. The use of Talairach coordinates is special to brain imaging, therefore it will not be considered further.

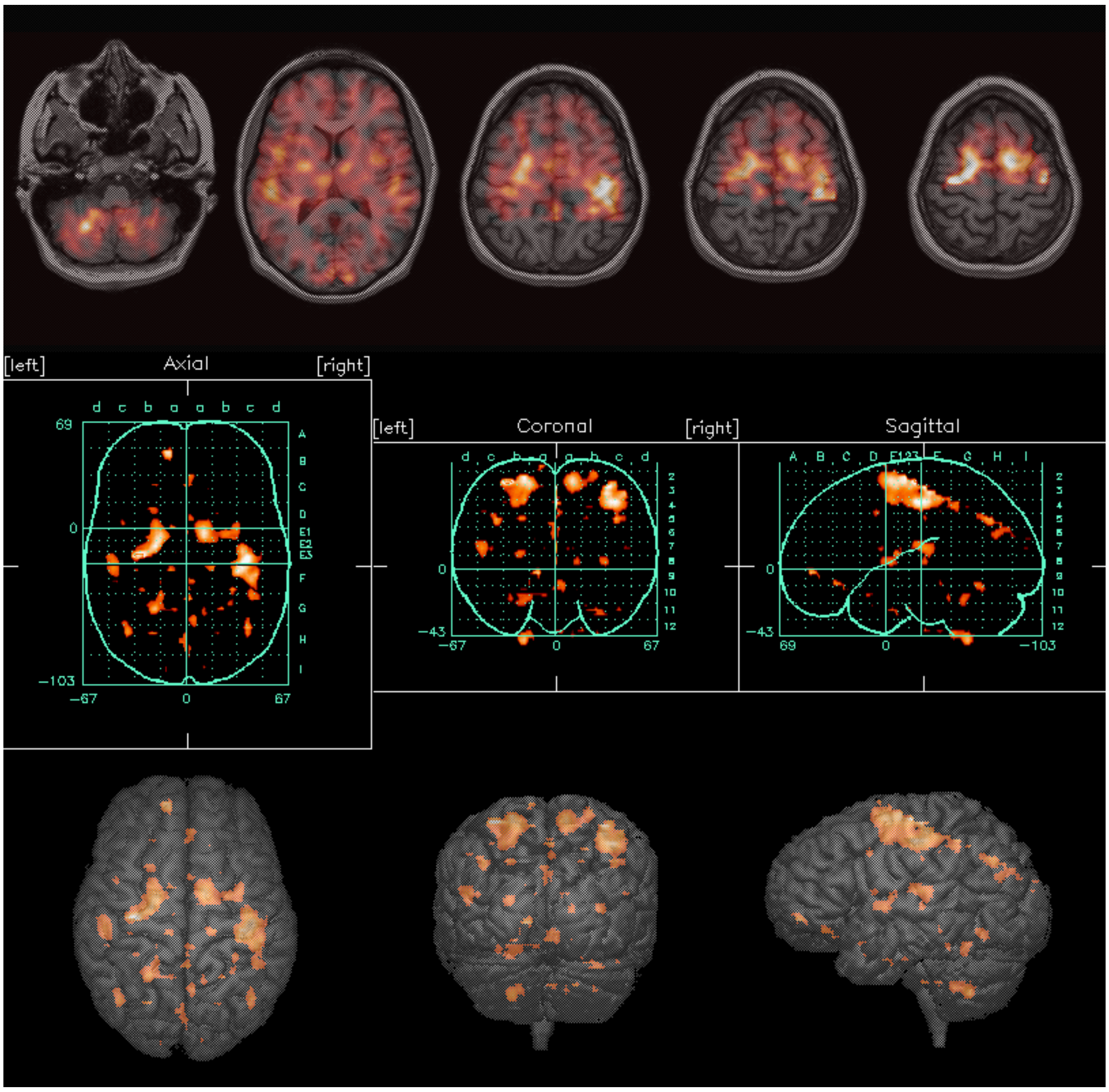


Figure 3.1: Image galleries for anatomical and functional data of the human brain. Top row: Fused slices of functional and anatomical data. Middle row: Functional data integrated with Talairach coordinates. Bottom row: Functional data projected onto surface rendering of anatomical data. Source: [RLF+98]

But the concept of a specialised coordinate system which allows the mapping of points in one object to points in another similar object could also be applied in other fields. The bottom row of the figure contains surface renderings, which are described in section 3.3.3.1.

3.3.1.2 Interactive orthogonal slice viewers for fused images

Orthogonal slice viewers for fused images as depicted in figure 3.2 allow the user to interactively browse through different slices of the brain. The technique used to fuse the two properties for this image is described in more detail in section 5.2.3.

To help the user navigate through the slices, often markers with the position of the corresponding other two slices are displayed on each slice. The advantage over the static image gallery approach is that the user can interactively choose different slices and thus better explore a data set.

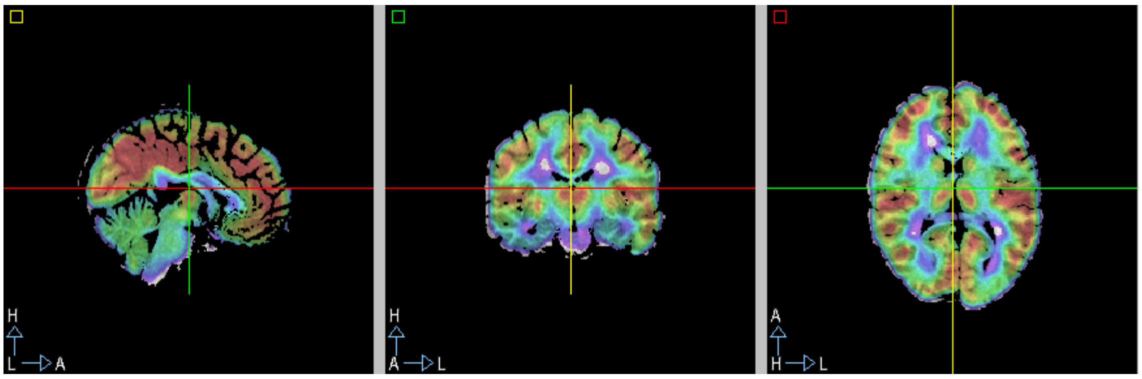


Figure 3.2: Orthogonal slice viewer for fused images.
Screen shot of the program described in section 5.2

3.3.1.3 Evaluation of 2D techniques

In general, the technique for displaying fused 2D slices is computationally inexpensive. This technique is not restricted to medical imaging in can be used for arbitrary properties.

The advantage of the 2D slice images is their precision, which is due to their high spatial resolution. Therefore, features can be localised precisely. However, this technique makes it hard for the observer to mentally reconstruct the 3D structures of the object. Another drawback is that only little context information is displayed, i.e. when the observer examines one slice, he receives no information on the adjacent slices. This makes it very hard to overview the object as a whole and difficult to explore data sets of objects of unknown structure.

Usually, only one property is displayed in its full extension, while the other property is only displayed in parts. This is especially true for image fusion techniques where one property is opaquely overlaid over another property. To visualise both properties in their full extension, the slices from both images must be fused non-opaquely, a process which often causes loss of information in at least one property (see e.g. [RSA$^+$94]).

3.3.2 Hybrid 2D/3D approaches

To improve the understanding of the 3D structure of the object, the 2D approaches have been extended by 3D elements in several ways.

3.3.2.1 The *corner cube* approach

In the *corner cube* technique described by [RLF+98] and shown in figure 3.3, a cubical scene is created which is bounded by displaying three orthogonal landmark slices from the anatomical data. In the middle of the scene, the functional data is rendered and additionally projected onto the anatomical slices. The functional information is

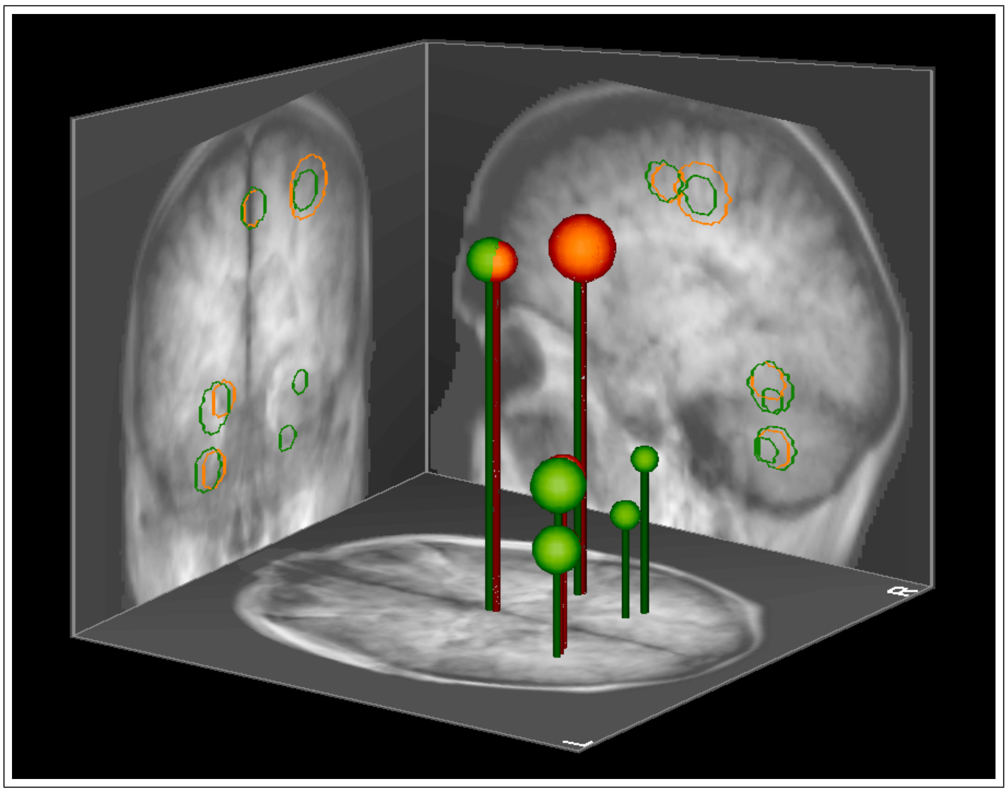


Figure 3.3: Corner cube approach.
Source:[RLF+98]

either rendered as special glyphs ("A glyph is a graphical object designed to convey multiple data values" [War99, p. 152].) or as a a surface rendering of the previously extracted regions of activation.

3.3.2.2 The *slicer* approach

In the *slicer* technique described in [GNK+99] and shown in figure 3.4, a scene is created by displaying three orthogonal slices through the anatomical dataset.

Further applications of this approach are described in [GNK+01] and [TOW+03]. The position of each slice can be varied, and the whole scene can be rotated, thus allowing a better understanding of the 3D structure of the data set. The functional data is either displayed by fusing the slice images of both data sets or by displaying glyphs or surface renderings of the functional data, like in the *corner cube* approach.

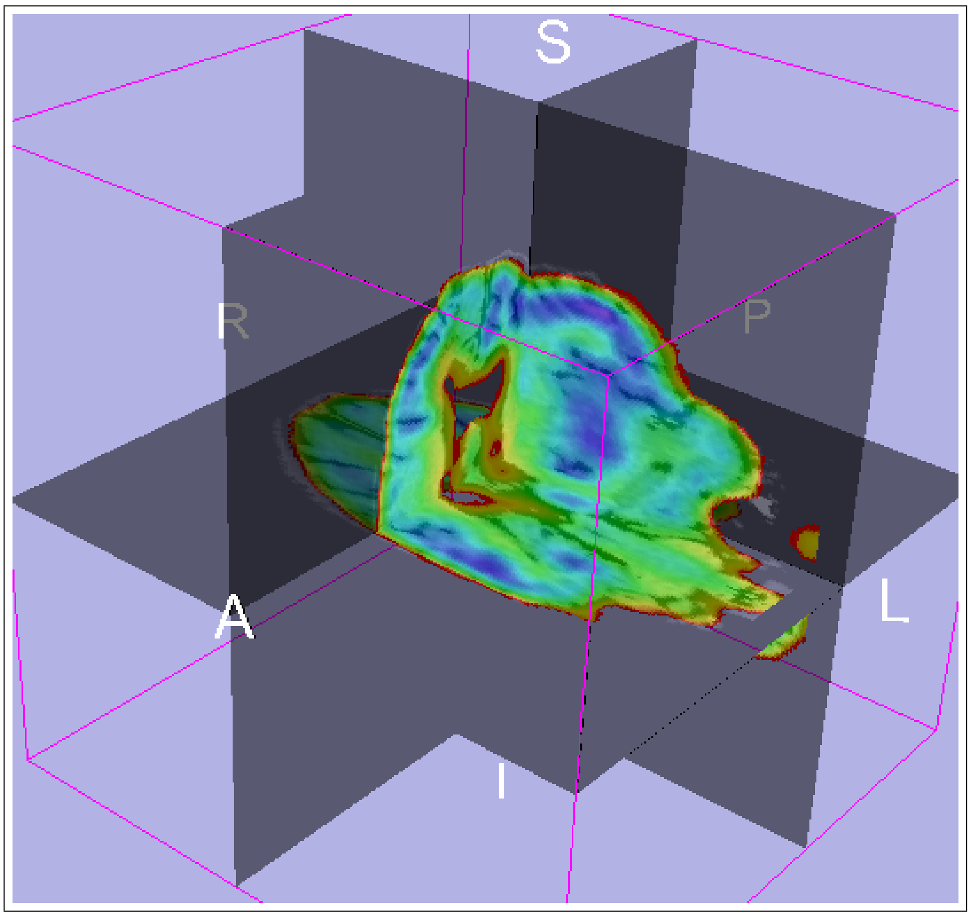


Figure 3.4: Slicer approach.
Image created using the slicer program [MS]

3.3.2.3 Evaluation of hybrid 2D/3D techniques

Images generated by the hybrid techniques are generally less precise than images of 2D slices, because the 2D slices used in the hybrid approaches are distorted due to the viewing transformation applied to them. Additionally, parts of one slice can be occluded by the other two slices and by the rendering of glyphs or surfaces of regions of the second property. The overall 3D impression is a little better than with the pure 2D approaches, because the slices are displayed in the correct geometrical relation to one another. This helps the user to reconstruct the 3D structure of the object. The techniques are not restricted to medical imaging.

A severe problem is the extension of the second property: If rendered as glyphs or surfaces of regions, the regions must be small and not too numerous, otherwise they would fill the whole rendered image and the renderings of the first properties would be hidden. This would make it impossible to examine the first property or localise features found in the second property. This problem can be avoided by using the *slicer* approach with fused image slice using a non-opaque fusion technique. To create surfaces or glyphs for regions of the second property, these regions have to be extracted before rendering, which makes these techniques useless for exploring unknown data sets.

3.3.3 3D approaches

3.3.3.1 Surface rendering

A very common approach (see e.g. [RDC96] or [Dim95]) is to visualise a surface in the reference structure, e.g. the surface of the brain, and project the other property onto this surface, as in figure 3.5 or in the bottom row of figure 3.1. Although this approach is computationally quite inexpensive, the images generated by surface rendering give no or little information about the inner parts of the object, therefore this technique is almost useless for investigating the inner structure of the objects.

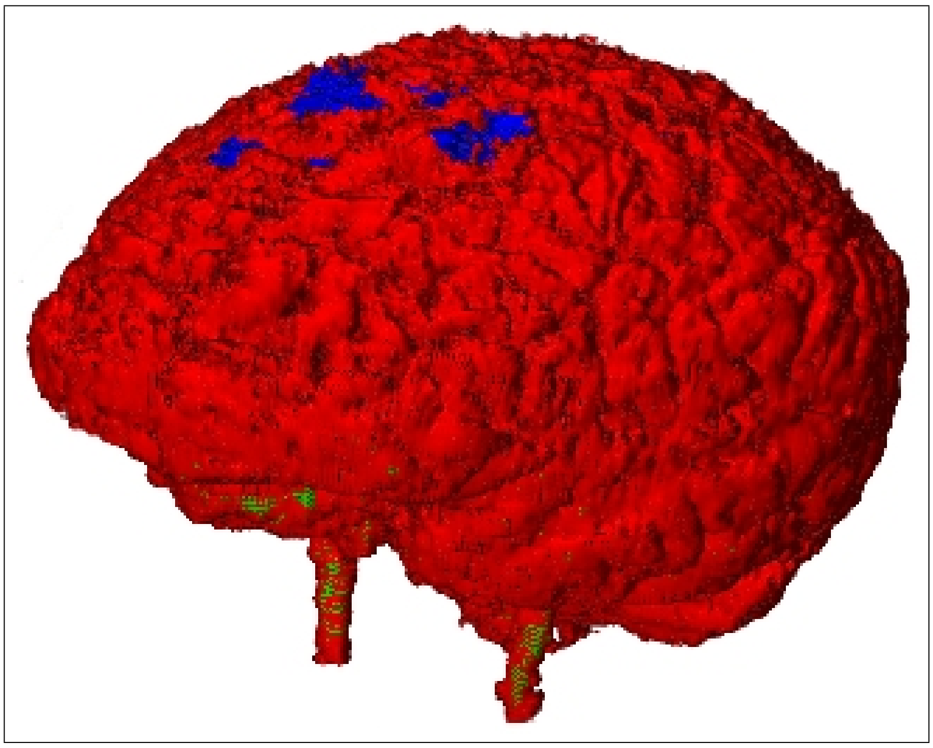


Figure 3.5: Surface rendering approach.
Source:[KDG99]

3.3.3.2 Direct volume rendering (DVR)

DVR has already been presented in detail in section 2.3.2. For one-dimensional volumes, this technique has proven to provide good hints on the location of features in objects, therefore it seems reasonable to enhance and use it for multi-dimensional volumes, too. Besides being computationally expensive, nearly all currently existing volume rendering systems have no support for multiple property dimensions.

One drawback of volume rendering is that it makes use of transparencies and therefore some precision may be lost [HP96]. Two extensions to conventional volume rendering systems are presented in the following paragraphs.

DVR using transfer functions To visualise activated regions in the human brain, this method was introduced in [KDG99]. Besides the matter distribution as reference data, a second property has been highlighted using transfer functions as shown in figure 3.6. In the figure, red and green denote different materials in the brain while the blue spots indicate regions of high activity. This method seems quite suitable for a second property that is only defined in certain areas inside the object and that only has a few distinct values (in the article, only two states are mentioned: activated or not activated). For a second property defined over the whole object, this method appears not appropriate because highlighted regions by themselves contain no spatial information. This approach requires the exact locations of the second property to be known in advance in order to create a transfer function volume for the highlights, therefore this technique might be useful for displaying an already examined data set but not to explore an unknown data set.

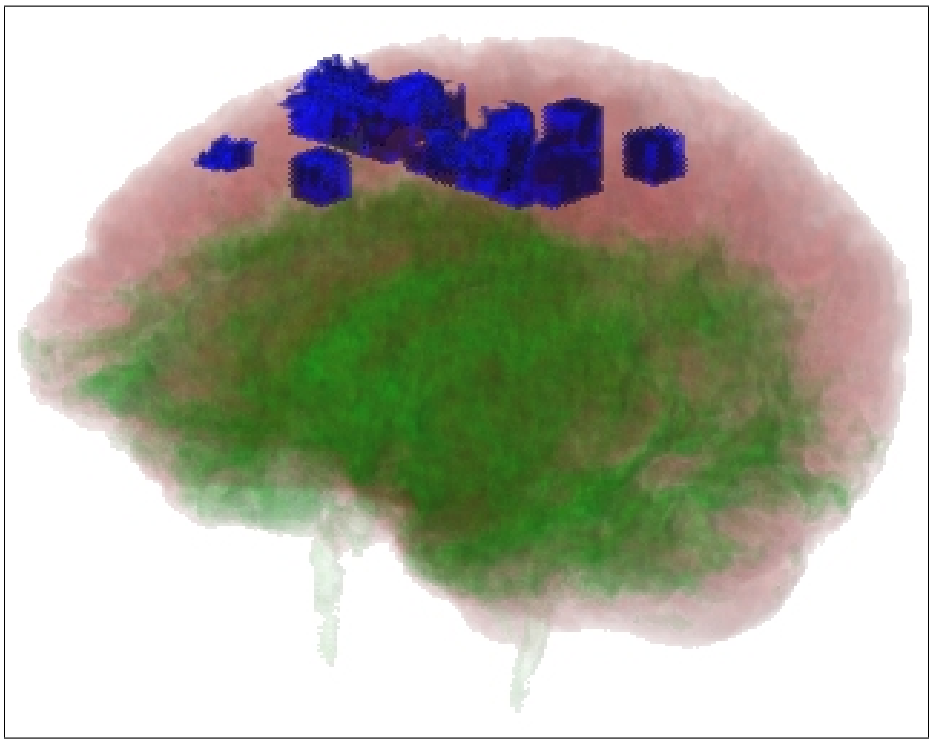


Figure 3.6: Volume rendering using transfer functions.
Source:[KDG99]

Multiple views The *DVR using transfer functions* technique has been extended into the *multiple views* technique to improve the understanding of spatial relationships by placing additional projection planes "behind" the volume as in figure 3.7 (see e.g. [KDG99]). This technique is an enhancement to the *corner cube* technique described in section 3.3.2. A further extension of the *multiple views* approach is the *magic mirrors* approach that uses a different transfer function for each of the additional projection planes to improve the spatial localisation of the highlighted regions relative to the anatomical volume. Besides improving the 3D impression a little bit, the same advantages and shortcomings found for the *DVR using transfer functions* approach apply here.

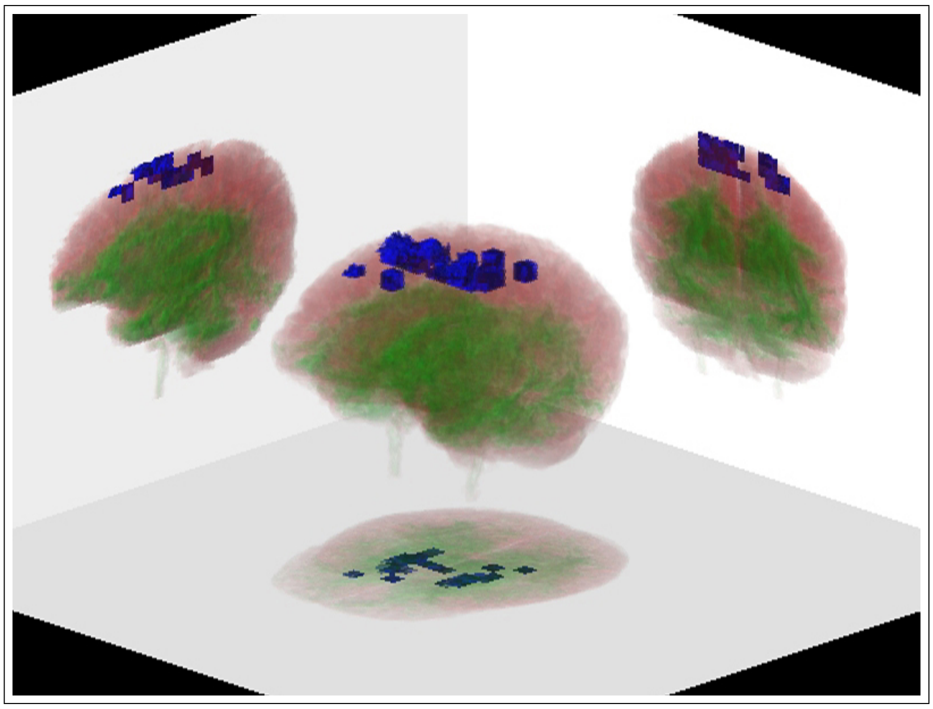
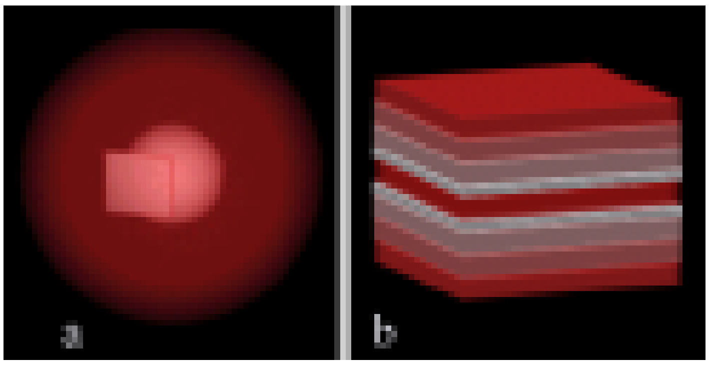


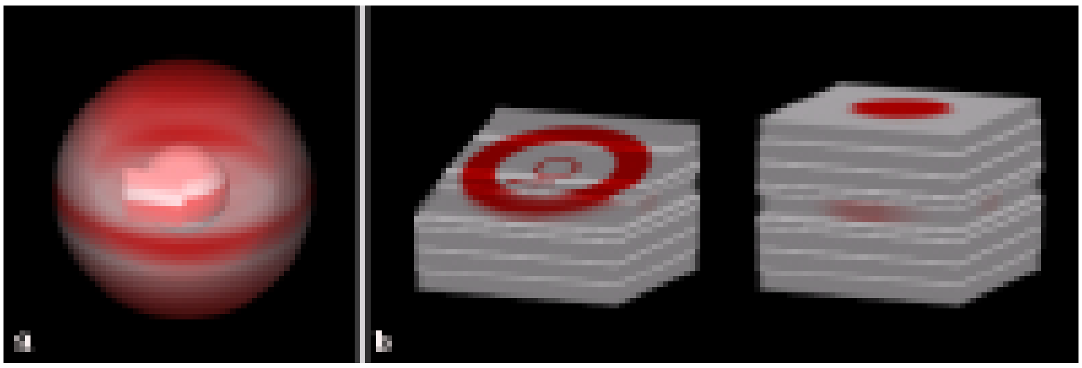
Figure 3.7: Volume rendering using multiple views.
Source:[KDG99]

DVR using data intermixing In the basic *data intermixing* approach as described by [MFOF02], the two volumes are traversed similar to Levoy's classic approach [Lev90]. For each pixel in the rendered image, two rays are cast simultaneously, one into each volume. During composition, the opacities are calculated from one volume, and the colours are calculated from the other volume, an example is shown in figure 3.8. Although this approach allows both properties to have the same extensions and is not specialized for a particular application domain, it has a fundamental drawback: There are no colour contributions from regions, where the opacity is low. This implies that, if regions in the volume that determines the opacity are hidden by the opacity transfer function, no information from the other volume will be shown for this region. In general, volume rendered images can provide the user with a very good 3D impression of the structure of the object. The techniques are not restricted to any special combination of properties and can potentially be used for arbitrary properties.

As already stated for the other approaches, the extension of the second property is a problem: The second volume must not be defined over the whole extension of the first one but merely "scattered" over some small regions. If the second property was defined over the whole extension of the first property, its "highlighting" in the rendered images would overlay the information from the first property and all structural information would be lost. Additionally, the second property is limited to a small dynamic range: For all images shown in this section except those in figure



Test volumes rendered separately: (a) Spheres and (b) cube



Test volumes rendered using intermixing (a) The sphere determine the opacity, and the cube determines the colour. (b) The spheres determine the colour and the cube determines the opacity. In the left image of (b), an additional cutting plane is used to look inside the object.

Figure 3.8: Volume rendering using data intermixing.
Source:[MFOF02]

3.8, the second property is represented by only two possible states (activated or not).

Although there are 3D approaches to visualise a second property with a high dynamic range (see e.g. [RDC96]), these approaches use surface rendering which is considered inadequate for exploring unknown data sets as surface renderings contain no information on the parts behind the surface.

3.3.4 Summary of the current techniques

The 2D techniques, especially the interactive orthogonal slice viewer technique, can provide the observer with very precise information about the localisation of features in both properties, but it is hard for the user to mentally reconstruct the spatial structure of the object from the 2D slices. Additionally, the exploration of unknown data sets is difficult because the slices do not show context information from the directions orthogonal to the displayed slice. Thus the 2D techniques, especially the interactive orthogonal slice viewers, are very useful for precisely examining features that have already been located, but they are unsatisfactory when the observer wishes to explore an unknown object.

The advantage of the 3D volume rendering techniques is that they show the 3D structure of the object very well. The user needs no or little a-priori knowledge about the structure of the object to localise features in both properties, both relative to other features in the same property (e.g. “feature A in the first property is located in front of feature B in the same property”) and relative to features in the other property (e.g. “feature A in the first property is located in the same location as feature C in the second property”).

The hybrid approaches are not satisfactory: The precision of the images is worse than that of the 2D techniques they are based on, and the 3D impression is worse than that of volume rendered images.

All these approaches have a general limitation in common: The second property is not regarded as a volume on its own but merely as an“accessory” to the first one, which is often only defined in some small regions. The only technique able to handle two volumes where the second property is defined over the whole extension of the first property are the 2D slice viewer techniques with an appropriate no-opaque fusion and techniques that directly derive from this like the *slicer* approach.

However, there are many problems in science and engineering where one is interested in a second property that is extended over the whole range of the first one (e.g. the heat distribution of an object or the pressure distribution).

Therefore, there is a need to develop a technique that is as powerful as volume rendering for the localisation of features and the 3D impression, but that allows the use of a second property that has the same extension as the first property. Such a technique is developed in the next chapter.

Chapter 4

The new dependent rendering algorithm

In this section, the problem of visualising multiple properties of an object is first re-formulated and then approached in a step-by-step fashion. In this context, the notation of *reference* and *dependent* structures is introduced. Then, the requirements for an algorithm for visualising two-dimensional volumes are identified, which finally leads to the development of the new *dependent rendering* algorithm. The developed algorithm will be evaluated by three implementations described in chapter 5.

4.1 Reference and dependent structures

To begin with, it is briefly recapitulated what the starting point to the development of the new algorithm is and what the new algorithm is expected to achieve. The starting point is a data set describing two properties (say: density and temperature) of an object. The observer is interested to see how these two properties are distributed in space, i.e. what kind of material and what temperature is present at a given point inside the object. Moreover, it would also be important to know how these two properties are related to one another.

Visualising only the object's density using volume rendering would be quite straightforward, but visualising only the temperature incorporates a problem: Although the overall characteristics of the temperature distribution can be recognised, one critical information is missing (see figure 5.3 a): It is not possible to precisely localise a given temperature inside the object because there is no reference to the object's "real" structure, i.e. the density distribution.

This leads to the conclusion that it is necessary to visualise both properties together, with the first property providing a precise localisation of features of the second property.

In the following, the property that serves as the reference is called a *reference structure* and the other properties *dependent structures*. This does not imply a mathematical dependency between these two properties, although this can be easily imagined (e.g. an insulated object with a heat source somewhere inside: The heat distribution $t = f_T(x, y, z)$ directly depends on the matter distribution $d = f_D(x, y, z)$).

4.2 Requirements for an improved algorithm for the visualisation of two-dimensional volumes

In this section, the requirements for an improved algorithm for visualising two-dimensional volumes are derived. These requirements relate to the core algorithm only, not to a complete system, because the focus of this research is on the development of an algorithm, not a system. It should be possible to implement the algorithm and plug it into a customisable 3D volume visualisation system which provides common basic features including editing of rendering parameters (like transfer functions) and arbitrary rotation of the data set. Therefore these basic features are not modeled here. That it is sensible to focus the requirements definition on the core algorithm only is demonstrated in section 5.2, where a successful implementation of the algorithm as a module for an existing visualisation platform is described.

According to [Hru97, p. 26], a requirement is *"a statement in the format of the chosen method of requirements representation, describing one ore more related characteristics that the system is required to have"*. So, requirements describe *what* a system is supposed to do, but not *how*.

To develop the requirements, first use cases for the visualisation of two-dimensional volumes are developed. Use cases are a fundamental element of object-oriented analysis. For a treatment of object-oriented analysis, see e.g. [Oes01]. For each use case, the functional requirements are derived. Additionally, non-functional requirements are added where neccessary.

As mentioned in the introduction, volume visualisation is *"a method for extracting meaningful information from volumetric data sets through the use of interactive graphics and imaging"* [KCY93, p. 51]. Based on this definition, a hypothetical use case scenario is created which consists of the examination of a two-dimensional volume by a user. This scenario is depicted in figure 4.1, using the Unified Modeling Language (UML). It is as follows:

The process of examining the data set is an abstract use case. Abstract use cases describe in a general way the characteristics several concrete use cases share. This use case can be further sub-divided:

- **Localise finding:** While examining the data set, the user finds an interesting feature in one or both of the data sets. To further examine and localise his finding, he has two possibilities:
  - **Localise in absolute coordinates:** The user wants to localise the finding in absolute coordinates, i.e. in terms of the reference property.
  - **Localise relative to other property** The user wants to localise the finding relative to the other property (e.g. "it is hot on a surface").
- **Examine relations between properties:** The user is interested in a broader overview on the relations between the two properties (e.g. "the upper left part of the object is colder than the rest").

To derive and formulate the requirements for the algorithm based on these use cases, a well-defined requirements analysis technique shall be applied. Three types

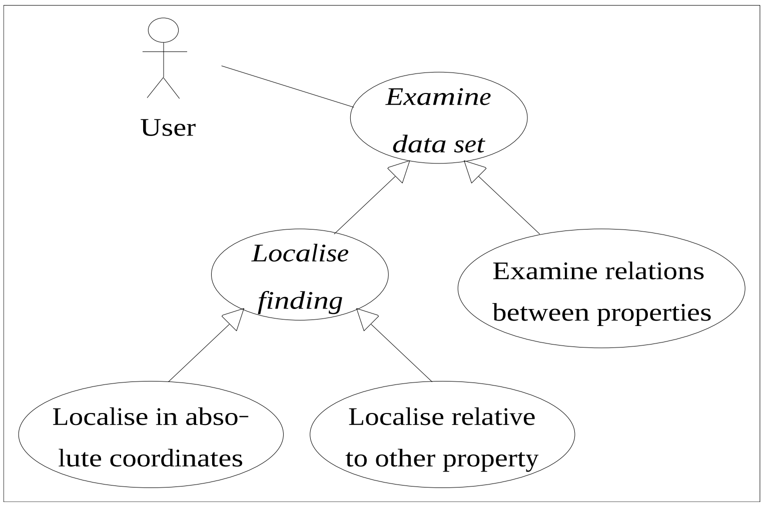

User
Examine data set
Localise finding
Examine relations between properties
Localise in absolute coordinates
Localise relative to other property

Figure 4.1: Use case diagram.

of approaches have been considered for use in the context of this thesis: The *Volere* approach [RR99], *Agile* techniques [PEM03], and *Quality Function Deployment (QFD)* [MA94]. The goal was to find a technique that allows the structured analysis of requirements, is appropriate to the anticipated relatively small number of requirements for the algorithm, and does not require direct involvement of customers, because customers were not available.

The Agile techniques were not considered further because they do not document requirements in a structured manner but rather rely on direct and continuous customer involvement ("joint application development").

The rationale of QFD is to develop or improve products to better meet customer needs. For how to adapt QFD for software engineering, see [HSM99]. While providing a structured and documented approach, QFD is not applied here because it is focussed on customer input, which was not available.

The *Volere* approach provides a structured documentation of the requirements using a standard template ("requirement shell"). Due to its simple structure, it seemed applicable for this project. Also, it does not require direct user participation. Therefore, an adapted "*Volere* requirement shell" is used to define the requirements for each use case. Each requirement is identified by a unique number and a name. The relevant use case(s) and the requirements type (here: functional or non-functional) are noted. Additionally, the requirement shell contains the following fields:

Description	A short description of the requirement.
Rationale	Gives a justification for the requirement.
Source	Who raised the requirement (e.g. from literature).
Fit criterion	Describes how the requirement can be tested.
Dependencies	Lists other requirements that have dependencies on this one.
Conflicts	Lists other requirements with which this one can possibly conflict.

The origininal Volere requirement shell also contains fields for the customer satisfaction or dissatisfaction in case the requirement is fulfilled or not fulfilled, respectively. These fields are left out here because customers were not available, but if the algorithm is implemented for a broader user group, they can be added and used in an evaluation.

To derive the requirements for the algorithm, several types of sources have been used: The first source is represented by the research goal for this work, i.e. to develop a visualisation technique for two-dimensional volumes that is as generally applicable as possible (see section 1.1). Moreover, the shortcomings of other techniques, which have been identified in section 3.3, are translated into requirements. Also, known problems discussed in the literature are taken into account. Furthermore, the use case scenario described above is used as a source. In addition, some technical requirements have been defined by the author.

The requirements are as follows:

1.0 Abstract use case "Examine data set": This abstract use case describes the general functionality: Visualise the two-dimensional volume such that a user can examine it. All use cases that are derived from this one inherit the requirements defined for this use case. Several non-functional requirements that are special to two-dimensional volumes are associated to this use case:

1.1 3D impression: An important goal of all hybrid 2D/3D approaches and the 3D approaches presented in section 3 was to support the user in building a mental 3D model of the object under study. It is desirable that 3D impression obtained from the renderings of the two-dimensional volume is comparable to that of standard volume rendering. Table 4.1 provides the complete requirement specification:

Requirement	**# 1.1**
Use case	Examine data set
Name	3D impression
Type	Non-functional
Description	The rendering must provide good hints on the 3D structure of the object.
Rationale	To improve the spatial understanding of the object under study, the user must be supported at building a mental 3D model.
Source	Obvious problem that is also stated in the literature, e.g. [KDG99], [RLF+98].
Fit criterion	Visual verification, ideally requires a user test.
Dependencies	-
Conflicts	May conflict with requirement # 2.0

Table 4.1: Requirement "3D impression"

1.2 Exploration: The exploration of unknown content must be supported (see table 4.2):

Requirement	**# 1.2**
Use case	Examine data set
Name	Exploration of data set
Type	Non-functional
Description	The user must be able to explore possibly unknown data sets.
Rationale	As the algorithm should not be specialised to a particular application domain, it must be possible to apply it to many different data sets.
Source	Research goal “as generally applicable as possible”.
Fit criterion	-
Dependencies	Directly depends on requirements # 1.2.1 and # 1.3
Conflicts	-

Table 4.2: Requirement “Exploration of data sets”

1.2.1 No need for segmentation The algorithm should be suitable for unsegmented data, as segmentation requires a-priori knowledge about the structure of the data set (see table 4.3):

Requirement	**# 1.2.1**
Use case	Examine data set
Name	No need for segmentation
Type	Non-functional
Description	The algorithm must not rely on segmented data, but it should work with segmented data if available.
Rationale	The a-priori knowledge needed to segment a data set is specific to particular application domains.
Source	Research goal “as generally applicable as possible”.
Fit criterion	Must be achieved by design.
Dependencies	-
Conflicts	-

Table 4.3: Requirement “No segmentation neccessary”

1.3 Interactivity: To efficiently examine a data set, the algorithm must allow interactive rendering rates, as specified in table 4.4.

Requirement	**# 1.3**
Use case	Examine data set
Name	Interactivity
Type	Non-functional
Description	The algorithm must allow for interactive use.
Rationale	For efficient examination, especially of unknown data sets, the user must be able to interactively view the data sets from different angles and with different rendering parameters.
Source	Typical requirement for a volume visualisation technique, see e.g. [KCY93].
Fit criterion	Time for rendering a frame must allow interactive use.
Dependencies	Requirement # 1.2 depends on this.
Conflicts	High quality renderings may take long to render.

Table 4.4: Requirement “Interactivity”

1.4 Any property can be an entire volume: The algorithm must allow both properties to be an entire volume, having the same extensions and comparable dynamic ranges (see table 4.5).

Requirement	**# 1.4**
Use case	Examine data set
Name	Any property can be an entire volume
Type	Non-functional
Description	Both properties must be regarded as an entire volume.
Rationale	It is possible that both properties have the same extension, so the algorithm must support this.
Source	In section 3.3.4, it has been shown that it is a fundamental conceptual limitation of many existing techniques to assume that the dependent property has a much smaller extension than the reference property.
Fit criterion	Must be achieved by design.
Dependencies	-
Conflicts	-

Table 4.5: Requirement “Any property can be an entire volume”

1.5 Arbitrary properties: As the algorithm should not be restricted to a particular application domain, it must not be restricted to certain kinds of properties (see table 4.6):

Requirement	**# 1.5**
Use case	Examine data set
Name	Arbitrary properties
Type	Non-functional
Description	The algorithm must not be restricted to data sets of specific properties.
Rationale	The algorithm should be as generally applicable as possible, therefore it must not be a-priori specialised to specific properties.
Source	Research goal “applicability to data sets of arbitrary properties”.
Fit criterion	Must be achieved by design.
Dependencies	-
Conflicts	-

Table 4.6: Requirement “Arbitrary properties”

1.6 Different sizes and resolutions: The algorithm must be able to handle data sets of different sizes and resolutions (as specified in table 4.7):

Requirement	**# 1.6**
Use case	Examine data set
Name	Data sets of different sizes and resolutions
Type	Non-functional
Description	The algorithm must allow the use of datasets of different sizes and resolutions.
Rationale	Data sets from different modalities often have different sizes and resolutions.
Source	Own requirement.
Fit criterion	Must be achieved by design.
Dependencies	-
Conflicts	-

Table 4.7: Requirement “Data sets of different sizes and resolutions”

2.0 Relations between properties (use cases “Examine relations between properties” and “Localise feature relative to other property”): The user wants to recognise relations between properties and to determine the values of one property at the location of a feature in the other property (see table 4.8):

Requirement	**# 2.0**
Use case	Localise a feature in one property relative to the other property. Examine relations between properties
Name	Relations between properties
Type	Functional
Description	The algorithm must allow the user to recognise relations between properties.
Rationale	It is of little value to the user to know that there is a certain feature in one property unless he also knows the corresponding values of the other property at the location of this feature.
Source	Use case analysis.
Fit criterion	Visual verification, ideally requires a user test.
Dependencies	
Conflicts	

Table 4.8: Requirement "Relations between properties"

3.0 Localisation of features (use case "Localise finding"): It must be possible to spatially localise a finding in either property (specified in table 4.9):

Requirement	**# 3.0**
Use case	Localise a feature
Name	Localisation of features
Type	Functional
Description	After discovering an interesting feature in one or both of the properties, the user needs to spatially localise it.
Rationale	To fully understand a feature and to start further actions, it is neccessary to localise it.
Source	Use case analysis.
Fit criterion	Visual verification, ideally requires a user test.
Dependencies	-
Conflicts	May conflict with requirement # 1.1.

Table 4.9: Requirement "Localisation of features"

Throughout the remainder of the thesis, these requirements will be regularly referred to. In the next section, they are employed to develop a new algorithm for the visualisation of two-dimensional volumes.

4.3 Development of the dependent rendering algorithm

In this section, a new algorithm is developed that is designed to fulfil the requirements set in the last section.

In the following, it is assumed that the data sets for both properties are registered on the same rectangular grid. Furthermore, it is assumed that the reference structure is given by some representation of the matter of the object. This is a sensible assumption because this is the property we are most familiar with to look at as most of the things we see is light reflected by matter and all other properties first have to be made visible by special aids.

The idea is to start by visualising the reference structure and keeping track of the depth at which each of the viewing rays becomes fully opaque (the very same point that is used in *early ray termination* (originally described in [Lev90], see section 5.1.1.2 for more details) to stop further processing of this viewing ray and to go on with the next one). The location where a viewing ray gets fully opaque can be understood as an opaque "wall" the observer cannot look through.

The next step is to render the dependent structure, but stopping the viewing ray at the "walls" in the reference structure. This is necessary because information originating from locations behind the reference structure's "wall" in the dependent structure cannot be matched with structural information from the reference structure.

A simple algorithm that represents this process using a ray casting technique is as follows:

```
for each pixel in the image:
 {
  set depth_until_wall to zero
  start a viewing ray in the reference structure:
   {
    initialize colour and opacity
    for each sample point on the viewing ray:
     {
      update colour and opacity
      if the ray is fully opaque:
       {
        set depth_until_wall to the current depth
        stop the viewing ray
       }
     }
   }
  start a viewing ray in the dependent structure:
   {
    initialize colour and opacity
    for each sample point on the viewing ray:
     {
      if the current depth equals depth_until_wall:
       {
        stop the viewing ray
       }
      update colour and opacity
     }
   }
  compose pixel's colour from both structures' viewing rays' colours
 }
```

The "walls" can be made transparent by changing the opacities assigned to the materials in the reference structure. So, it is possible to explore regions at different depths inside the volumes. There are different possibilities for handling the colours returned by the viewing rays. One approach would be to compose the two rays' colours to the pixel's final colour, as indicated in the above algorithm sketch. A more flexible approach, which is used in the implementation described in section 5, is to create separate images for both volumes and then fuse these images together with a user-defined weighting for both images.

A general rendering pipeline for the algorithm is depicted in figure 4.2. This figure intentionally does not refer to any specific rendering technique: Although the algorithm has been designed with direct volume rendering in mind for rendering both structures, any rendering technique can be used as long as it is possible to obtain the depth information from the reference structure and feed this information in the renderer for the dependent pipeline. It is even imaginable to use different rendering techniques for both structures.

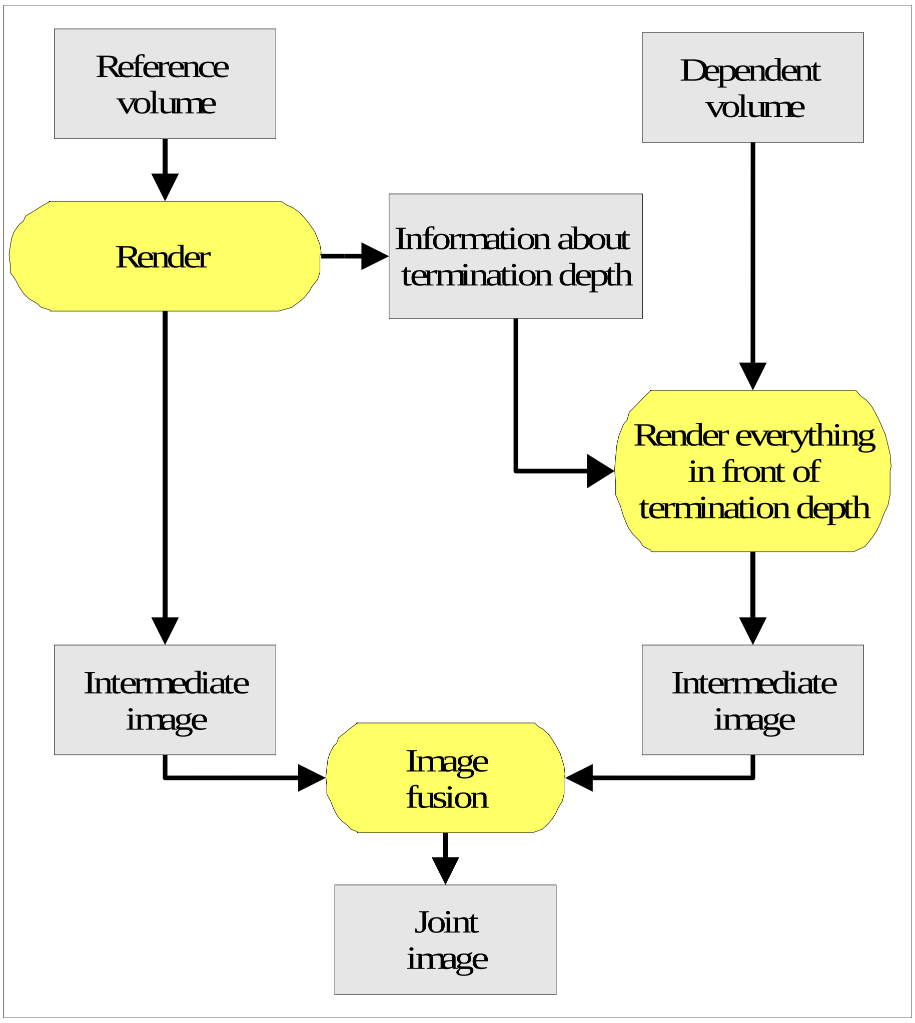


Figure 4.2: The dependent rendering pipeline

This algorithm does not represent a strict mathematical algorithm, so it cannot be assessed in terms of function errors, error propagation characteristics or precision. It is assessed by evaluating the results of an implementation against the requirements derived in section 4.2. To do so, the algorithm has been implemented in two prototypes and a module for a biomedical imaging platform, which are described in the next chapter. By evaluating each of these implementations against the requirements, the algorithm itself is evaluated.

Chapter 5

Implementations

In this section, implementations of the dependent rendering algorithm developed in chapter 4 are presented. First, two prototype implementations are described and discussed. Thereafter, a more sophisticated implementation of the algorithm for a module for a biomedical visualisation platform is described and discussed. Each of the these implementations is evaluated against the requirements derived in section 4.2.

5.1 Prototype implementations

5.1.1 Initial prototype

5.1.1.1 Motivation

As a proof of concept for the algorithm, an initial prototype has been implemented and tested with artificial data sets.

The purpose of the first prototype was to analyse whether the algorithm would be able to fulfil the requirements defined in section 4.2 at all. The speed of the algorithm was not important.

5.1.1.2 Description of the implementation

This prototype is a straightforward implementation of the dependent rendering volume rendering pipeline shown in figure 4.2. To improve the performance, the volume's normal vectors and gradient magnitudes are pre-calculated. The normal vectors are then encoded into indices to allow the use of a look-up table (LUT) for the shading (see below). The goal when designing an indexing function for LUTs is to use as little indices as possible to keep the LUT small while keeping the precision of the encoded normals as high as possible. The indexing technique used here is a variation of the technique used in the VolPack volume rendering library [Lac] and described in [Lac95]. The normal vectors are encoded to 13-bit values using the following rationale: By definition, any normal vector $\vec{n} = (x, y, z)$ has a magnitude of $1 = \sqrt{x^2 + y^2 + z^2}$. It follows $\|z\| = \sqrt{1 - (x^2 + y^2)}$, therefore only one bit is needed to encode the sign of the z-component of a normal vector. Furthermore, $x^2 + y^2 \leq 1$, which implies that the vector (x, y) describes a point inside a unity circle. Thus, the vector (x, y) can be described by an angle $\phi \in [0 \ldots 2\pi]$ and a length $r \in [0 \ldots 1]$ with $r = \sqrt{x^2 + y^2}$ and $\phi = \text{signum}(y) \cdot \arccos\left(\frac{x}{r}\right)$. Both r and ϕ are encoded with 6 bits each, which has been found to be sufficiently precise (see [Lac95], chapter 7.1.2).

The core volume rendering engine is based on material weight maps for the material classification and a pre-calculated colour LUT for the indexed volume normals: The user can define a set of different materials for the volume. The materials are stored in a data structure containing information about the voxel values occupied by this material and the material's optical properties:

```
struct Rgb {double red, double green, double blue};
struct Material {
  Rgb ambientColour;
  Rgb diffuseColour;
  Rgb specularColour;
  double specularExponent;
  double opacity;
  int center;
  int width;};
```

The three colours `ambientColour`, `diffuseColour` and `specularColour` and the value `specularExponent` describe the colour properties of the material in Phong's illumination model (see equation 2.3.2). The opacity of the material is described by `opacity`. The values `center` and `width` describe by which voxel values the material is represented: Usually, a material occupies not only one voxel value, but extends over several values around a centre voxel value. In this implementation, it is assumed that the probability that a given voxel value belongs to a given material depends on both this voxel value's distance to the material's centre voxel value and the material's voxel value width, which describes over how many voxel values the material is extended to both sides of the centre value. The centre value and the width are set by the user, either by experimenting with different values or by using a-priori knowledge about the data set. For each material, a weight map is built. This weight map contains a factor for each possible voxel value denoting the probability that a voxel value belongs to the material. The centre voxel value and the voxel value width must be supplied by the user.

For a material with central voxel value c and width w, where $c, w \in [0, \ldots, 255]$, the probability p for voxel value v to be part of this material is calculated as follows:

$$\begin{aligned} \text{if } \left|\frac{c-v}{w}\right| > 1 &: \ p(v) = 0 \\ \text{else} \quad p(v) &= 1 - \left(\frac{|c-v|}{w}\right) \end{aligned} \tag{5.1}$$

This equation is plotted in figure 5.1 for materials with different widths and the same central voxel value. When all materials and their respective weight maps are

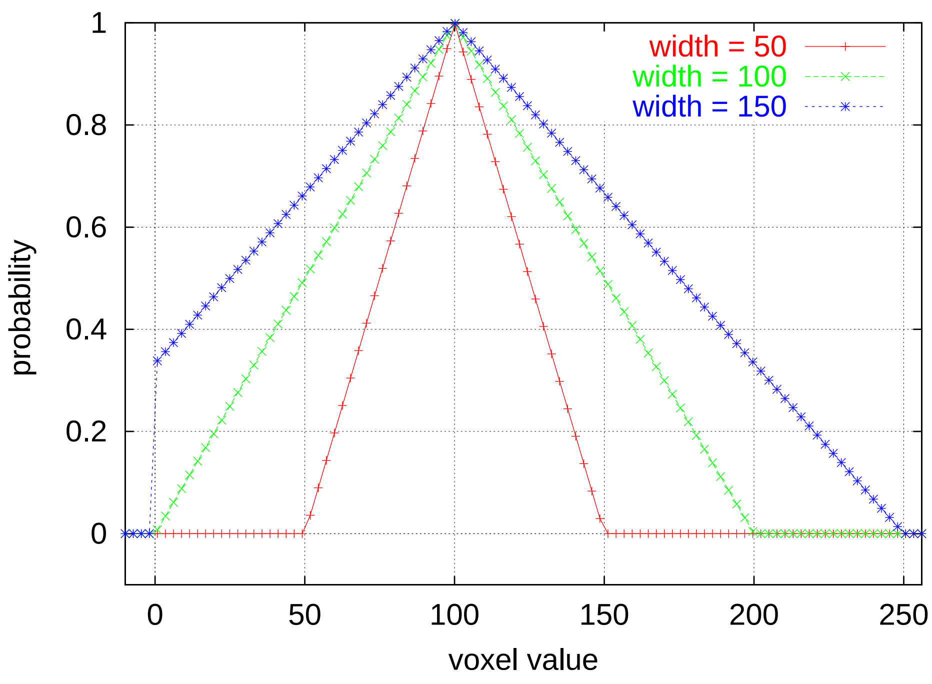


Figure 5.1: The material probability equation (see equation 5.1) plotted for three materials with the same central voxel value (c=100) but different widths (w={50,100,150})

set, these weight maps have to be normalised such that for no given voxel value the sum of the different materials' weights is greater than 1.

This weight map approach is derived from the *material percentage volumes* technique described in [DCH88]. It corresponds to the classification branch in Levoy's classical volume rendering pipeline (see figure 2.2). The pipeline's other branch, the shading, is implemented as follows:

At the beginning of the rendering process, the viewing vector is calculated and the lights are rotated according to the current viewing direction such that the lights' direction and magnitude remain constant in image space.

For each material and for each possible normal index, a resulting colour is calculated by applying the Phong illumination model (see section 2.3.2), using the current view point vector, the surface normal corresponding to the normal index and the material's ambient, diffuse and specular colours.

To avoid the so-called *self-occlusion effect* (see [MWG98]), the colour is multiplied by the material's opacity, this is known as *opacity weighted interpolation*. The self-occlusion effect is an error which occurs when interpolating colours and opacities separately. It causes artifacts such as colour shifting and contributions from empty space. Using opacity weighted interpolation implies a change in the volume rendering pipeline, as both branches now are linked (the colours from the shading branch are multiplied by the opacities from the classification branch to yield the opacity weighted colours), as shown in figure 5.2. The resulting colour is written to the colour LUT. Thus, for three colour components (the RGB colour scheme is used),

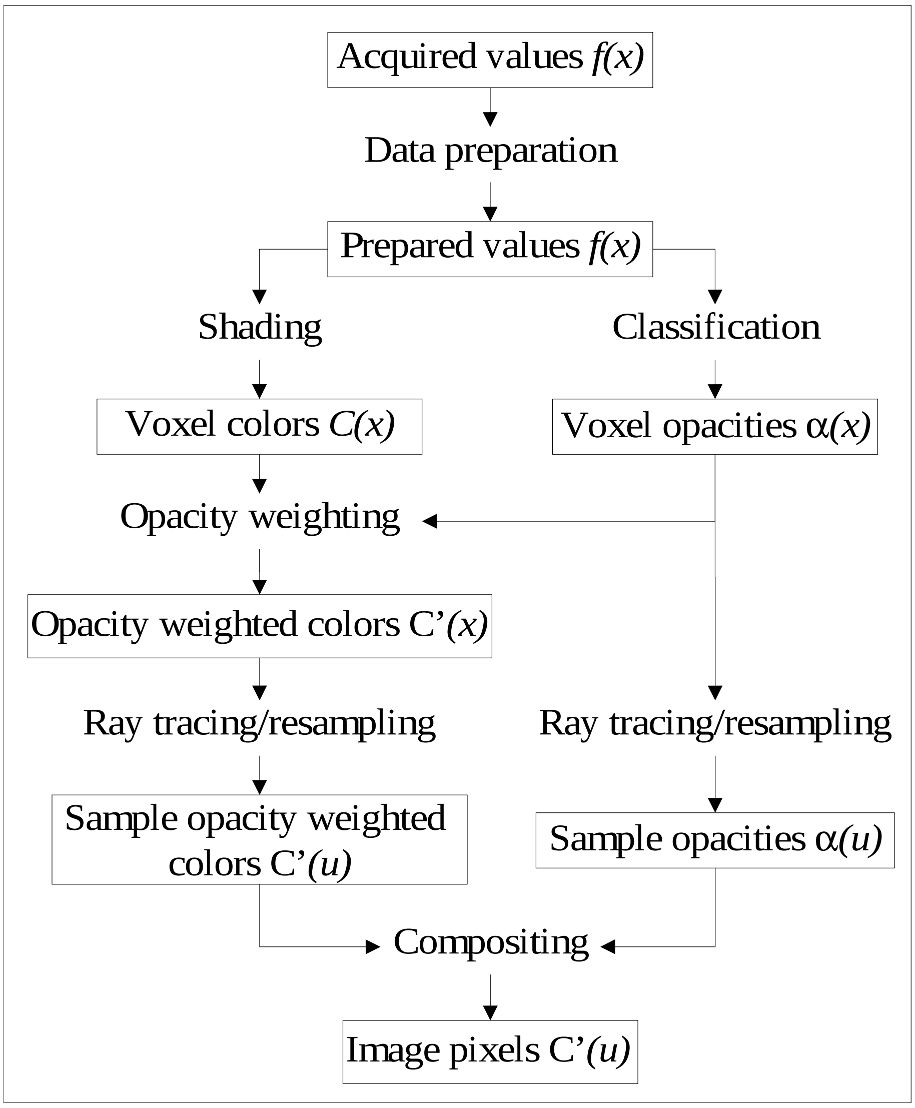


Figure 5.2: Volume rendering pipeline when using opacity weighted interpolation
Source:[MWG98]

m defined materials and i indices for the volume normal encoding, the colour LUT is a $3 \times m \times i$ matrix. Each volume has its own colour LUT, set of materials and material weight maps.

As sample points on a viewing ray seldom fall on grid points in the volume data set but in between, the colour at a sample point has to be interpolated from the surrounding grid points. In this implementation, trilinear interpolation is used. This method takes a sample point (x, y, z) and calculates the colours for each of the eight neighbours and interpolates them tri-linearly. A grid point's colour is calculated by taking the grid point's voxel value and surface normal index and calculating the colour by looking up each defined material's colours for this normal index from the colour LUT and multiplying it by the material's weight from the weight map. The grid point's final colour and opacity are the sum of all materials' weighted colours and opacities, respectively.

The colours and opacities on the ray are then accumulated using the over operator as described in section 2.3.2.

To implement the dependent rendering algorithm from section 4.3, both volumes are traversed with common viewing rays:

```
for each pixel in the image:
 {
  start a common viewing ray:
   {
    initialize colour and opacity for reference structure
    initialize colour and opacity for dependent structure
    for each sample point on the viewing ray:
     {
      update colour and opacity for reference structure
      if reference structure opacity > threshold:
       {
        stop the viewing ray
       }
      update colour and opacity for dependent structure
     }
   }
  compose the pixel's colour from both structures' colours
 }
```

For each sampling point on a viewing ray, first the colour and opacity for the reference volume are calculated and updated. If a viewing ray is nearly fully opaque, the contributions from further samples do not significantly change the final colour of the ray. Therefore, the viewing ray can be stopped when its opacity is larger than a certain threshold, which saves a substantial amount of rendering time. This technique is called *early ray termination* and has been introduced in [Lev90]. In this implementation, the threshold is set to an opacity of 0.97, with 1.0 being fully opaque and 0.0 fully transparent (this is a little bit more precise than the opacity of 0.95 reported by [Lev90]).

When the opacity for the reference volume reaches the threshold for the early ray termination , the viewing ray for both reference and dependent volume is terminated and the next viewing ray is started. Otherwise the colours and opacities for the dependent volume are calculated, and the same procedure is started at the next sample point on the viewing ray.

To test this prototype, a pair of artificial volumes has been created (see figure 5.3 a-d for renderings of these artificial volumes). The reference volume contains two large hollow cylinders of a high voxel value, which are both surrounded by a grid of smaller tubes. The dependent volume initially contains only two cubes of a high voxel value at locations that fall at the centres of the two cylinders in the reference volume. These data sets may be thought of as a very simple model of two pots with one heat source in each, surrounded by a cooling system. Using the cubes from the dependent volume as a constant heat source, a simple heat distribution is given by Poisson's equation $\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = \rho(x, y, z)$, where for the "charge distribution" $\rho = -\frac{\text{heating}}{\text{thermal conductivity}}$ it is assumed that the heating is everywhere constant at 1 and the thermal conductivity is equal to the "density", i.e. the voxel values from the reference volume. This is an equilibrium problem and can be solved numerically using the Gauss-Seidel method, which basically tries to iteratively equalise neighbouring values until the difference of Poisson's equation at neighbouring points is below a given threshold. This approach is by no means physically correct, but is a simple means to create an artificial dependent volume for testing purposes.

5.1.1.3 Evaluation of the initial prototype

As already mentioned, this initial prototype serves as a proof-of-concept for the algorithm. Therefore, evaluating the prototype means at the same time evaluating the viability of the dependent rendering algorithm.

The main goal when designing the algorithm was to be able to localise features in the dependent structure relative to the reference structure.

A key feature of the dependent rendering algorithm is that it is not possible to see features of the dependent structure behind "walls" in the reference structure. How sensible this feature is can be seen in figure 5.3 c: If it had been possible to look through the green tubes, the (correct) optical impression that the hot parts of the pots are behind the green tubes would have been destroyed.

A challenge for the algorithm would be to show whether the temperature on the surface of the pots is constant or not. Displaying the heat distribution only, as shown in figure 5.3 a, clearly fails to resolve the challenge: Although the general structure of two warm pots can be recognised, the precise location of features is not possible, especially the borders of the pots cannot be recognised.

Visualising both structures together using the dependent rendering algorithm, as shown in figure 5.3 b, nearly solves the problem, but the contributions from the reference structure are very strong and hide features of the dependent structure.

When using the dependent rendering algorithm and displaying only the dependent structure, as in figure 5.3 c, the power of the algorithm is clearly visible: No heat originating from behind the cylinders surface is visible and it is clearly recognisable that the temperature in the middle of the pots is higher than at the top and the bottom. The grey parts at the top of both grey tubes are not an artifact or error but a feature: In the reference structure, the cylinders are classified as being fully opaque. In the dependent structure, a very low opacity is assigned to the cold parts (if the cold parts were assigned a high opacity, they would occlude the hot parts) and a high opacity is assigned to the hot parts. In the grey regions, there is no contribution from the hot parts in the dependent structure in front of the "wall" (i.e. the cylinder surface) in the reference structure. Therefore, the viewing ray in the dependent structure contains only some low opacity samples from the cold parts and therefore has a colour nearly identical to the background colour. By changing the voxel values that are classified as being "hot", the user can explore how the temperature changes over the cylinder's surface.

Although the algorithm gives good hints about the localisation of features in the dependent structure, it cannot provide precise information about which voxel represents a given feature. For this task, the image gallery technique described in section 3.3.1.1 seems appropriate.

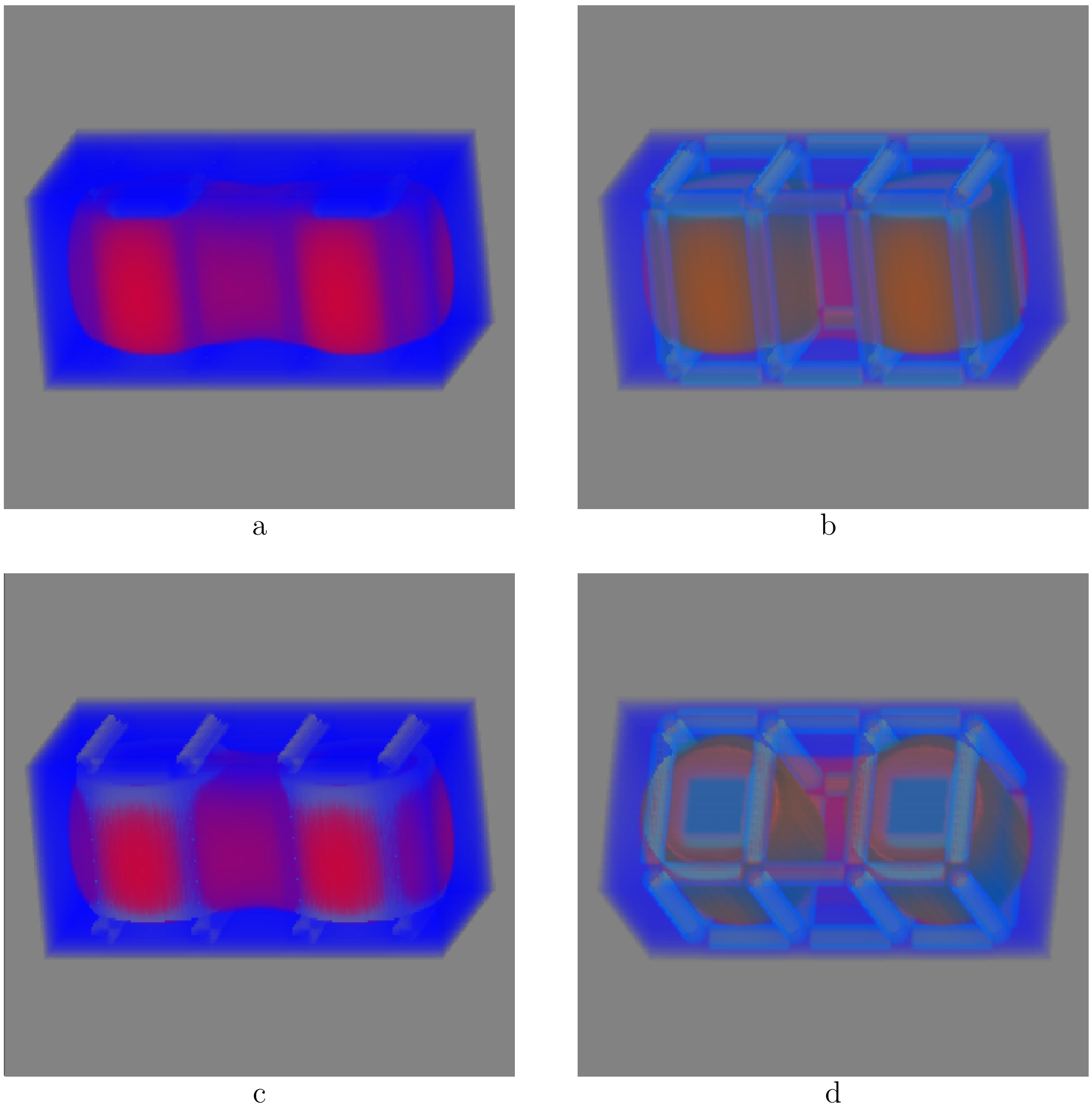


Figure 5.3: Dependent volume rendering. Images created using the initial prototype. The images show the artificial data sets described in section 5.1.1.3 (the reference property is the matter distribution and the dependent property the heat distribution of an simulated object). In image a, only the heat distribution is rendered. In images b,c and d, both properties are rendered.

Thus, for exploring unknown data sets or for searching for features whose location is unknown, the dependent rendering algorithm seems suitable and superior to all techniques introduced in section 3.3.

To evaluate the viability of this prototype in a structured manner, the fulfilment of the requirements defined in section 4.2 is checked.

- **Localisation:** As shown in figure 5.3, the user can easily and precisely locate features of the dependent property in relation to the reference property. For a full understanding of the second property, the user must be familiar with the first property.

- **3D-impression:** The renderings convey a good 3D impression of the structure of the rendered object, but their quality highly depends on the settings for the materials. The difficulty of finding appropriate settings for the transfer functions and strategies for efficiently finding appropriate settings are described in [RT01].

- **Arbitrary properties:** The reference volume is assumed to be a representation of the matter distribution. This assumption conforms to the assumptions made when developing the algorithm in section 4.3. For the dependent volume, no assumption on what property is represented is made, so arbitrary properties can be used for the dependent volume.

- **Exploration**

  - **No need for segmentation:** If volume rendering is used, there is no a-priori need for segmenting the data sets. By providing facilities for the user to change the classification of materials in standard volume rendering techniques, regions of interest can easily be highlighted and other parts can be faded out. However, if segmented data sets are available, these can be used.

  - **Interactivity:** This prototype implementation is not interactive, rendering of a pair of 256^3 volumes takes about 60 seconds on a standard 1GHz PC. This must be improved.

  Thus this requirement is only partially fulfilled: Data sets can be examined from all directions by rotating the volumes and by changing the material classifications, arbitrary parts of the volumes can be hidden or highlighted. This makes it possible to explore unknown data sets. But the rendering time is too slow for interactive use.

- **Relations between properties:** For simply localising a feature, it might be enough to add a coordinate system to the visualisation. But in order to recognise two properties in relation to one another, it is crucial to visualise them jointly. This feature is automatically achieved by the algorithm: As both properties are shown in the same projection image, all local relations are immediately visible.

- **Any property can be an entire volume:** From figure 5.3, it can be seen that the prototype can handle a dependent volume that is extended over the whole range of the reference volume. In particular, the prototype assumes that both volumes have the same extensions, so this requirement is fulfilled.
- **Different sizes and resolutions of the data sets:** As both volumes are traversed simultaneously with identical viewing rays, both volumes must be of the same size and the same resolution. If they are not, they must be transformed to a common grid before rendering. This restriction is not a general deficiency of the dependent rendering algorithm. The algorithm could also be implemented such that by using different transforms for the viewing rays of the two volumes, volumes of different sizes and resolutions can be used. This option was abandoned because at the time of implementation, all data sets available to the author were of the same size and resolution.

This evaluation shows that this initial prototype does in fact fulfil the most important requirements, especially the localisation of features and the possibility to have a complete volume for the dependent property. This also implies that the algorithm itself represents a viable approach to visualise two-dimensional volumes.

The main problem of this prototype is low speed which prevents interactive use. This solve this problem, a second and faster prototype was developed.

5.1.2 Shear-warp prototype

5.1.2.1 Motivation

The goal of this prototype is to overcome the speed limitation of the initial prototype. Several high-speed volume rendering techniques exist. Some of them need special hardware, like the VolumePro volume rendering board ([PHK$^+$99], currently developed and sold by TeraRecon, San Mateo, CA, USA), which is supported by e.g. the visualisation toolkit VTK (see [Inc], [Sch01]). Other techniques, like texture-based volume rendering, convert the volume's slices to textures and use hardware graphics accelerators for the rendering (see e.g. [VK96]). All techniques that make use of special hardware have not been considered further for the implementation because no such hardware was available to the author.

Two techniques that do not depend on special hardware are *frequency domain volume rendering* (FVR, see [DNR90], [Mal93], [TL93], [Lic95]) and the *shear-warp technique* (see [LL94]).

The FVR technique is based on the *projection slice theorem*, which says that the 2D image created by taking the line integrals through a 3D volume along rays perpendicular to the projection plane, and the 2D spectrum created by extracting a slice parallel to the projection plane and passing through the origin of the Fourier transform of the volume, form a Fourier transform pair. The image of line integrals through a volume is similar to a volume rendered image. Thus, by first Fourier transforming the volume, then extracting a 2D spectrum through the origin of the transformed data set and inverse Fourier transforming this spectrum, a projection similar to volume rendering can be generated. The complexity of volume rendering a cubical volume with n^3 voxels is $O(n^3)$ because all voxels must be touched during rendering. The computationally expensive 3D forward Fourier transform has to be done only once. For projections to different viewing directions, only the slice with the 2D spectrum has to be interpolated and inverse Fourier transformed. The 2D inverse Fourier transform of a slice of n^2 samples using the inverse *fast Fourier transform (FFT)* is $O(n^2 \log n)$. As the 2D interpolation of the slice from the Fourier transformed volume is computationally inexpensive compared to the inverse Fourier transform, the overall complexity of rendering a projection is $O(n^2 \log n)$. Thus, using FVR for the rendering is computationally much less expensive than volume rendering in the spatial domain. However, operations like shading and classification are difficult to use with FVR and changes in the classification require a re-computation of the forward Fourier transform of the volume, which is computationally expensive. There are variations of this technique using different transforms to transform the volume into the frequency domain (see e.g. [Mal93] for a use of the fast Hartley transform instead of the fast Fourier transform).

The dependent rendering technique relies on the knowledge of the termination of viewing rays in the reference structure and the ability to stop the viewing rays in the dependent structure at this termination depth. Unfortunately, the FVR technique does not provide this information, thus FVR can not be used for dependent volume rendering.

The shear-warp algorithm does not have the shortcomings of the FVR and has therefore been chosen for the implementation. It is shortly described, together with its adaption to the dependent rendering algorithm, in the next section. This implementation has already been described in ([LM02b]).

5.1.2.2 Description of the implementation

The shear-warp algorithm as described by [LL94] is a high-speed volume rendering algorithm. The basic idea of this algorithm is to factorise the viewing transform M_{view} into three separate transforms:

$$M_{view} = P \cdot S \cdot M_{warp} \tag{5.2}$$

First, the main viewing direction is determined and the coordinate system axes are permuted such that the z-axis in object space is parallel to the main viewing direction. This is done using the permutation matrix P. Then, the planes orthogonal to the main viewing direction are sheared using the shear matrix M_{shear}. The sheared data set is projected orthogonally onto an intermediate image plane. This projection is done front-to-back in a plane by plane fashion. The intermediate image is then warped using the warp matrix M_{warp} to yield the final image. By construction of the algorithm, scan lines in the intermediate image are parallel to scan lines in the sheared volume.

The main advantage over the classical image-order ray casting volume rendering approach as described by [Lev88] is that the volume is traversed slice by slice in *object order* (aka. *volume order*), i.e. voxels are accessed in the order they are stored. This allows several optimisations, the most important being that coherent runs of

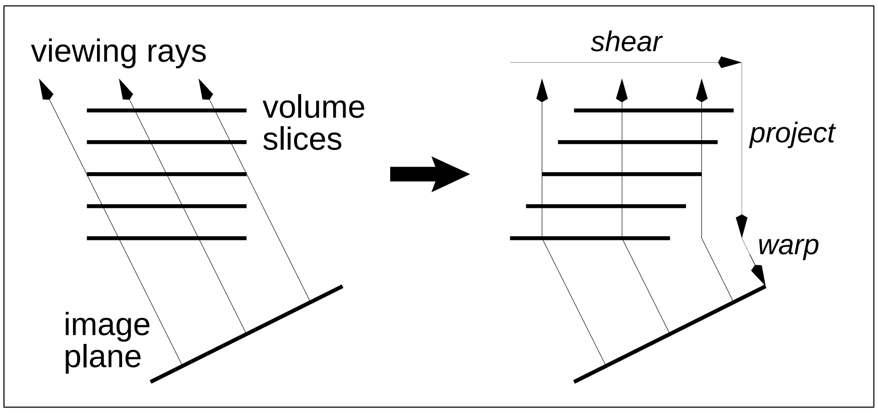


Figure 5.4: Sketch of the factorisation of the viewing transform (left) into a shear and a warp transform (right) in the shear-warp algorithm.
Source:[LL94]

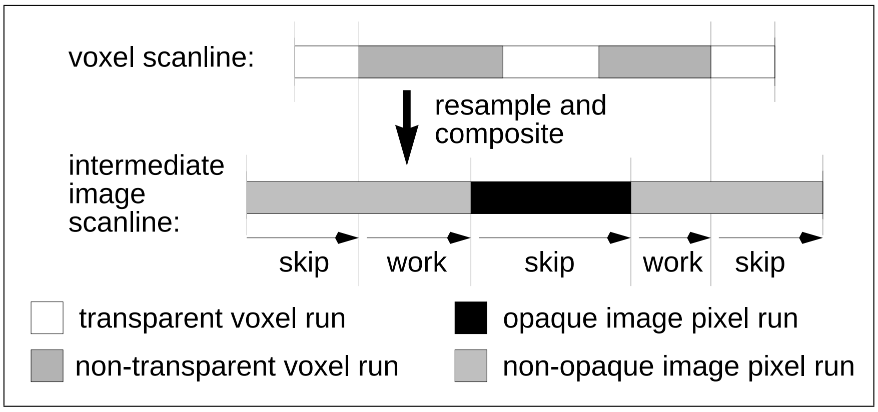


Figure 5.5: Skipping of transparent voxel runs and opaque pixel runs in the shear warp algorithm.
Source:[LL94]

transparent voxels (which usually make up a large part of the volume) and voxels contributing only to opaque pixels can be skipped (see figure 5.5). Furthermore, the interpolation factors in one slice are constant for all voxels, so they can be computed for the first voxel of a slice and then reused for all other voxels in this slice.

Besides being very fast, using the shear-warp algorithm appears straightforward because it uses a map containing the intermediate opacity values for each pixel in the rendering image to determine the voxel runs that are occluded and can therefore be skipped. This map can be extended to a depth map to determine whether a given voxel in the dependent volume has a visible reference in the reference volume.

To use the shear-warp algorithm, the dependent rendering technique has to be adapted as depicted schematically in figure 5.6: A depth map with one entry for each pixel in the rendering image is created. First, the reference volume is rendered. While doing this, the depth map is filled: The first time a pixel becomes opaque, the depth of the voxels at this step is written to the depth map at the same position as the pixel. Then the second property is rendered. To take care of the depth

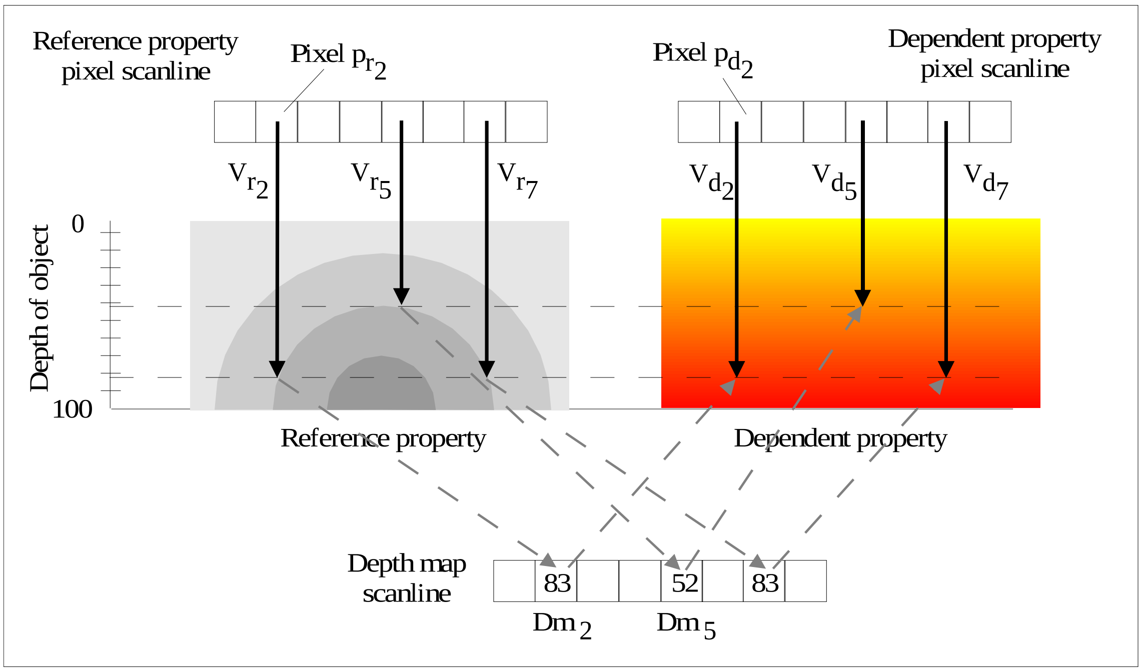


Figure 5.6: Shear warp prototype: Interaction of reference volume, depth buffer and dependent volume.

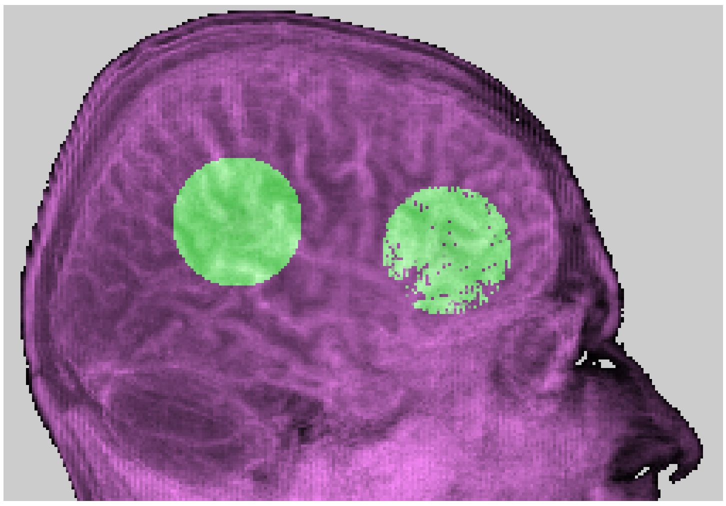
information while rendering the second property, the aforementioned optimisation technique of skipping opaque pixels is extended: For each pixel to be updated, it is first checked if the pixel is opaque. If it is opaque, one skips to the next non-opaque pixel. If the pixel is not opaque, it is checked whether the current depth is higher than the corresponding depth in the depth map. If it is not, the pixel is updated as usual, otherwise the current pixel is marked as opaque. This allows for a seamless integration into the algorithm for efficient skipping of runs of opaque pixels, which is described in detail by [LL94].

This depth map approach similar to the classical z-Buffer technique. The conceptual difference is that the z-Buffer is intended to store the depth of different objects while the purpose of the depth map is to compare the viewing depth of different properties of the same object.

Compared to the first prototype implementation described in section 5.1.1, only the traversal scheme is different, while the same techniques for material classification, shading and composing are used.

5.1.2.3 Evaluation of the shear-warp prototype

In this section, it is investigated how well the algorithm fulfils the requirements defined in Section 4.2. Figure 5.7 shows two example images generated by our implementation using an MRI scan of a human head and an artificial second property consisting of a small ball of constant density located inside the brain. The reference volume is an MRI of a human brain (with parts of the skull removed), the dependent


(a) From the right side.

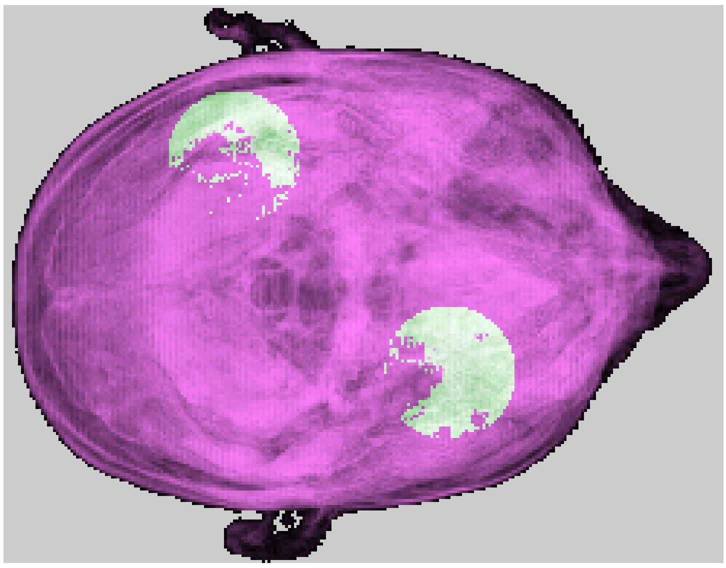

(b) From the bottom.

Figure 5.7: Two example renderings created with the shear-warp prototype. (MRI data set of the head courtesy of, and copyright by, Mark Bentum).

volume has been artifically created and is meant to represent regions of suspected tumour localisations. From image (a) it is perceivable that the right sphere from the second property is farther to the left side of the skull than the other. This is confirmed by image (b)

As the shear-warp prototype is an enhancement of the first prototype and uses the same core classification and composing techniques, most of the evaluation results of the first prototype in section 5.1.1.3 also apply here. Therefore, the following list only describes the requirements with differing evaluation results.

4.(b) **Interactivity:** Compared to the first prototype implementation, this implementation is significantly faster, although it has not been specifically tuned for efficiency. Nevertheless, on a standard 1GHz-PC workstation with 512 MB RAM running Linux, the rendering of a pair of 256^3 volumes takes about one second. Thus, interactivity is achieved by this implementation. Furthermore, the algorithm is very well suited for parallel or distributed rendering (see e.g. [Lac96] or [MS98]), which allows for interactive implementations even for larger volumes.

7. **Data sets of different sizes and resolutions:** The algorithm allows the use of differently sized data sets, as long as no data set's size is larger than the reference property's. Different resolutions and even different coordinate systems for the data sets cause no problems as long as there is a mapping that allows the use of the depth map generated using the reference data set together with the other data sets. It is proposed to interpolate bi-linearly between the coordinate system of the depth map and that of the dependent property's data set, and then to decide on a threshold for whether a point is to be considered as occluded or not.

The technique of using a depth map provides further flexibility: After generating the depth map by rendering the reference volume, the dependent volume could then be rendered using another technique like surface rendering, as long as this other technique appropriately uses the depth map.

5.1.3 Conclusion for the prototypes

The evaluations of the prototypes show that the algorithm fulfils the requirements defined in section 4.2.

However, the prototypes have been tested only with (at least partially) artificial data sets. To test whether the algorithm can be used in a real-world environment, it must be tested in an application with real data. Such an implementation will be described in the next section.

5.2 Implementation for a biomedical imaging platform

5.2.1 Motivation

The prototypes have shown that the dependent rendering algorithm is promising for the visualisation of multi-dimensional volumes. Therefore, the algorithm should be implemented in real-word settings and tested with real data. The algorithm has been implemented as a module for the biomedical image analysis and visualisation platform of the medical imaging group at the Universidad Politécnica de Madrid and the Hospital General Universitario Gregorio Marañón in Madrid (HGUGM). The implementation has already been partially described in [LPDS03].

5.2.2 Extension of the requirements

The HGUGM already uses a custom-made software package for the analysis and visualisation of biomedical images which has been described in [DLB+99]. For functional imaging, this program is able to handle two registered volumes (usually one anatomical and one functional volume) at one time, and employs a 2D slice viewer to jointly visualise these volumes. Typical anatomical volumes are produced by CT or MRI imaging facilities, while the functional volumes come from fMRI or PET. To extend the platform, it was decided to implement a module for the joint 3D visualisation of anatomical and functional data.

The requirements for this implementation can be classified into two groups: Visualisation requirements that describe what the new visualisation module is supposed to deliver, and environmental requirements that describe requirements and restrictions that depend on the already existing implementation.

The visualisation requirements were as follows:

- Joint 3D visualisation of functional and anatomical data;
- Improved mental 3D reconstruction of the context;
- Colouring of properties according to a standard colour table: In the already existing 2D fusion interface, the values of the functional data are represented by different colours. This colouring scheme must be maintained in the 3D rendering to allow easy comparisons between the 2D and the 3D visualisations.

The environmental requirements were:

- The module must be easily integrated in the existing platform with as little changes to the existing code as possible. The platform is written in IDL (Interactive Data Language by Research Systems Inc.), a language for processing and visualising large data sets. IDL includes a GUI widget collection. This implies that the module has to be written in IDL, too.

- The module must be integrated such that it does not disturb any existing workflows of the users of platform.
- The module must be easy to use, especially the setting of the transfer functions for the 3D renderings must be easy to use.
- The implementation must allow interactive use.

In the next sections, the implementation is described.

5.2.3 Description of the implementation

In the following, the newly developed module for the joint visualisation of functional and anatomical data will be referred to as just *the module*. It was decided to use the dependent rendering technique for the module's core rendering technique as the prototypes suggest that it can fulfil the visualisation requirements. The reference volume is given by the anatomical volume and the dependent volume is given by the functional volume. The existing joint 2D visualisation interface, called *fusion*

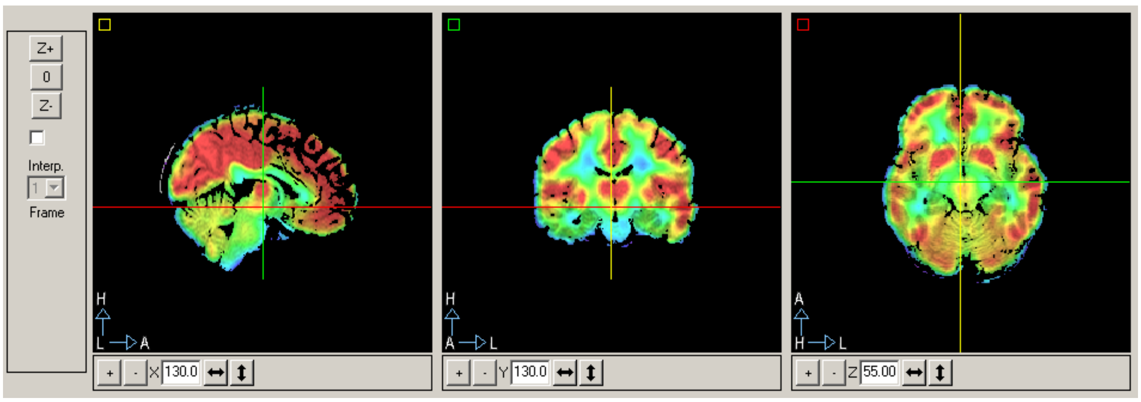


Figure 5.8: Screen shot of the fusion interface's 2D display widgets.

interface, is shown in figure 5.8. It consists of a special *ortho-viewer*, i.e. a widget that displays three orthogonal slices of a volume: Three images are displayed, each contains a fused image of the same slice from both volumes. The slices for the three images are orthogonal to one another. The fused slice images are created in the HSV (Hue-Saturation-Value) colour space by taking the grey level values from the anatomical volume for the HSV *value* channel (thus representing the intensity) and the grey level values from the functional volume for the HSV *hue* channel (thus representing the colour). In the HSV colour space, both the *saturation* and the *value* channel have a range from 0 to 100%. The hue values are represented by the angles $0°$ to $360°$ on the HSV colour circle, where $0°$ and $360°$ are red, $120°$ is green, $240°$ is blue, and the other angles represent mixtures between these colours.

The grey level values from the anatomical volume are linearly mapped to the full range of [0%..100%] for the *value* in the HSV space. The grey level values from the functional volume are linearly mapped to [240..0] for the *hue* in the HSV space, thus creating a smooth colour map from blue for low functional volume values via green to red for high functional volume values.

Additionally, the HSV *saturation* channel is set to a user-defined value (with 100% as default) everywhere where the functional volume is defined and to 0% at all other points. This effectively prevents the background from taking the colour defined by the lowest hue value and thus unnecessarily polluting the rendered image. The fusion interface contains several controls to set the window/level function parameters for both volumes.

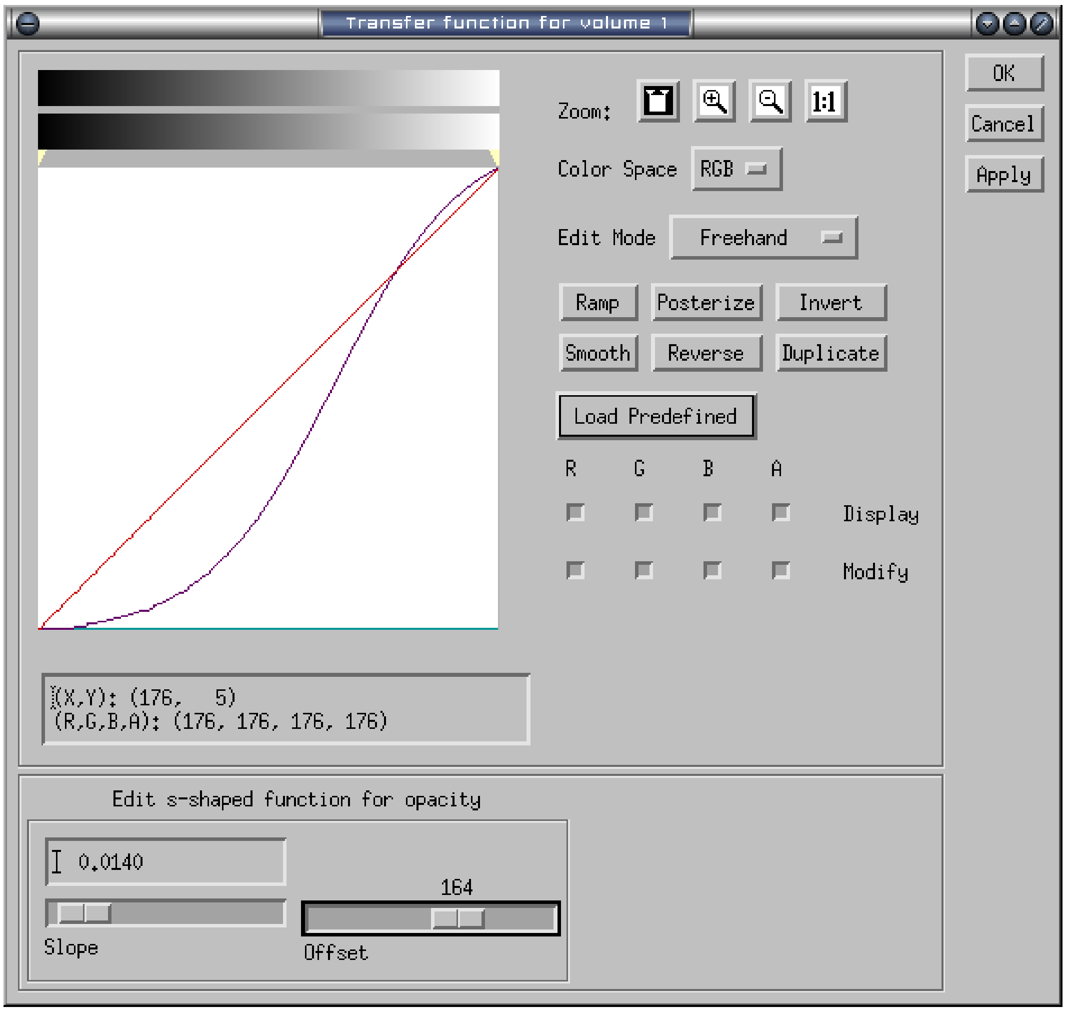
All settings in the fusion interface are applied in real-time.

The new module is integrated in this fusion interface. It is initialised from the colour and window/level settings in the fusion interface. One of the difficulties in volume rendering is to find the appropriate volume rendering transfer function, i.e. the function that maps voxel values, sometimes additionally considering the local voxel gradient, to colours and opacities. There is no recipe on how to find the best transfer function for a given volume, and usually a lot of manual fine-tuning is needed. In [RT01], the most common techniques to find a transfer function are compared, with no clear recommendation on which technique to use. The module uses the information already available from the 2D fusion interface to create simple yet meaningful transfer functions: Assuming that the user already has set the window/level settings such that the features he is interested in are displayed, the opacity transfer function for the anatomical volume is initially set to a simple ramp. For the functional volume, it is usually desirable to give the regions of low function signal values (which cover the largest part of the data set) very little opacity and the smaller regions of high activity a high opacity. Therefore, an S-shaped transfer function is employed for the opacity. The function that creates the S-shape is $f(x) = 1 + \tanh(slope * (x - offset))$. The position of the S in the vertical direction can be changed using the parameter $offset$, while changing the parameter $slope$, the slope of the S can changed. Both the centre point and the slope of the S can be changed by the user (see figure 5.9).

For both volumes, the colour transfer function is a linear ramp, mapping voxel value to grey levels. Thus, the intermediate images created are grey level images. The image fusion technique used to fuse these two grey level intermediate rendering images to a joint image uses the same technique as the existing ortho-viewer described above to facilitate the comparison of images created using the 2D technique with those created using the 3D technique.

The transfer functions can be manually edited and fine-tuned to obtain optimal results.

IDL's object graphics interface already contains a volume class with an integrated volume rendering engine. Although this volume rendering engine supports up two volumes to be rendered at a time, it is not suitable for the dependent rendering technique, because its composing technique multiplies the contributions of each voxel in one volume by the contribution of the according voxel in the other volume. Thus, supposed in the reference structure a voxel is classified as totally transparent, the contributions from the dependent volume at this location are discarded. Therefore, IDL's multi-volume rendering capability is not considered any further.



(X,Y): (176, 5)
(R,G,B,A): (176, 176, 176, 176)

Edit s-shaped function for opacity

0.0140

164

Figure 5.9: Screen shot of the transfer functions dialogue for the functional volume. The colour transfer function is a linear grey level ramp, while the opacity transfer function has the shape of an S. The slope and the offset of the S can be set using the two sliders at the bottom of the dialogue.

As IDL is a closed source software, it is difficult to integrate a new volume rendering system into the existing volume class. Although IDL provides functionality to create external modules in the C programming language and call these from IDL, there are several reasons to try to use IDL's built-in volume rendering engine instead of implementing a new external volume rendering module:

- The integrated renderer automatically interacts with other objects in the scene, which is important in case any further objects are placed into the scene, e.g. a 3D cursor.
- It is fully integrated in IDL's rendering and widget system and provides ready support for the viewing transformations.
- It is highly optimised and provides out-of-the-box symmetric multi-processor (SMP) support.

- A new external module must be ported and tested for each platform it will be run on, while IDL code is the same on all supported platforms. As at least two platforms (Linux and Windows) and potentially a third one (Solaris) are used, this would be a lot of extra work.

First tests of implementing the dependent rendering algorithm in IDL code without using IDL's standard volume renderer made this idea look feasible but extremely slow. Thus a technique to mimic the behaviour of the dependent rendering algorithm by using the standard IDL volume renderer had to be developed. This can be done

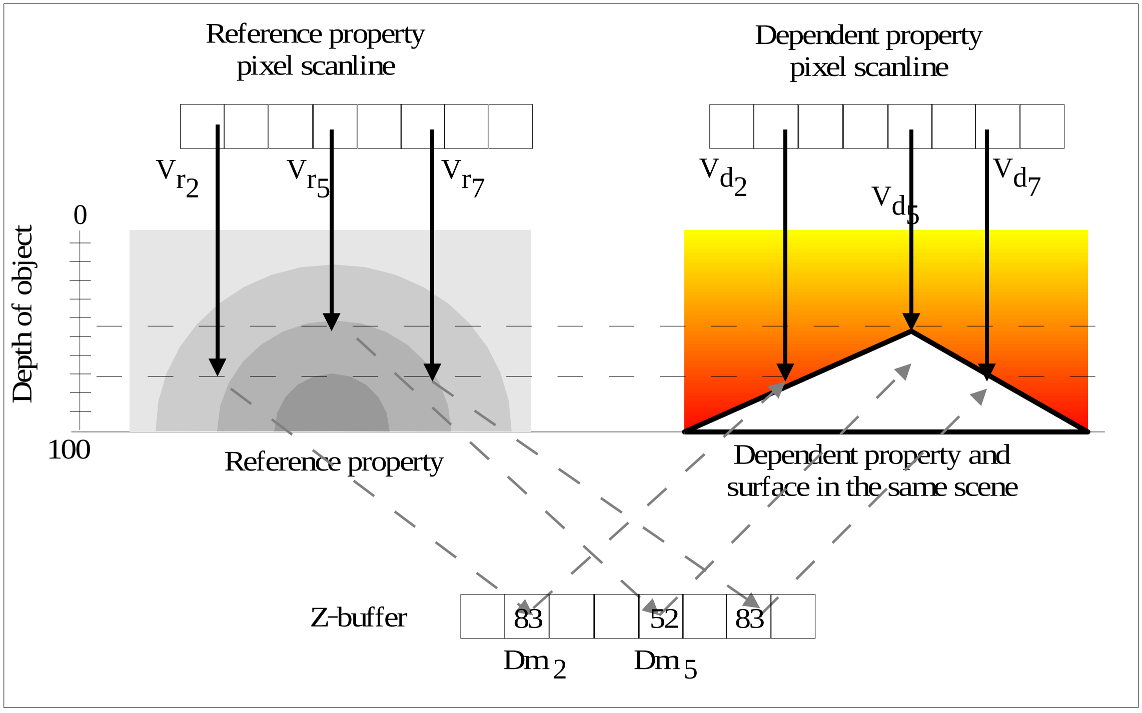


Figure 5.10: Biomedical imaging platform: Interaction of reference volume, z-buffer, background surface and dependent volume.

by using a depth map similar to the one employed in the shear-warp technique described in section 5.1.2. The technique is depicted schematically in figure 5.10: The reference volume is rendered first into a frame buffer and it is kept track of the termination position of the viewing rays using the z-buffer. From the values in the z-buffer, a totally opaque surface with the colour of the image background is created. This surface represents the fully opaque "wall" described in section 4.3. A new scene is created, in which this surface and the dependent volume are placed. This scene is rendered into a separate frame buffer, with the surface being rendered first and then the dependent volume. This ensures a proper termination of viewing rays of the dependent volume at the same depth as in the reference volume.

The difference between this technique and the technique for the shear-warp algorithm shown in figure 5.6 is that for the shear-warp technique, explicit control code is necessary to stop the dependent volume's viewing rays when the depth from the depth map is reached, while in this approach, the viewing rays “automatically” stop when hitting the opaque surface.

The frame buffers now contain two images which are equivalent to those created by the dependent rendering technique. These images are then fused to yield the final joint image.

The resulting rendering pipeline is shown in figure 5.11. Compared with the original

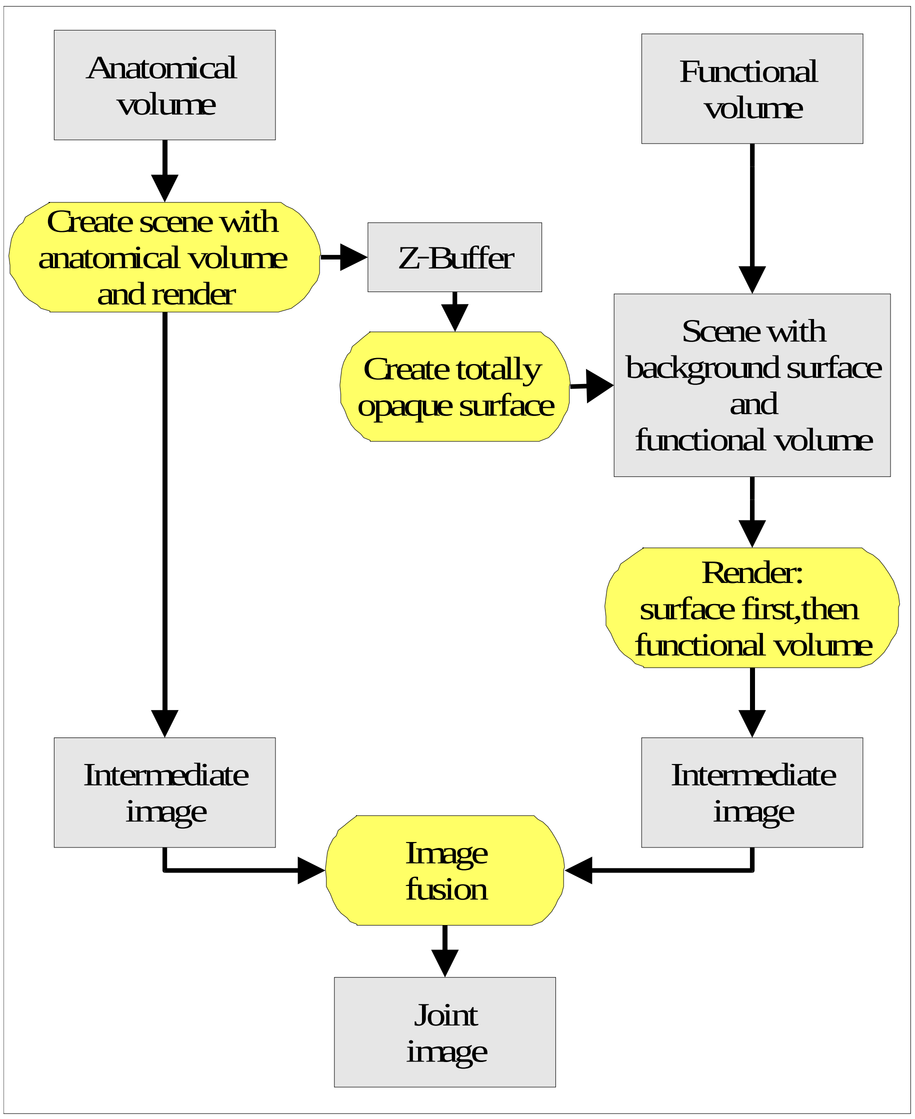

Anatomical volume
Functional volume
Create scene with anatomical volume and render
Z-Buffer
Create totally opaque surface
Scene with background surface and functional volume
Render: surface first,then functional volume
Intermediate image
Intermediate image
Image fusion
Joint image

Figure 5.11: Biomedical imaging platform: Adapted dependent rendering pipeline.

dependent rendering pipeline depicted in figure 4.2, there are only small differences in the handling of the depth map (which is represented by the z-buffer) and the interaction of the depth map and the dependent volume (instead of direct interaction, an auxiliary surface is created from the z-buffer and placed in one scene with the dependent volume).

5.2.4 Additional requirements and their implementation

During the development, additional requirements evolved, which will be described in the following, together with their implementation. These requirements reflect special needs for the visualisation of human brain volumes where the anatomical volume is an MRI and the functional volume is a PET. The anatomical volumes contain only the brain, the surrounding skull and tissue have been removed by initial segmentation.

Visibility of ventricles The ventricles are an important landmark in the anatomical volume. In the 2D slice images, they are clearly discernable, but in volume renderings of MRI data sets, they are often difficult to locate (they usually have the same voxel values as the background and are rendered as totally transparent). In a first approach, the ventricles had been segmented from the data set using a simple region-growing technique, converted into a coloured polygon and then rendered together with the reference volume. Drawbacks of this technique are that segmentation takes additional time and that the rendered polygon representing the ventricles occludes those parts of the brain that are behind the ventricles.

A more convenient and more flexible approach is to use voxel gradient dependent classification for the volume rendering of the anatomical volume. In this approach, the classification is based not only on the voxel value, but also on the local gradient of the voxel. By increasing the contribution of a voxel's local gradient to its classification, surface boundaries can be emphasised.

This technique has been introduced by [Lev88] and [DCH88] and is a standard technique (see [LCN98], pp. 88-95). Unfortunately, it is not supported in a configurable manner by IDL. Therefore, the gradient for the anatomical volume is pre-calculated and the volume is weighted by a user-defined factor, thus giving more or less emphasis to surface boundaries like the ventricles. The user can change the gradient-weighting between renderings. Using this technique, the surface of the ventricles can be highlighted while the parts of the brain surrounding the ventricles are still visible.

Display of cortex only In functional biomedical imaging, often the most interesting region is the brain's cortex. Therefore it is desirable to show only the cortex and not the white matter. Properly segmenting grey from white matter is non-trivial and usually requires much manual help from experts, therefore this cannot be used here.

For volume rendering, a non-perfect but sufficient solution is as follows: Grey and white matter are represented by slightly different voxel values. It is not possible to fully separate them using a simple voxel value threshold, as the regions overlap and due to imaging modality imperfections parts of the grey matter contains voxel values of white matter and vice versa. When using a voxel value threshold such that all grey matter and parts of the white matter are selected, the still selected white matter consists mainly of small chunks. During volume rendering, these chunks are nearly invisible as the contribution of only a few voxels to the rendered image is small. The gradient-weighting further improves this effect: The gradient is calculated on the volume before removing the white matter. The border between grey and white matter is very blurry and the gradient there is very small, which leads to a small weight for the remaining chunks of white matter.

This threshold is set by the window/level settings of the anatomical volume. While the standard window/level function clamps all values outside the window to the values at the respective window boundary, the module sets all values outside the window to 0. This is depicted in figures 5.12 and 5.13: For a normalised data set with voxel values between 0 and 1, the input voxel values from the x axis are mapped through the window/level function to the output voxel values on the y-axis.

Although this approach is very simple, it is efficient, user-friendly and fulfils its purpose.

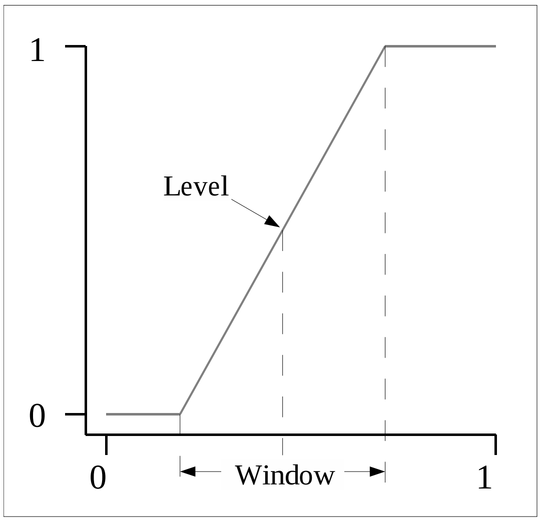


Figure 5.12: Standard clamped window/level function

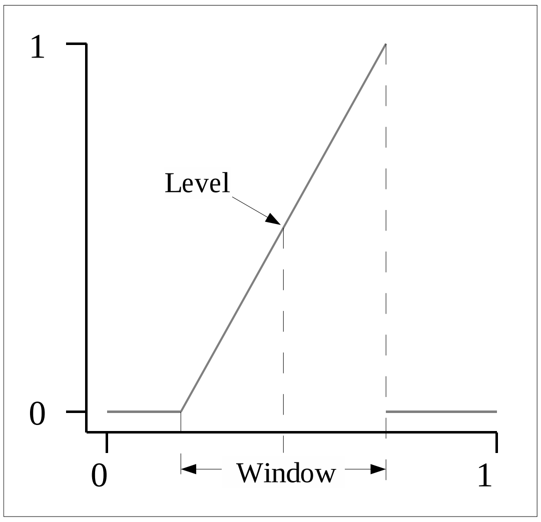


Figure 5.13: Window/level function with values at both borders set to zero and scaled to the range [0..1]

Interactive 3D cursor To improve the transfer of knowledge from the 3D to the 2D visualisation and vice versa, it is essential to have a marker that is present in both visualisations. In the 2D ortho-viewer, in each slice, the position of the other two slices relative to the current slice is indicated by coloured lines. In a similar fashion, a 3D cursor has been created that indicates the intersection point of the three 2D slices in the 3D rendering. This 3D cursor is updated interactively from the position of the slices, i.e. the cursor position can be changed by scrolling through the slices.

To be interactive, the 3D cursor has been realized using IDL's "instancing" feature: Instancing allows static parts of a scene to be rendered once into a buffer and changing parts to be rendered afterwards. This greatly improves the performance if the static part is expensive to render while rendering the moving part is cheap. This is the case here: Rendering the volumes is very expensive, while the cursor consists of just three lines.

Standard cutting features The module has been extended by two cutting features: Cutting planes just cut off parts of the volume parallel to the data set boundary surfaces. Cutting cubes allow the user to cut-off a rectangular piece. The planes that determine the cutting surfaces can be updated from the positions of the slices in the 2D fusion interface.

5.2.5 Evaluation of the implementation

In this section, some results are presented and discussed. Based on these results, the implementation is evaluated.

To show the improvements of the 3D approach over the 2D approach, a 2D visualisation in figure 5.16 is compared to a 3D rendering in figure 5.17. The anatomical volume is an MRI and the functional volume is a PET of a human brain, where parts have been removed by surgery.

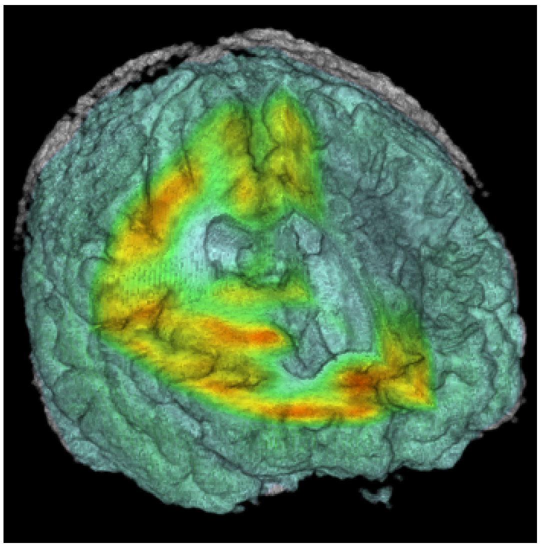


Figure 5.14: 3D rendering with clearly recognisable ventricles, which can be used as landmarks for the localisation of features. This is the same data set as in figure 5.16.

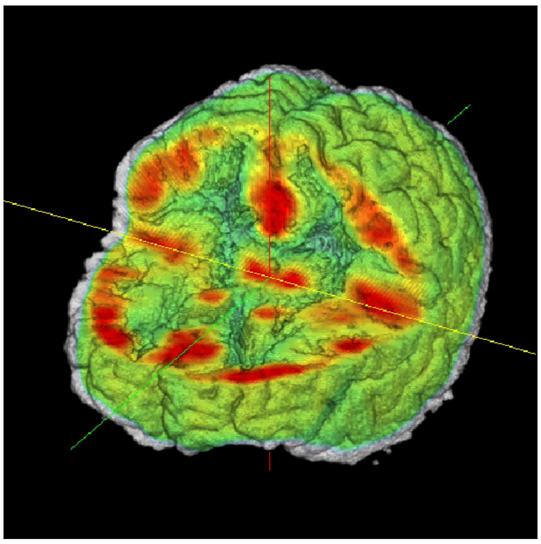


Figure 5.15: 3D rendering where the cortex is well recognisable while the white matter has been renderer transparently. This is the same data set as in figure 5.8

In the 3D rendering in figure 5.17, the 3D structure of the object is easy to perceive. The 3D cursor is exactly at the same position as the according slices in the 2D display, which fulfills the "interactive 3D cursor" requirements from section 5.2.4.

As additional requirements for this implementation (see section 5.2.4), two special visualisation features were implemented: First, the possibility to recognise the ventricles, which are important landmarks for the localisation of features, in the 3D renderings. This is shown in figure 5.14. Second, as shown in figure 5.15, it is possible to highlight the cortex by rendering everything else transparent.

To evaluate this implementation under the aspect of the dependent rendering technique, the fulfilment of the requirements derived in section 4.2 are evaluated.

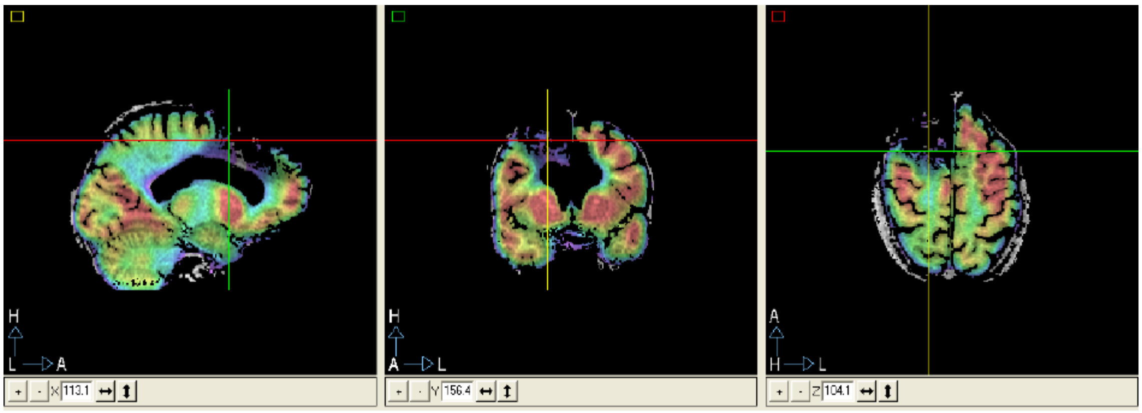


Figure 5.16: Two orthogonal slices of two data sets jointly visualised in the classical 2D fused view. The anatomical data set is an MRI, the functional one a PET. Note the pathological asymmetry in the functional data set.

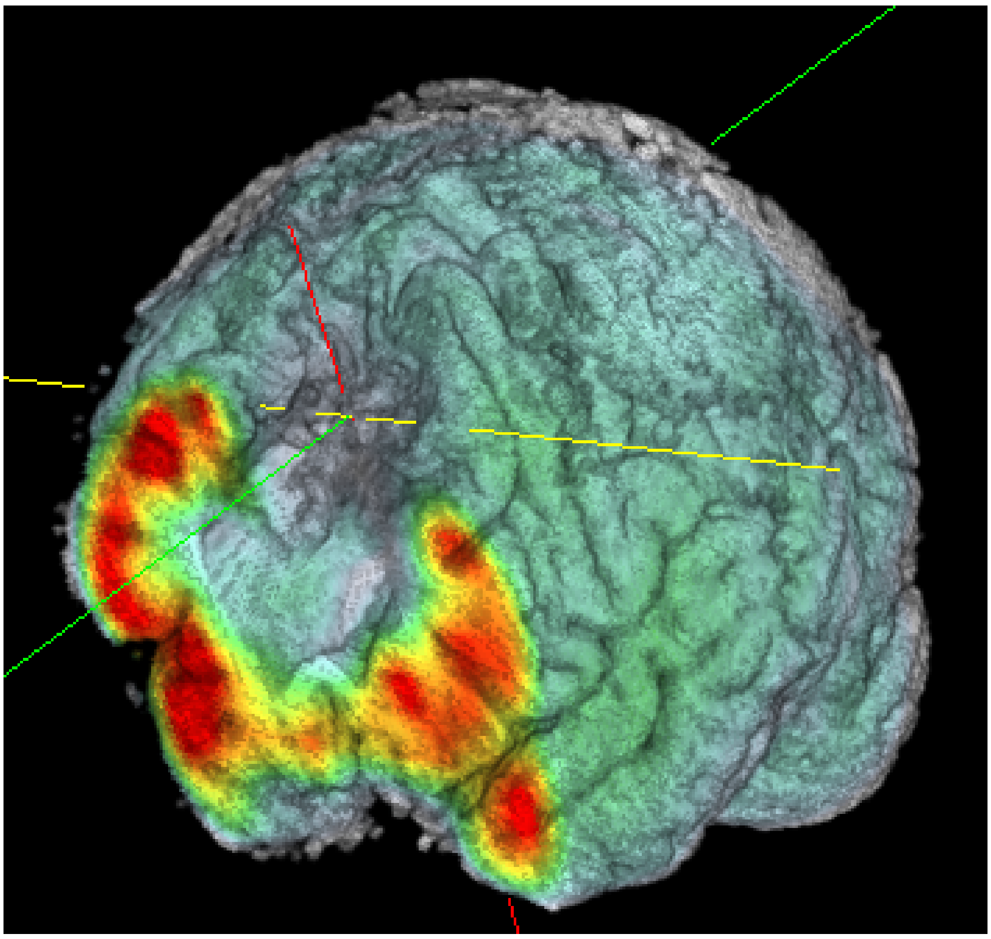


Figure 5.17: The same data sets as in figure 5.16 jointly rendered in 3D using dependent rendering. The position of the 3D cursor corresponds to the position of the slices in the 2D view in figure 5.16.

The requirements can be grouped into requirements that are mostly of a technical nature and can be tested objectively, and requirements where the evaluation is more subjective.

First, the objectively confirmable requirements are considered:

- **No need for segmentation:** This implementation assumes the reference structure to contain only a segmented brain, but the brain does not have to be further segmented (e.g. the ventricles or the cortex do to not have to be segmented).
- **Interactivity:** This implementation allows for interactive use by employing different magnifications and levels of detail: Images with lower quality are rendered interactively, while large and detailed images take longer.
- **Any property can be an entire volume:** This implementation assumes that both volumes are entire volumes.
- **Data sets of different sizes and resolutions must be applicable.** This implementation can handle data sets of different sizes and resolutions (MRI data sets usually have a higher resolution than PET images), but both volumes must be interpolated onto a common grid.
- **Arbitrary properties:** This is a specialised implementation which assumes an MRI data set describing the anatomy of the brain as reference volume and a PET or fMRI data set describing functional information as dependent volume. Nevertheless, this implementation could also be used for other combinations of properties, but some features (e.g. fixed colour map, the special window/level function) might then be useless.

So, for these technical requirements, it can be concluded that all are fulfilled, considering that some requirements (arbitrary properties, no need for segmentation) have been adapted to fit the users' demands for this specialised implementation.

The evaluation of the requirements that describe what should be perceivable from the rendered images is partially subjective. The following is the subjective assessment of the author, which may be biased due to his involvement in the development of the proposed algorithm. Different users could also come to different conclusions for the same rendered image because they have different backgrounds or are more or less experienced or trained. Techniques for a more reliable assessment of these requirements are suggested below.

- **Localisation:** Features found in the 3D rendering can be localised in terms of surrounding anatomical landmarks (e.g. the ventricles), as can be seen in figure 5.14 To further improve the localisation of features found in the 3D rendering, the 3D cursor can be moved interactively to the point of interest. The cursor position is synchronised with the cursor and the position of the slices in the 2D ortho-viewer, which allows for a high-precision localisation. This can easily be recognised in the example rendering in figures 5.16 and 5.17.
- **Relations between properties:** As both properties are visualised jointly, features found in one property can directly be localised relative to features in the other property, as shown in figures 5.14, 5.15 and 5.17.

- **3D impression:** The 3D renderings shown in this section also indicate that the algorithm can generate images that provide a good impression of the 3D structure of the object.

These last three requirements are concerned with the quality of the rendered images in terms of the insight a user can draw from them. To determine the quality of an image, there are two common approaches [Pra91]:

The image *fidelity* or *technical image quality* is determined by comparing the rendered image to an ideal image of the object under study. An example is comparing renderings of CT data directly to the object under study or to photographs thereof [HT85]. The comparison to an ideal image can either be performed in an objective way by using a mathematically defined measure, or in a more subjective way by visual comparions performed by human users [MHB+00]. This approach is not applicable for two-dimensional volumes as there exists no ideal joint image (like a photograph) of two properties of an object: The human eye can only perceive one property (the light reflected by a matter distribution), but not the other one (e.g. brain activity).

The *intelligibility* or *diagnostic image quality* indicates how well a user can recognise relevant findings in the image. The evaluation of the intelligibility depends not only on the rendered image itself, but also on the user, and may therefore be partially subjective. To determine the intelligibility in an objective manner, blind studies with a large number of expert users have to be conducted [PH02, PH03]. In the following, a possible evaluation of the diagnostic image quality of the dependent rendering algorithm is outlined:

First, an application domain has to be chosen (e.g. functional brain imaging) and the test questions have to be defined (e.g. "Determine the locations of abnormal functional values in the brains", where the term "abnormal" has to be clearly defined). Then, test data sets have to be prepared, either by collecting real data sets or by using simulated or manipulated data. For each test data set, the correct answers to each test question have to be known, such that the test data sets can be used as a ground truth.

The test has to be performed by a significant number of experts. A possibility is to start with a relative small number of experts and then add experts until the result statistics do not change significantly any more, i.e. a saturation is reached [Mer03].

For analysing the tests, for example receiver operator characteristics (ROC) can be determined. A ROC consists of a graph plotting the rate of true positive decisions (e.g. an existing tumor is identified as a tumor) against the rate of false positive decisions (e.g. a tumor is identified where there is none). To obtain a measure for the accuracy of the test, the area under the curve (AUC) for the ROCs can be determined. The AUC for a ROC is a positive number between 0.0 and 1.0, with 1.0 indicating perfect diagnostic quality and 0.5 indicating a worthless diagnaostic quality.

Additionally, it could be measured how well an implementation of the dependent rendering algorithm satisfies the demands of users in a specific application domain. Such quality assurance is, for example, part of QFD and the *Volere* framework (see section 4.2). To do so, qualified users from this domain would first be asked to rate the importance of the requirements derived in section 4.2 and possibly add new requirements (as described in section 5.2.4). Afterwards, they would be asked how well the implementation fulfils their expectations. This could be rated, for example using a four-point Likert scale (e.g. “not satisfied at all” – “hardly satisfied” – “quite satisfied” – “completely satisfied”) [Rob93]. By weighting the level of fulfilment by the importance of the requirement, it can be determined how well the implementation fits the users’ needs.

Such user studies have not been conducted here, for two reasons: First, user studies with expert users are very costly, both in terms of time as well as financial funding, and especially the last one was not available. Second, the purpose of the work presented in this thesis was the development of a new technique, and the results of user studies “give little or no clues on how to choose parameters, or improve imaging and visualization procedures.” [PH02, p. 600].

To conclude, this implementation indicates that the dependent rendering algorithm is a powerful technique to visualise two-dimensional volumes. Especially, using a 3D cursor synchronised to a conventional 2D slice viewer allows high precision for localising features found in the 3D rendering. In order to obtain an objective evaluation, an assessment strategy involving expert users has been outlined.

Chapter 6

Conclusion and Outlook

In this chapter, the achievements of this work are shortly presented and an outlook to possible improvements of the algorithm and new fields of research is given.

6.1 Conclusion

A new algorithm for the joint 3D visualisation of two properties of an object has been developed. This algorithm is called *dependent rendering algorithm*, because one of the properties is visualised in dependency on the other. This algorithm has been implemented first in two prototypes to test its general viability. Then, it has been specialised for a biomedical imaging application. For this specialisation, additional a-priori knowledge about the expected data has been used to improve the visualisation.

The algorithm has already been evaluated together with the implementations in sections 5.1.1.3, 5.1.2.3 and 5.2.5. These evaluations have shown that the requirements formulated initially in section 4.2 are met. The evaluation of the algorithm and all three implementations can be summarised as follows:

- **Localisation:** As both properties are visualised jointly, the observer can localise features in the dependent property in terms of references to the reference property. The precision of the localisation of features in the reference property is mainly determined by the quality of the rendering, which in turn depends on the choice of appropriate volume rendering transfer functions. Additionally, the implementation for the biomedical imaging platform provides an interactive 3D cursor whose position is also displayed in the 2D interface to allow a very precise localisation of features (see section 5.2.4 and figure 5.17).

- **3D-impression:** The renderings allow an easy mental 3D reconstruction of the scene. The general 3D impression given by the renderings can be further improved by using a cutting cube as described in section 5.2.4.

- **Arbitrary properties:** The algorithm allows the use of arbitrary combinations of properties and makes no a-priori assumptions on the properties used. However, the implementation for the biomedical imaging platform described in section 5.2 has been specialised for biomedical imaging with a combination of anatomical data as reference data and functional data as dependent data.

- **Exploration:**
  - **No need for segmentation:** The algorithm does not rely on segmented data. However, the additional requirements for the implementation for the biomedical imaging platform (see section 5.2.4) expect a segmented brain as reference data set.
  - **Interactivity:** The interactivity of the algorithm highly depends on the implementation and the quality of the images. While the initial prototype can clearly not be used interactively, the shear-warp prototype can be used interactively. For the implementation for the biomedical imaging platform, different zoom and image quality levels can be selected, such that low to medium quality images can be rendered interactively, while rendering high quality images takes longer. The prototype implementation can be parallelised and the IDL volume render used for the biomedical imaging platform implementation already is parallelised, so using multi-processor computers can further improve the interactivity.

  Although it is possible to use this algorithm to explore unknown data sets, a-priori knowledge of the general structure of the data sets is recommended because this facilitates the recognition of features.

- **Any property can be an entire volume:** In contrast to the existing 3D visualisation techniques for multiple properties of an object (see section 3.3.3), the dependent rendering technique not only supports but is explicitely designed to handle several non-sparse volumes with the same extension. This is an outstanding novelty of the proposed technique.

- **Relations between properties:** The spatial relations between the properties are clearly visible because both properties are displayed simultaneously.

- **Data sets of different size and resolution:** The algorithm imposes only two restrictions on the size and resolution of the data sets used: First, no dependent data set must have a larger spatial extension than the reference data set, because otherwise there would be regions of the dependent data without spatial reference. Second, there must be a mapping from the grid points in each data set to the grid points in the other data sets. However, the implementations use data sets that have the same size and resolution and that are registered with one another, because this simplifies the implementation. Where data sets had a different size and resolution (e.g. PET data sets usually have a lower resolution than MRI data sets) the data sets with the lower resolution have simply been mapped to the grid structure of the data set with the high resolution.

The dependent rendering algorithm can also be regarded as a framework for the visualisation of two-dimensional volumes. As a framework, the dependent rendering technique describes how to jointly visualise two volumes using volume rendering or similar volume visualisation techniques. This aspect of the dependent rendering technique has already been described in detail elsewhere ([LM02a]).

Hence, the achievement of the work presented in this thesis is the development of an algorithm, namely the dependent rendering algorithm, that allows the joint 3D visualisation of two properties of an object. In contrast to existing techniques, this algorithm is not restricted to specific properties. And most important, this algorithm allows that the second property is extended over the whole range of the first property.

6.2 Outlook

There are three major areas for further research related to the work presented in this thesis.

6.2.1 A field study

The evaluations of the implementations of the dependent rendering algorithm in this thesis suggest that the technique has advantages over the classical 2D visualisation. However, it would have exceeded the scope of this work to conduct a field study in order to determine the diagnostic quality of the proposed algorithm.

To find out whether the technique is useful for daily diagnostic use and which parts need to be further improved, it is suggested to conduct a broader field study. A possible evaluation procedure has been proposed in section 5.2.5.

6.2.2 Application to other domains

The dependent rendering algorithm itself is generic and not bound to any specific application. However, the final implementation described in section 5.2 represents a specialisation to the domain of biomedical imaging and has been further specialised to the visualisation of anatomical and functional data, with the anatomical data coming from MRI and the functional one from PET. It would be interesting to apply the algorithm to other domains, such as fluid dynamics or heat flow research, and to determine whether it can be used as-is or whether and how it needs to be specialised.

6.2.3 Improved image fusion

While the fusion of two images in order to enhance the understanding of one property of an object is an active field of research, there is little systematic research and literature about how to merge images in order to simultaneously understand two different properties ([RSA$^+$94] proposes an pixel interleaving technique to solve this problem, but encounters a camouflage effect which makes this technique difficult to use). An example of the former is the fusion of visible-wavelength photographs with infrared photographs in order to improve the perception of shapes of objects. This is used e.g. for navigation guidance in aviation, where the goal is to obtain the best perception of the shapes of objects (see e.g. [Sha99]): The observer does not need to discern information stemming from visible wavelength photographs from information stemming from infrared photographs.

That implies that information from one modality is allowed to hide information from the other modality. However, for the work presented in this thesis, it is essential that the user can exactly tell the source of each piece of information, e.g. to discern anatomical from functional information.

Another well-researched question is how to discern discrete chunks of information from a background and on how to mark discrete sets of items by different colours or markers to discern as many different sets as quickly as possible (see e.g. [Hea96] or, for a general introduction, [War99]). An application domain for this fusion of data from different sources is the integration of whether forecast data with other data like infrastructure maps, which is discussed in [Tre00]. Yet, the findings from this field cannot be applied here as the values of both properties are changing smoothly and are distributed over the whole scene.

In the final implementation described in section 5.2, this problem has been solved by creating the final image in the HSV colour space and encoding the grey level values of one input image in the HSV *value* channel and the other in both the HSV *hue* and *saturation* channel. This approach works, but still has deficiencies: The human perception for changes is not the same for these three channels and furthermore is not linear in any of the channels. Supposed both source images are normalised to the range [0..1], this has two implications: First, a change of 0.1 in one source image is perceived differently in the final image than the same 0.1 change in the other input image. Second, a change of 0.1 between 0.1 and 0.2 is perceived differently from the 0.1 change between 0.8 and 0.9. These deficiencies might be compensated for by using normalisation functions or by using a normalised colour space like CIE Luv (for an explanation of different colour spaces used in computer graphics, see [FvDFH96, pp. 563–603]). For a discussion about the human perception of light intensity and colour see [War99, pp. 73–148]. For images of a single property, the problem of finding adequate colour maps has been discussed by e.g. [LH92] or [RT96]. Recent discussions about the evaluation of the intelligibillity of application domain specific information of different image fusion techniques can be found in [TF03] and [PSKR01].

Another, much more distorting problem is the fact that the change in one HSV channel affects the perception of the other channels. For example, if one source image sets the HSV *hue* channel to 240 and the HSV saturation channel to 100%, the perception of the colour varies together with the HSV *value* channel set by the other input image from black (*value* = 0%) to pure blue (*value* = 100%). This means that the perception of the two channels is not separated.

Thus, the problem to solve is: How can the information of two distinct source images be simultaneously displayed in one final image, given that each source image contains smoothly changing grey level values, and information from both images must be clearly perceptible and discernable at all locations in the final image with the perception of one input image not affecting the perception of the other? To solve this problem, the development of a colour space that allows at least two channels that are perceived separately and with scales for the channels that change proportionally to their perception is proposed.

Bibliography

[CMP99] Steven Conolly, Albert Macovski, and John Pauly. Magnetic resonance imaging. In Joseph D. Bronzino, editor, *The Biomedical Engineering Handbook*, volume 1, chapter 63.1. CRC Press, second edition, 1999.

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, August 1988.

[Dim95] Leonid I. Dimitrov. Pseudo-colored visualization of EEG-activities on the human cortex using MRI-based volume rendering and Delaunay interpolation. In Yongmin Kim, editor, *Medical Imaging 1995: Image Display*, volume Proc. SPIE 2431, pages 460–469, San Diego, California, 1995. SPIE—The International Society for Optical Engineering.

[DLB$^+$99] Manuel Desco, José López, Carlos Benito, Andrés Santos, P. Domínguez, Santiago Reig, Celso Arango, and Pedro G. Barreno. A multimodality workstation in practice. In H.U. Lemke, M.W. Vannier, K. Inamura, and A.G. Farman, editors, *Proceedings Computer Assisted Radiology and Surgery 1999*, pages 218–222. Elsevier, June 1999.

[DNR90] Shane Dunne, Sandy Napel, and Brian Rutt. Fast reprojection of volume data. In *Proceedings of the 1st Conference on Visualization in Biomedical Computing*, pages 11–18. IEEE, 1990.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practise.* The systems programming series. Addison-Wesley, second in C edition, 1996.

[FvKS03] Uwe Flick, Ernst von Kardorff, and Ines Steinke, editors. *Qualitative Forschung. Ein Handbuch.* Rowohlt, Hamburg, second edition, 2003.

[GNK$^+$99] David Gering, Arya Nabavi, Ron Kikinis, Eric Grimson, Noby Hata, Peter Everett, Ferenc Jolesz, and William Wells III. An integrated visualization system for surgical planning and guidance using image fusion and interventional imaging. In *Proceeding of Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 809–819, September 1999.

[GNK$^+$01] David Gering, Arya Nabavi, Ron Kikinis, Noby Hata, Lauren ODonnell, Eric Grimson, Ferenc Jolesz, Peter Black, and William Wells III. An integrated visualization system for surgical planning and guidance using image fusion and an open MR. In *Journal of Magnetic Resonance Imaging*, volume 13, pages 967–975, June 2001.

[Gou71] Henri Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623–629, 1971.

[Hea96] Christopher G. Healey. Choosing effective colours for data visualization. In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 263–270, 1996.

[HP96] Karl Heinz Höhne and Andreas Pommert. Volume visualization. In Arthur W. Toga and John C. Mazziotta, editors, *Brain Mapping: The Methods*, chapter 17, pages 423–443. Academic Press, San Diego, CA, 1996.

[Hru97] Peter Hruschka. Detailing and deriving system requirements. In *Proceedings of the International Conference and Workshop on Engineering of Computer-Based Systems*, pages 25–32, March 1997.

[HSM99] Georg Herzwurm, Sixten Schockert, and Mellis. Higher customer satisfaction with prioritizing and focused software quality function deployment. In *Proceedings of the Sixth European Conference on Software Quality, April 12-16, Vienna, Austria*, April 1999.

[HT85] David C. Hemmy and Paul L. Tessier. CT of dry skulls with craniofacial deformities: Accuracy of three-dimensional reconstruction. *Radiology*, 157(1):113–116, 1985.

[Inc] Kitware Inc. The visualization toolkit. URL: http://www.vtk.org.

[JFS$^+$00] Pierre Jannin, Oliver J. Fleig, Eric Seigneuret, Xavier Morandi, Mélanie Raimbault, and Jean–Marie Scarabin. Multimodal and multi-functional neuronavigation. In H.U. Lemke and al. Editors, editors, *Computer Assisted Radiology and Surgery*, pages 167–172, 2000.

[Kau91] Arie Kaufman. *Volume Visualization.* IEEE Computer Society Press, first edition, January 1991.

[KCY93] Arie E. Kaufman, Daniel Cohen, and Roni Yagel. Volume graphics. *IEEE Computer*, 26(7):51–64, 1993.

[KDG99] Andreas König, Helmut Doleisch, and Eduard Gröller. Multiple views and magic mirrors - fMRI visualization of the human brain. In *Proceedings of Spring Conference on Computer Graphics and its Applications 1999 (SCCG'99)*, pages 130–139, 1999.

[Lac] Philippe Lacroute. VolPack volume rendering library. URL: http://graphics.stanford.edu/software/volpack/. Version 1.0beta3.

[Lac95] Philippe Lacroute. *Fast volume rendering using a shear-warp factorization of the viewing transformation.* PhD thesis, Stanford University, Stanford, CA, USA, 1995.

[Lac96] Philippe Lacroute. Analysis of a parallel volume rendering system based on the shear-warp factorization. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):218–231, September 1996.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[LCN98] Barthold Lichtenbelt, Randy Crane, and Shaz Naqvi. *Introduction to Volume Rendering*. Hewlett-Packard professional books. Prentice Hall, Upper Saddle River, New Jersey, first edition, 1998.

[Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 5(4):29–37, May 1988.

[Lev90] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[LH92] Haim Levkowitz and Gabor T. Herman. Color scales for image data. *IEEE Computer Graphics and Applications*, 12(1):72–80, January 1992.

[Lic95] Barthold B.A. Lichtenbelt. Fourier volume rendering. Technical Report HPL-95-73, Hewlett Packard Laboratories, July 1995.

[LL94] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics*, 28(4):451–458, August 1994.

[LM02a] Sebastian Löbbert and Steffen Märkle. Dependent rendering: Visualizing multiple properties of an object. In Alfredo Colosimo, Alessandro Giuliani, and Paolo Sirabella, editors, *Medical Data Analysis. Proceedings of ISMDA 2002*, volume 2526 of *Lecture Notes in Computer Science*, pages 198–209. ISMDA, Springer Verlag, Berlin, Heidelberg, October 2002.

[LM02b] Sebastian Löbbert and Steffen Märkle. An extension of the shear-warp volume rendering algorithm for the visualization of multiple properties of an object. In J.J. Villanueva, editor, *Proceedings of the Second IASTED International Conference on Visualization, Imaging, and Image Processing, held September 9-12, 2002 in Malaga, Spain*, pages 514–518. IASTED, September 2002.

[LPDS03] Sebastian Löbbert, Javier Pascau, Manuel Desco, and Andrés Santos. Advanced joint 3d visualization of functional and anatomical data. In H.U. Lemke, M.W. Vannier, K. Inamura, A.G. Farman, K. Doi, and J.H.C. Reiber, editors, *Proceedings of Computer Assisted Radiology and Surgery 2003*, page 1305. Elsevier, June 2003.

[MA94] Shigeru Mizuno and Yoji Akao. *Quality Function Deployment: The Customer-Driven Approach to Quality Planning and Deployment*. Productivity Press Inc., April 1994.

[Mal93] Tom Malzbender. Frequency volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[Max95] Nelson L. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

[Mer03] Hans Merkens. *Auswahlverfahren, Sampling, Fallkonstruktion*, chapter 4.4, pages 286–299. In Flick et al. [FvKS03], second edition, 2003.

[MFOF02] Isabel Harb Manssour, Sérgio Shiguemi Furuie, Sílvia D. Olabarriaga, and Carla Maria Dal Sasso Freitas. Visualizing inner structures in multimodal volume data. In *15th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2002), 7-10 October 2002, Fortaleza-CE, Brazil*, pages 51–58. IEEE Computer Society, 2002.

[MHB$^+$00] Michael Meißner, Jian Huang, Dirk Bartz, Klaus Mueller, and Roger Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 Symposium on Volume Visualization*, pages 81–90. ACM Press, 2000.

[MHG00] Lukas Mroz, Helwig Hauser, and Eduard Gröller. Interactive high-quality maximum intensity projection. In *Computer Graphics Forum*, volume 19(3), August 2000.

[MS] MIT Artificial Intelligence Lab and Surgical Planning Lab at Brigham & Women's Hospital. Slicer web page. URL: http://www.slicer.org. Downloaded 2003-11-10.

[MS98] Steffen Märkle and Axel Spikermann. Distributed visualization. how to improve the quality of medical volume rendering at almost no costs. In Joaquim Piqueras and Joan-Carlos Carreño, editors, *Proceedings of EuroPACS'98*, pages 225–228. EuroPACS-Society, October 1998.

[MWG98] Thomas Malzbender, Craig M. Wittenbrink, and Michael E. Goss. Opacity-weighted color interpolation for volume sampling. In *Proc. 1998 Symposium on Volume Visualization*, pages 135–142, New York, October 1998. ACM. Also as HP Labs Technical Report HPL-97-31.

[NBC$^+$00] Stavri G. Nikolov, Dave R. Bull, Nishan Canagarajah, Mike Halliwell, and Peter N. T. Wells. Fusion of 2-d images using their multiscale edges. In *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*. IEEE, 2000.

[Oes01] Bernd Oestereich. *Objectorientierte Softwareentwicklung. Analyse und Design mit der Unified Modeling Language*. Oldenburg, München, Wien, 5th edition, 2001.

[PD84] Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, July 1984.

[PEM03] Frauke Paetsch, Armin Eberlein, and Frank Maurer. Requirements engineering and agile software development. In *Proceedings of the Twelfth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria*, pages 308–313. IEEE, June 2003.

[PH02] Andreas Pommert and Karl Heinz Höhne. Evaluation of image quality in medical volume visualization: State of the art. In Takeyoshi Dohi and Ron Kikinis, editors, *Proceeding of Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2002, Part II*, number 2489 in Lecture Notes in Computer Science, pages 598–605. MICCAI, Springer, 2002.

[PH03] Andreas Pommert and Karl Heinz Höhne. Validation of medical volume visualization: A literature review. In H.U. Lemke, M.W. Vannier, A.G. Farman, K. Doi, and J.H.C. Reiber, editors, *Proceedings of Computer Assisted Radiology and Surgery 2003*, pages 571–576. Elsevier, June 2003.

[PHK$^+$99] Hanspeter Pfister, Jan Hardenbergh, Jim Knittel, Hugh Lauer, and Larry Seiler. The VolumePro real-time ray-casting system. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 251–260, Los Angeles, 1999. Addison Wesley Longman.

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

[Pra91] William K. Pratt. *Digital Image Processing*. John Wiley and Sons, second edition, 1991.

[PSKR01] Nupoor Prasad, Sameer Saran, S. P. S. Kushwaha, and P. S. Roy. Evaluation of various image fusion techniques and imaging scales for forest features interpretation. *Current Science*, 81(9):1218–1224, November 2001.

[RDC96] C. Rocha, Jean–Louis Dillenseger, and Jean–Louis Coatrieux. Multi-array EEG signals mapped with three-dimensional images for clinical epilepsy studies. In Karl Heinz Höhne and Ron Kikinis, editors, *Proceedings of the 4th International Conference on Visualization in Biomedical Computing 1996*, number 1131 in Lecture Notes in Computer Science, pages 467–476. Springer, 1996.

[RKC$^+$01] Nils Riefenstahl, Gerald Krell, Roman Calow, Bernd Michaelis, and Mathias Walke. A multimodal image fusion framework applied in radiotherapy. In *Proceedings of the Fifth International Conference on Information Visualisation (IV'01)*, pages 173–180. IEEE Computer Society, 2001.

[RLF$^+$98] Kelly Rehm, Kamakshi Lakshminaryan, Sally Frutiger, Kirt A. Schaper, De Witt Sumners, Stephen C. Strother, Jon R. Anderson, and David A.

Rottenberg. A symbolic environment for visualizing activated foci in functional neuroimaging datasets. *Medical Image Analysis*, 2(3):215–226, September 1998.

[Rob93] Colin Robson. *Real world research.* Blackwell Publishers, 1993.

[RR99] James Robertson and Suzanne Robertson. *Mastering the Requirements Process.* Addison-Wesley, London, 1999.

[RSA$^+$94] Kelly Rehm, Stephen C. Strother, Jon R. Anderson, Kirt Schaper, and David A. Rottenberg. Display of merged multimodality brain images using interleaved pixels with independent color scales. *Journal of Nuclear Medicine*, 35:1815–1821, 1994.

[RT96] Bernice Rogowitz and Lloyd Treinish. How not to lie with visualization. *Computers in Physics*, 10:268–274, May/June 1996.

[RT01] Theresa-Marie Rhyne and Lloyd Treinish. The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–22, May/June 2001.

[Sch01] William J. Schroeder, editor. *The Visualization Toolkit User's guide, Updated for Version 4.0.* Kitware Inc., 2001.

[Sha99] Ravi K. Sharma. *Probabilistic Model-based Multisensor Image Fusion.* PhD thesis, Graduate Institute of Science and Technology, Portland, Oregon, USA, 1999.

[SLPB96] D. Schwartz, D. Lemoine, E. Poiseau, and C. Barillot. Registration of MEG/EEG data with 3d MRI: Methodology and precision issues. *Brain Topography*, 9:159–166, 1996.

[SPHW02] Bernhard Sturm, Kimerly A. Powell, Sandra Simon Halliburton, and Richard D. White. Image fusion of 4D cardiac CTA and MR images. In *30th Applied Image Pattern Recognition Workshop (AIPR 2001), Analysis and Understanding of Time Varying Imagery, 10-12 October 2001, Washington, DC, USA, Proceedings*, pages 21–24. IEEE Computer Society, 2002.

[TF03] Alexander Toet and Eric M. Franken. Perceptual evaluation of different image fusion schemes. *Displays*, 24(1):25–37, 2003.

[TL93] Takashi Totsuka and Marc Levoy. Frequency domain volume rendering. In *Proc. SIGGRAPH '93*, pages 271–278. ACM, August 1993.

[TOW$^+$03] Florin Talos, L. O'Donnell, C.-F. Westin, Simon Warfield, Willian Wells III, S.S. Yoo, L. Panych, A. Golby, H. Mamata, S. Maier, P. Ratiu, C. Guttmann, P. Black, Ferenc Jolesz, and Ron Kikinis. Diffusion tensor and functional MRI fusion with anatomical MRI for image-guided neurosurgery. In *Sixth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'03)*, pages 407–415, Montreal, Canada, November 2003.

[Tre00] Lloyd Treinish. Visual data fusion for applications of high-resolution numerical weather prediction. In *Proceedings of the IEEE Computer Society Visualization 2000*, pages 477–480, October 2000.

[TT88] Jean Talairach and Pierre Tournoux. *Co-Planar Stereotaxic Atlas of the Human Brain.* Thieme Medical, New York, 1988.

[VK96] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. In *Proceedings of the 1996 Symposium on Volume Visualization*, pages 23–ff. IEEE Press, 1996.

[Vog97] Helmut Vogel. *Gerthsen Physik.* Springer, 19th edition, 1997.

[War99] Colin Ware. *Information Visualization: Design for Perception.* Academic Press, first edition, 1999.

[Wes89] Lee Westover. Interactive volume rendering. In *Proceedings of the Chapel Hill Workshop on Volume Visualization*, pages 9–16, May 1989.

[WTT$^+$02] Simon K. Warfield, Florin Talos, Alida Tei, Aditya Bharatha, Arya Nabavi, Matthieu Ferrant, Peter Mc. Black, Ferenc A. Jolesz, and Ron Kikinis. Real-time registration of volumetric brain MRI by biomechanical simulation of deformation during image guided neurosurgery. In *Computing and Visualization in Science*, volume 5(1), pages 3–11, Heidelberg, 2002. Springer.

[Yag93] Roni Yagel. Volume viewing: State of the art survey. ACM SIGGRAPH '93 Volume Visualization Course Notes, 1993.