Florian Lindemann

Using Advanced Illumination Techniques to Enhance Realism and Perception of Volume Visualizations

Münster • 2013

Informatik

Using Advanced Illumination Techniques to Enhance Realism and Perception of Volume Visualizations

Inauguraldissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften durch den Fachbereich Mathematik und Informatik der Westfälischen Wilhelms-Universität Münster

vorgelegt von

Florian Lindemann

aus Marl

2013

Typeset with LaTeX2e and KOMA-Script using the fonts Palatino, Bera Sans, and Bera Mono.

Dekan:	Prof. Dr. Martin Stein
Erster Gutacher:	Prof. Dr. Klaus Hinrichs
Zweiter Gutacher:	Prof. Dr. Xiaoyi Jiang
Tag der mündlichen Prüfung:	21.01.2014
Tag der Promotion:	21.01.2014

Abstract

The use of volumetric data has become more and more important and frequent in recent years. Sources for volumetric data now include many scientific branches, from medicine and biology to physics and geology. The generation of meaningful and comprehensible renderings from volumetric data is therefore more important than ever. The simulation of illumination phenomena is one way of improving the perception and realism of volume renderings. This dissertation concerns itself with the effectiveness of existing volume illumination techniques and presents some novel approaches and applications for this branch of computer graphics. First, we introduce some methods to simulate the interaction of light and material in the context of volumetric data and to reproduce phenomena such as subsurface scattering. Furthermore, we present the results of a comprehensive user study in which we examined the influence of seven different existing volume illumination models on the viewer. For this purpose, we conducted and evaluated four series of tests on a group of 55 participants aimed at different aspects of image perception. Finally, we introduce a framework for the efficient visual analysis of multimodal images of the human brain, in which volume illumination is employed along with a number of novel visualizations to improve the spatial comprehension of the generated images.

Kurzfassung

Die Nutzung volumetrischer Daten ist in vergangenen Jahren immer wichtiger und häufiger geworden. Volumetrische Daten stammen inzwischen aus einer Vielzahl von wissenschaftlichen Bereichen, angefangen bei der Medizin und der Biologie bis hin zur Physik und zur Geologie. Die Erzeugung von aussagekräfigen und verständlichen Bildern aus diesen Daten ist daher wichtiger denn je. Die Simulation von Beleuchtungsphänomenen ist eine Möglichkeit, die Wahrnehmung und den Realismus solcher Bilder zu verbessern. Diese Dissertation beschäftigt sich mit der Effektivität von existierenden Modellen zur Volumenillumination und präsentiert einige neue Techniken und Anwendungen für diesen Bereich der Computergrafik. Wir stellen zunächst einige Methoden vor, um die Interaktion von Licht und Material im Kontext von Volumendaten zu simulieren und Beleuchtungsphänomene wie Volumenstreuung nachzubilden. Weiterhin werden die Ergebnisse einer umfangreichen Nutzerstudie präsentiert, deren Ziel es war, den Einfluss von sieben verschiedenen existierenden Modellen zur Volumenillumination auf den Betrachter zu untersuchen. Dazu wurden bei einer Gruppe von 55 Teilnehmern vier Testserien zu verschiedenen Aspekten der Bildwahrnehmung durchgeführt und ausgewertet. Abschließend wird eine Anwendung zur Darstellung und visuellen Analyse multimodaler Hirndaten präsentiert, in der Volumenillumination neben einigen weiteren neuartigen Visualisierungen zum Einsatz kommt, um das räumliche Verständnis der generierten Bilder zu verbessern.

Contents

Preface

However vast the darkness,
we must supply our own light.

(Stanley Kubrick)

This dissertation represents the work and research carried out during the time between April 2010 and November 2013 at the Visualization and Computer Graphics Research Group at the Department for Computer Science of the University of Münster. I would like to thank my supervisor Prof. Dr. Klaus Hinrichs for the opportunity to engage in this research and for his professional feedback and guidance during my time as a member of his group. I would also like to thank Prof. Dr. Timo Ropinski for his support and our discussions which inspired much of the work presented in this dissertation. Furthermore, I would like to express my thankfulness towards all current and former members of the Voreen development team, especially Tobias Brix, Stefan Diepenbrock and Jörg-Stefan Praßni, for providing a friendly and productive working atmosphere.

This work was partly funded by grants from the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) through the Collaborative Research Center 656 - Molecular Cardiovascular Imaging (projects Z1 and Ö). I would like to thank its members for our prolific collaboration.

Without the support by my parents, Martina and Siegfried, and my girlfriend Nina, this dissertation would not have been possible. I would like to extend my deepest gratitude to them for their continuous support over the years.

Münster, November 2013 *Florian Lindemann*

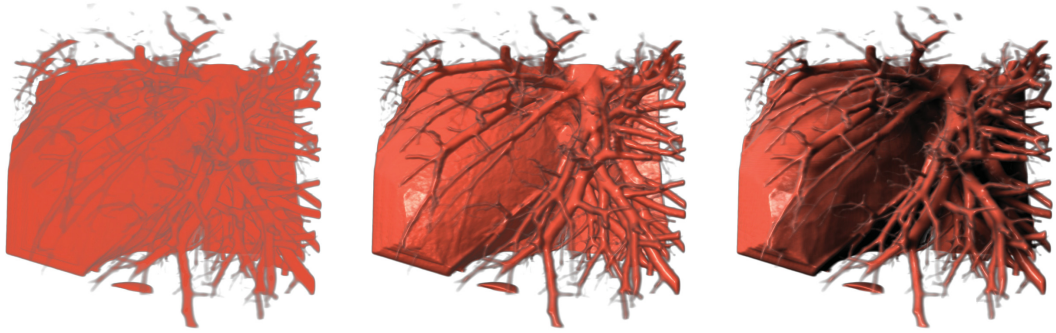
Chapter 1

Introduction

Volume rendering is a branch of computer graphics that has grown extensively in the last decade, due to the increased demand for algorithms and techniques that allow the interactive generation of high-quality, comprehensible images from volume data sets that are now typically several gigabytes in size. This process is not only challenging due to the amount of data that have to be visualized, but also because rendering concepts that have been established and validated for traditional graphics that are based on geometric primitives have to be either adapted or even newly invented for volume rendering. One of those concepts which has been transferred to volume rendering is the illumination of a scene, which in this context is called *volume illumination*.

Volume illumination deals with the simulation of the influence of light sources on a given volume rendering scene. Existing volume illumination techniques range from the implementation of simple single-reflection models to complex, non-interactive multi-scattering illumination frameworks. In general, volume illumination introduces an additional level of complexity into the generation of volume rendered images, often at the cost of increased computation time and memory resources. One incentive to accept this cost is the increase of the perceived realism of the generated images. Although many illumination phenomena that have already been simulated in traditional computer graphics have been successfully realized in volume rendering, there is still room for improvement concerning realistic, natural lighting.

Many researchers who have introduced volume illumination techniques in past years also hope to better communicate information contained in the original data to the viewer, for example by providing additional depth cues in volume rendered images with the help of their techniques. Figure 1.1 demonstrates the differences between no volume illumination, a simple volume illumination technique and an advanced volume illumination technique when applied to the same scene. It is obvious that due to the improved contrast introduced by surface-like shading, the simple technique in Figure 1.1(b) provides additional helpful information about the



(a) **(b)** **(c)**

Figure 1.1: A volume rendering scene with a scan of a human heart. The same scene is rendered with no volume illumination (a), a simple illumination model which emphasizes the different implicit surfaces contained in the dataset (b) and an advanced illumination model that extends the simple model by including shadows (c). Using the more sophisticated illumination technique, the complex structure of the blood vessels is easier to recognize.

structures in the dataset when compared to Figure 1.1(a). The blood vessel system within the dataset can be distinguished even better considering Figure 1.1(c) in which shadows are cast across the scene. However, it would be premature to assume that all volume rendering scenarios benefit from advanced illumination models. The question of how much these techniques improve the users perception of volume rendered images has so far received little attention. In particular, a comparison how the different features of the presented techniques influence the different aspects of human image perception would be useful.

Nowadays, in most volume rendering applications advanced volume illumination is not an established or regularly used feature. This may be due to the fact that volume illumination is a relatively new branch of research. Furthermore, the introduction of volume illumination to existing volume rendering software would require users to adapt to differently looking rendering results, which may initially impede the visual analysis process to which users are accustomed. Nevertheless, if evidence would be found indicating that volume illumination can positively influence the image perception of volume rendered images, it could be considered as an optional feature for the development of future volume rendering applications.

Following these considerations, this thesis addresses three different aspects of volume illumination research:

1. The further development of illumination techniques to extend the list of lighting phenomena that can be simulated in volume rendering.

2. The investigation of the actual influence of the use of volume illumination on the ability of users to comprehend volume rendered images.

3. The practical use of illumination in a real-world volume rendering application.

Goal of this thesis is to produce more aesthetically pleasing volume renderings by using volume illumination techniques, while at the same time increasing the amount of information that is conveyed to the users of volume rendering applications.

The remainder of this thesis is structured as follows: In Chapter 2, an introduction into the basic principles of volume rendering is presented. In Chapter 3, an overview of the different properties of volume illumination techniques is given, along with an introduction of several established techniques. Chapter 4 provides a description of the volume rendering software framework that was used to implement the different parts of this thesis. Chapter 5 addresses the first of the previous list of volume illumination research aspects, in which the simulation of advanced light-material interaction effects is described. The second aspect is discussed in Chapter 6, in which a comprehensive user study comparing several established volume illumination techniques is introduced. The third aspect is tackled in Chapter 7, in which an illumination technique operating in screen space is used to enhance renderings in a framework designed to visually analyze multimodal brain tumor segmentation data. Finally, a conclusion and outlook is given in Chapter 8.

The contributions that are part of this thesis are based on the following publications: Chapter 5 has been presented at the IEEE / EG International Symposium on Volume Graphics in 2010 [46]. Chapter 6 has been presented at the IEEE Visualization Conference in 2011 [43], and Chapter 7 at the Vision, Modeling, and Visualization Workshop in 2013 [45]. Further contributions to the field of volume visualization have been made as part of the winning entry of the IEEE Visualization Contest 2010 [14], which presented novel techniques for the pre-operative planning of surgery of the human brain, also represented in the associated publication in the IEEE Computer Graphics and Applications journal [15]. Moreover, contributions to the field of molecular visualization were made as part of the second place entry of the IEEE Visualization Contest 2012 [44]. As part of this thesis, significant extensions were made to the Voreen volume rendering framework [55].

Chapter 2

Concepts of Volume Rendering

In this chapter we present an overview of the basic concepts of direct volume rendering as well as two popular implementations of this paradigm.

Volume Rendering is a branch of computer graphics which aims to visualize three-dimensional scalar data. In contrast to traditional graphics, the basic building blocks of a scene do not consist of geometric primitives such as lines and polygons. Instead, one or several datasets are given which contain a field of scalar values called *voxels*. These voxels usually correspond to the intersection points of a 3D grid. While complex grid configurations exist (such as the arrangements of voxels on a tetrahedal grid), this thesis will focus on the more common case of grids with uniform spacing, forming a 3D array. For volume data presented in this form, it is easy to interpolate a value for any real-valued coordinates within the grid, essentially providing a mapping

$$\phi : \mathbb{R}^3 \to \mathbb{R}$$

from 3D space to the associated scalar domain.

Recent history has seen a growing number of sources from which such volumetric data sets emerge. One major source are medical examinations carried out to research and diagnose diseases. Traditional methods include *computed tomography* (CT), *magnetic resonance imaging* (MRI) and *ultrasound*, which have been extended more recently by techniques such as *positron emission tomography* (PET). Other sources of volumetric data include geographical surveys, microscopy data or numerical simulations of natural phenomena. The size of the resulting datasets and the contained amount of information have increased steadily over time. The challenge lies in the extraction of meaningful 2D renderings from these 3D volume datasets while incorporating as much of the original data as possible.

Traditional graphics often relies on the presence of given surface primitives such as triangles to represent complex objects. In volume rendering, the lack of these

primitives has to be compensated. One way to accomplish this is to extract so-called *isosurfaces* from volumetric datasets. After specifying a certain scalar *isovalue*, the scalar values of the volumetric grid are searched for transitions from values below the isovalue to values above the isovalue. Between these discrete grid values, the previously mentioned continuous mapping will yield a point where the scalar field is equal to the isovalue. The set of all these points forms the isosurface. An established method for the extraction of isosurfaces is the *marching cubes* algorithm, presented by Lorensen and Cline in 1987 [49]. The resulting surface can then be visualized using traditional polygonal rendering which is widely supported by rendering hardware. However, focusing on isosurfaces may result in a loss of information contained in other regions of the dataset where values are different from the isovalue. Furthermore, in datasets containing regions in which values vary strongly, a meaningful isosurface for a given value may not always exist at all. A more direct approach to the visualization of volume datasets which includes as much of the original data in the resulting rendering as possible is *direct volume rendering* (DVR). All volume rendering methods discussed in this thesis belong to this family of techniques and are extensions of its basic algorithms.

2.1 Basics of Direct Volume Rendering

In DVR, rendering is based on the physical behavior of light while it traverses a participating medium. In order to do so, a volume dataset represented by a 3D scalar grid is interpreted as a region filled with different participating media. As light traverses these regions, its optical properties are changed, depending on its path through the dataset. The traversal itself can be mathematically modeled by using an integral to sum up the contributions of different voxels in the dataset to each individual ray of light. In DVR, a sufficiently large number of these virtual rays are tracked from the back of dataset towards an observer's point of view to reconstruct a 2D image of the original dataset. In the remainder of this chapter, we will discuss the mathematical foundation as well as two of the most common implementations of DVR in detail.

2.1.1 The Volume Rendering Integral

The *volume rendering integral* is the basic mathematical equation on which DVR is based. It can be used to describe different optical models with varying complexity. However, for this thesis the following basic form of the integral will be sufficient. Following the notation of Engel et al. [16], to specify the light energy or *radiance I* of

a ray after traveling a distance D through the medium towards the viewer, starting at a point s_0 at the back of the dataset, we need to solve

$$I(D) = I_0 \mathrm{e}^{-\int_{s_0}^{D} \kappa(t)\mathrm{d}t} + \int_{s_0}^{D} q(s)\mathrm{e}^{-\int_{s_0}^{D} \kappa(t)\mathrm{d}t}\mathrm{d}s. \tag{2.1}$$

This equation models the so-called *emission-absorption model* of light transport [53]. It is a sum of two separate terms, the absorption term on the left and the emission term on the right. The absorption term expresses how an initial radiance I_0 is reduced on the ray's way through the medium, depending on the distance it has traversed and the absorption coefficient κ which is a property of the traversed medium. While the original energy of the light decays exponentially over distance due to this term, the ray may gain in radiance due to the emission term. For each position s on the ray, this term adds a certain emission $q(s)$, depending on the position of the ray, which is again multiplied by a weighting factor which decays exponentially with the accumulated absorption caused by the medium. This distance-dependent weighting term is called the *optical depth* τ of the medium:

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(t)\mathrm{d}t. \tag{2.2}$$

In a medium with high optical depth, light is absorbed fairly quickly, and vice versa. Subsequently, it is possible to base the more familiar term *transparency* on the optical depth:

$$T(s_1, s_2) = \mathrm{e}^{-\tau(s_1, s_2)} = \mathrm{e}^{-\int_{s_1}^{s_2} \kappa(t)\mathrm{d}t} \tag{2.3}$$

Using this definition, Equation 2.1 can be rewritten in a more intuitive form as

$$I(D) = I_0 T(s_0, D) + \int_{s_0}^{D} q(s)T(s, D)\mathrm{d}s. \tag{2.4}$$

Almost all of the advanced illumination techniques presented in this thesis use a modified version of this equation to simulate various illumination phenomena.

2.1.2 Discretization of the Volume Rendering Integral

Due to the complexity of typical volume datasets, it is in general not possible to solve Equation 2.4 analytically. To be able to find a numerical solution, the integral has to be split into discrete pieces which can then each be solved in a single step. In order to achieve this goal, the range of the integral can be divided into n sampling points between the start point s_0 and the end point D. At sampling point s_i, the sum of the

radiance can be expressed recursively as

$$I(s_i) = I_{s_{i-1}} \underbrace{T(s_{i-1}, s_i)}_{T_i} + \underbrace{\int_{s_{i-1}}^{s_i} q(s) T(s, s_i) \mathrm{d}s}_{c_i}. \tag{2.5}$$

In Equation 2.5, two abbreviations have been introduced: T_i for the transparency contribution and c_i for the color contribution in the i-th step. Expanding Equation 2.5 for $I(D)$ leads to

$$I(D) = I(s_n) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j, \text{with } c_0 = I(s_0). \tag{2.6}$$

This equation enables an iterative approach to the solution of the volume rendering integral. By approximating the integral with a Riemann sum consisting of segments of equal length $\Delta x = \frac{D - s_0}{n}$, it is possible to simulate the traversal of a ray of light through the volume dataset by iteratively evaluating the sum presented in Equation 2.6. For each summand, the color and transparency contributions can then be approximated by

$$T_i \approx \mathrm{e}^{-\kappa(s_i)\Delta x} \tag{2.7}$$

and

$$c_i \approx q(s_i)\Delta x. \tag{2.8}$$

In practice, the sum is calculated by updating it for each segment using a *compositing scheme*. The two main compositing schemes are *front-to-back* and *back-to-front* compositing, which differ in the direction in which the sum is evaluated: Either starting at the location of the virtual camera and traveling into the volume dataset, or traveling from the background towards the viewer. Usually, front-to-back compositing is preferred since it enables performance optimizations such as terminating the ray after its transparency value falls below a given threshold, thereby preventing the following segments from contributing to the final result.

For compositing, color is usually accumulated as an RGB-value C_i, which is equivalent to the previously used light radiance values I and c_i. Instead of transparency T_i, compositing uses accumulated opacity $\alpha_i = 1 - T_i$. By re-using the results of the previous step, a single variable for each of these two values can be updated with the current results of Equations 2.7 and 2.8: C_{dst} for the color and α_{dst} for the opacity. Using this notation and Equation 2.6, C_{dst} and α_{dst} must then be modified in each

step with the new values C_{src} and α_{src} as follows:

$$\begin{aligned} C_{dst} &\leftarrow C_{dst} + (1 - \alpha_{dst})C_{src}, \\ \alpha_{dst} &\leftarrow \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src}. \end{aligned} \tag{2.9}$$

While there are different approaches to implement the actual propagation of the virtual ray of light through the volume, the color and opacity accumulations presented in Equation 2.9 are the most common compositing method used in DVR. An overview of how front-to-back compositing is used to generate a 2D volume rendering is presented in the next section.

2.2 Direct Volume Rendering Techniques

As direct volume rendering aims at the interactive visualization of potentially large volume datasets, efficient implementations of the basic concepts introduced in Section 2.1 have to be used. To leverage the massive parallel computing power of modern *graphics programming units* (GPUs), it is easily possible to divide the rendering process into many independent tasks, where each task computes the contribution of the volume data to a single pixel of the final image. Although there are several approaches to this parallelization, almost all of them rely on a basic progression of steps which have to be carried out repeatedly for each pixel. This progression is known as the *volume rendering pipeline*.

2.2.1 Volume Rendering Pipeline

In this section, the volume rendering pipeline is introduced. Two implementations of this abstract process will be established in the following subsections. A schematic overview of the pipeline is shown in Figure 2.1.

Sampling The basic building block of the volume rendering pipeline is the repeated look-up and processing of samples of the volume dataset at different sampling points. Therefore, the first step in the pipeline is the division of the dataset into a set of sampling points.

Interpolation For each of the samples defined in the previous step, the value of the given volume dataset at that point must be looked up. Since the dataset itself consists of discrete values placed at the intersection points of a regular grid, a sampling point will usually fall between the eight closest neighboring grid values of the dataset. In

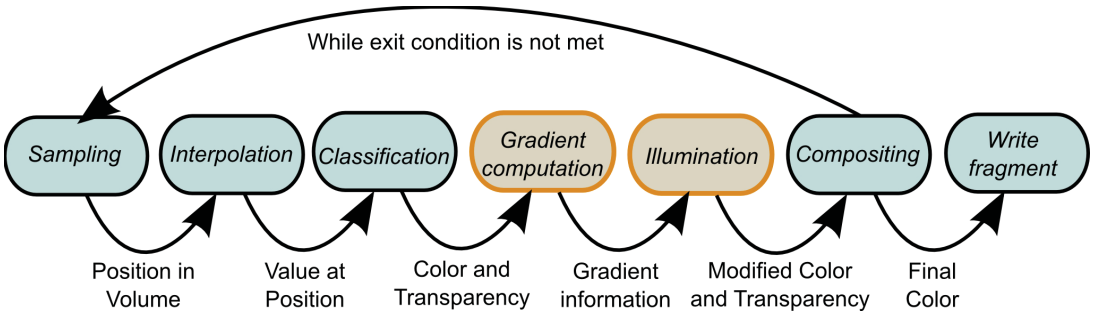


Figure 2.1: The direct volume rendering pipeline. The highlighted steps are optional and increase computation time, but may improve the visual quality and comprehensibility of the rendered image.

order to realize this mapping, these neighbor values must be interpolated to yield a single value, based on their individual distance to the sampling point. Usually, trilinear interpolation is used for this step since it is fast and supported directly by current hardware.

Classification After obtaining the scalar value of the dataset at the current sampling position in the previous step, this value now has to be assigned a color and opacity in order to be used in Equation 2.9. This classification of the sampling point is usually done by using a *transfer-function*, which can be used to distinguish between different materials contained in a dataset. In its most common form, a transfer function directly maps a scalar value interpolated from the dataset to a color and transparency value. There are other, more advanced forms of transfer functions which include additional information such as gradient length (see the following paragraph) or local curvature. However, the techniques presented in this thesis focus on the former, simple case.

Gradient Computation, Shading and Illumination In this optional step, the color values obtained in the previous step can be modified to simulate the presence of external light sources in the scene. A large part of this thesis deals with techniques called *volume illumination models* which aim to simulate the resulting lighting phenomena such as surface shading, shadows or light scattering in different ways. Some of these illumination models require the presence of a normal vector for each voxel. This vector can be approximated by using the direction of largest change of scalar values starting at a given voxel. This direction vector is called the *gradient*, which has to be calculated for each sampling point. A more in-depth overview of available

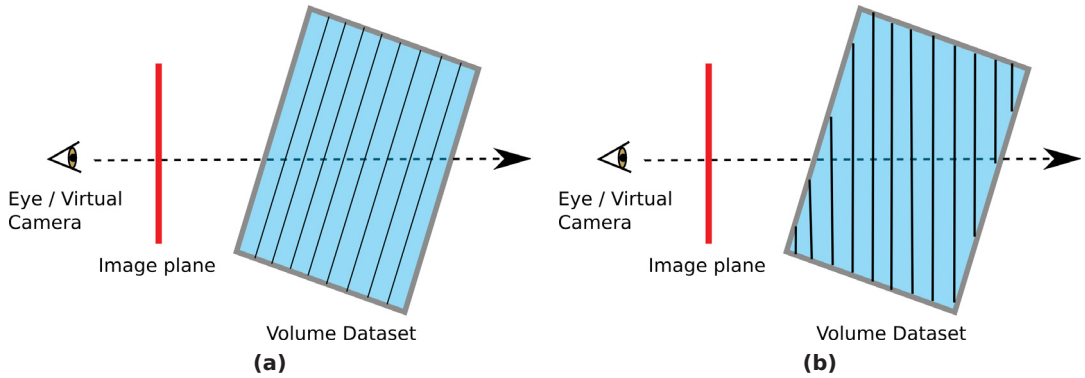


Figure 2.2: The illustration shows two basic approaches to texture slicing. 2D slicing (a) blends axis-aligned slices over each other. The chosen main axis depends on the angle between viewing direction and dataset. A more involved approach is 3D slicing (b), in which the slices are re-oriented to be perpendicular to the viewing direction.

volume illumination models can be found in Chapter 3.

Compositing Finally, all color and transparency values corresponding to a given pixel of the resulting image must be combined to obtain a final color. This can be done iteratively by exploiting a compositing scheme like the one given in Equation 2.9.

2.2.2 Slice-Based Rendering

The first presented paradigm to realize the pipeline introduced in Section 2.2.1 is called *texture slicing*. This family of techniques uses a set of parallel 2D planes to divide the scalar field of a given volume dataset. These so-called *slices* must first be extracted from the original 3D dataset. The task of rendering the dataset is then reduced to a series of blending operations in which all generated slices are blended on top of each other.

Early graphics hardware was not able to directly handle a volume dataset as a 3D texture object. In order to be able to use hardware supported blending at all and thereby generate volume renderings with an acceptable frame rate, Hibbard and Santek [27] suggested a preprocessing in which stacks of object-aligned slices are generated for a volume dataset which can be used later on to perform texture blending operations. This approach called *2D texture slicing* is shown in Figure 2.2(a). Before rendering, for each of the three main axes of a dataset, a texture stack is

generated by extracting slices in regular distances along the given axis from the original dataset. During rendering, one of these object-aligned slice-stacks is selected, depending on the angle of the camera's view vector and the plane normal. If this angle becomes larger than 45°, another stack must be selected in order to hide the gaps between individual slices from the viewer. To generate the final rendering, the transfer function color values from each slice behind a given pixel of the image plane must be blended over each other, which is possible using efficient hardware-based texture blending to achieve compositing. The order of the blending (back-to-front or front-to-back) depends on the location of the virtual camera relative to the dataset. This technique allows for interactive frame rates even on hardware without 3D texture support. However, due to the restriction to object-aligned slices, this technique is prone to rendering artifacts for certain viewing angles between the virtual camera and the dataset.

With the advent of direct 3D texture support of graphics hardware, it was possible to alleviate the drawbacks of strictly axis-aligned slices. In 1994, Cullip and Neumann [12] introduced *3D texture slicing* which takes advantage of modern hardware capabilities. Instead of extracting the slices in a preprocessing, with 3D texture support it is possible to access the 3D scalar field provided by a 3D texture directly in texture space using arbitrarily oriented slices. This enables the orientation of the slices to be always perpendicular to the viewing vector of the virtual camera, thereby preventing the previously mentioned artifacts as well as guaranteeing a constant sampling rate for all camera positions (see Figure 2.2(b)). This approach was later efficiently implemented using modern programmable hardware by Rezk-Salama and Kolb [66] who calculate the slice geometries on-the-fly during rendering. Their implementation is the basis for a number of volume illumination techniques discussed in this thesis.

2.2.3 Ray Casting

Another popular direct volume rendering paradigm that has been made possible by programmable graphics hardware is *ray casting*, which was first introduced by Röttger et al. [73] in 2003. In ray casting, for each pixel of the resulting volume rendering a virtual ray is tracked through the given volume dataset. At first, a geometry surrounding the dataset, the so-called *proxy geometry*, is projected onto the near plane, thereby associating pixels on the image plane with a so-called *entry point* representing the front as well as an *exit point* representing the back of the dataset. The ray direction for each pixel through the dataset for front-to-back compositing can then be computed by subtracting the entry from the exit point vector. By advancing

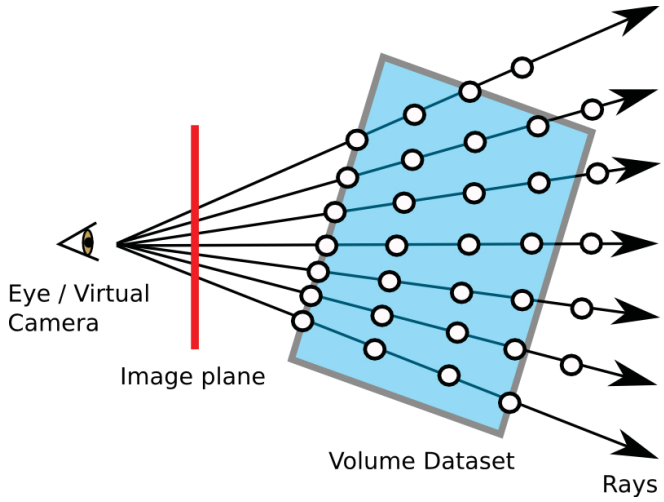


Figure 2.3: The illustration shows the basic principle of ray casting. For each pixel of the image plane, a virtual ray is cast from the observer's position towards the back of the dataset. On its way, the ray regularly samples the dataset to composite a final color for the pixel.

the sampling position step by step, starting at the entry point, the dataset is traversed until the ray exits the bounds provided by the proxy geometry (see Figure 2.3). At each sampling step, classification and optionally illumination is performed and the resulting color value is composited with the previous intermediate result. The final color for each pixel can be computed independently from all other pixels in a *fragment shader*, which is a specialized program running on the graphics hardware. This approach is well suited for the massive parallel computing power provided by modern GPUs and results in highly interactive frame rates even for large rendering sizes. An example for a volume rendering generated using ray casting can be seen in Figure 2.4(a).

The original approach by Röttger et al. has been extended and further improved. Krüger and Westermann [39] introduced a few fundamental optimizations in 2003 to limit the length of each virtual ray to only those regions of the dataset which actually contribute to the final pixel color. According to Equation 2.9, if the accumulated opacity α_{dst} reaches a value close to 1, new incoming colors will hardly have any influence on the final color. Therefore, Krüger and Westermann suggest *early ray termination* for opacity values beyond a given threshold value to avoid unnecessary ray traversal. Similarly, using the bounding box of the volume dataset as proxy geometry is straightforward (see Figure 2.4(b) and (c)), but will often include regions

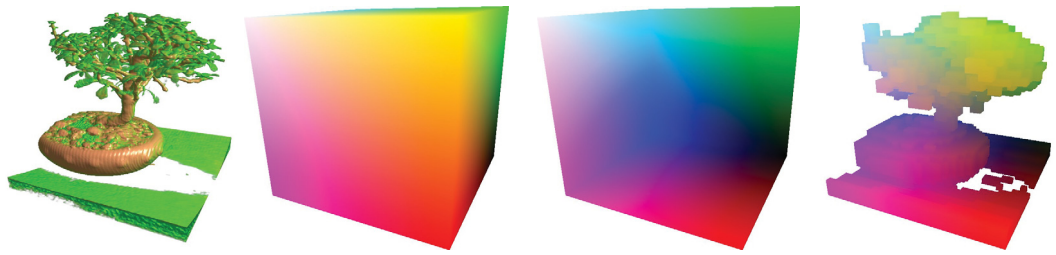


Figure 2.4: A volume dataset containing a bonsai rendered using ray casting (a). Images with color-coded points on the dataset's bounding box can be used to look up entry (b) and exit (c) points for individual rays. Optionally, optimized proxy geometries can be used to skip empty regions surrounding the dataset (d).

that are not of interest to the viewer. To avoid tracking the ray through these regions that will not contribute to the result due to a completely transparent associated transfer-function value, Krüger and Westermann introduced *empty space skipping*. This optimization can be realized for empty regions surrounding the dataset by using a proxy geometry which discards transparent space in front of and behind the volume (see Figure 2.4(d)). This geometry must be recomputed for every change of the transfer function. However, the speed up during rendering due to shortened ray lengths is significant.

Although some of the illumination techniques mentioned in this thesis significantly extend and alter the basic ray casting approach, these optimizations are almost always applicable and help to achieve interactive frame rates.

Chapter 3

Illumination Models for Direct Volume Rendering

In this chapter we introduce some differentiating features of volume illumination models. Furthermore, an overview of existing established volume rendering techniques that have been presented over the years is given.

Illumination models for DVR aim to extend the volume emission-absorption rendering integral introduced in Section 2.1.1 by additional terms that account for light arriving at the dataset from external sources in a scene. Instead of simply modeling how a point within the dataset affects non-directional background light, the introduction of light sources implicates the simulation of lighting phenomena such as reflection, refraction and scattering. The degree to which these phenomena are incorporated into a given illumination model varies greatly. This chapter will first introduce some basic differences between existing techniques before describing an exemplary subset of established volume illumination models in detail.

3.1 Classification of Volume Illumination Models

In this section, an overview of different illumination model attributes is given which can be used to categorize techniques, ranging from basic and light-weight to advanced and computationally intensive models.

Modeling of Scattering

One of the most fundamental attributes of a volume illumination model is the way light scattering is simulated [53]. Light scattering in the context of volume rendering generally describes how rays of light are deflected as they travel through the given participating medium. In reality, scattering in this situation is a complex process in

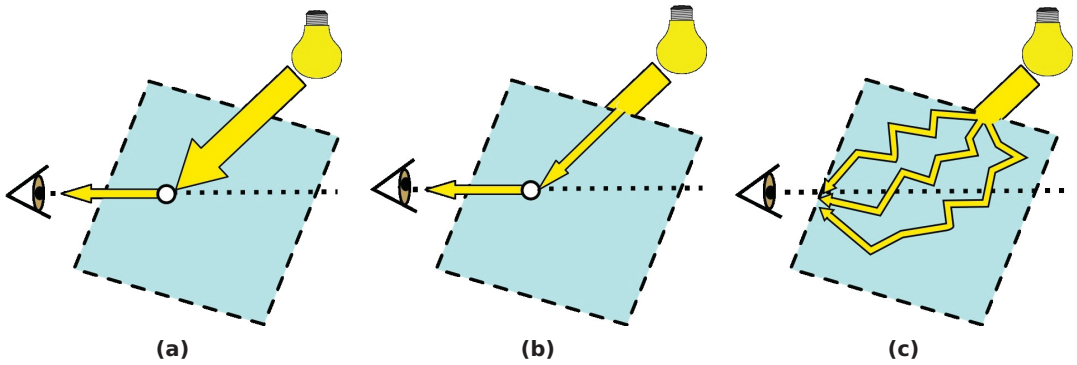


Figure 3.1: Scattering is one aspect in which volume illumination models differ. Simple models only consider single scattering (a). More advanced techniques capable of including shadows additionally model attenuation of the incoming light by the surrounding medium (b). The most realistic models include multiple scattering of light on its way through the medium (c).

which each ray is partially reflected and refracted many times before exiting the medium. A complete simulation of this process would require too many calculations for each ray of light to allow for interactive rendering. Therefore, most illumination models make some simplifying assumptions which essentially allow only a subset of possible deflections for each ray as it enters the medium from an external light source. For this section, we will reformulate the volume rendering integral given in Equation 2.4 to refer to a location $\vec{x}$ in a volume rendering scene and to introduce an additional parameter $\vec{\omega}_o$ representing the viewing vector of the virtual camera which is needed to calculate reflection effects:

$$I(\vec{x}, \vec{\omega}_o) = I_0 T(\vec{x}_f, \vec{x}) + \int_{\vec{x}_f}^{\vec{x}} q(\vec{x}', \vec{\omega}_o) T(\vec{x}, \vec{x}') \mathrm{d}\vec{x}'. \tag{3.1}$$

In equation 3.1, $\vec{x}_f$ denotes the scene exit point of the ray that is tracked from the virtual camera. Furthermore, a modified version of the accumulated transparency function $T(s_1, s_2)$ defined in Equation 2.3 is used, which now receives two point vectors $\vec{x}_1$ and $\vec{x}_2$ within the medium as arguments instead of scalar values and returns the remaining intensity of light after it has traversed the linear distance between these two points. Volume illumination models modify the emission term $q(\vec{x}', \vec{\omega}_o)$ to simulate light reflection, to include an attenuation of the transfer function color due to occlusion or to add an additional scattering color.

Generally, illumination models can be divided into two groups: Those modeling

local illumination and those modeling *global illumination*. Local illumination models assume that light arrives at a given point in the medium without any previous interaction with the medium, and that the light, once it has interacted with the medium at that point, is directly reflected to the eye of the viewer without any further deflection. This is achieved by using

$$q(\vec{x}', \vec{\omega}_o) = L_e(\vec{x}', \omega_o) + L_i(\vec{x}') \cdot R(\vec{x}', \vec{\omega}_o), \tag{3.2}$$

where $L_e(\vec{x}', \omega_o)$ is the radiance emission term due to the transfer function and $R(\vec{x}', \vec{\omega}_o)$ is a simple function that computes how much of the incoming light $L_i(\vec{x}')$ is reflected in direction $\vec{\omega}_o$. In this simple approach, we assume that light arrives unimpeded at $\vec{x}'$ from a number of discrete directions. This approximation is known as *single scattering* (see Figure 3.1(a)). Local illumination models using this approach can usually be evaluated directly during traversal of the DVR pipeline, as the influence of the region surrounding the current sampling point is not accounted for. In addition to the location of the sampling point, these models typically only require the locations of the point light sources and the virtual camera. Although they provide only a coarse reproduction of reality, local illumination models already enhance the spatial perception of volume rendered images significantly.

An extension of single scattering is *single scattering with attenuation*. Illumination models belonging to this category model occlusion of external light sources at a given sampling point due to the surrounding medium (see Figure 3.1(b)). This is accomplished by introducing a more involved computation of the light distribution around $\vec{x}'$ in the volume rendering integral rather than assuming discrete point light sources as in 3.2. We introduce an additional parameter $\vec{\omega}_i$ for the reflection function and the incoming light function to include different radiance intensities depending on the direction. Now, an integral over the sphere around $\vec{x}'$ has to be solved:

$$q(\vec{x}', \vec{\omega}_o) = L_e(\vec{x}', \omega_o) + \int_{4\pi} R(\vec{x}', \vec{\omega}_o, \vec{\omega}_i) L_i(\vec{x}', \vec{\omega}_i) T(\vec{x}', \vec{x}'_{f\vec{\omega}_i}) \mathrm{d}\vec{\omega}_i. \tag{3.3}$$

For every direction $\vec{\omega}_i$ around $\vec{x}'$, a ray extending from $\vec{x}'$ towards the corresponding scene exit point of the ray $\vec{x}'_{f\vec{\omega}_i}$ accumulates the decay of transparency between $\vec{x}'$ and $\vec{x}'_{f\vec{\omega}_i}$ due to the medium using the function $T(\vec{x}_1, \vec{x}_2)$. This term is weighted by the incoming radiance $L_i(\vec{x}', \vec{\omega}_i)$ from that direction, as well as the reflection function $R(\vec{x}', \vec{\omega}_o, \vec{\omega}_i)$ which calculates how much of the incoming radiance is reflected towards the viewer. This leads to a possible attenuation of the locally reflected radiance, as some features in the scene will cast shadows onto the point at $\vec{x}'$ since they are located between the point and incoming light.

Techniques employing Equation 3.3 are considered a simple form of global illumination model, since the whole scene can potentially influence the lighting at the current sampling point. However, rays of light originating from a light source are still considered not to change direction before or after the sampling point in this model, but rather only their intensity. Although some of the techniques introduced later in this chapter support more than single scattering with attenuation, we will use this scattering model to compare and evaluate volume illumination models in Chapter 6.

Some techniques allow an even more involved simulation of lighting phenomena by allowing the light to be reflected and refracted many times on its way through the medium, which is closest to the behavior of light in reality (see Figure 3.1(c)). This form of scattering is therefore called *multiple scattering*. To correctly model this behavior, the reflection term must now consider the result of the volume rendering equation from all directions in the surrounding medium to correctly compute the amount of incoming light which can be reflected. This leads to the following equation for the emissive term $q(\vec{x}', \vec{\omega}_o)$:

$$q(\vec{x}', \vec{\omega}_o) = L_e(\vec{x}', \omega_o) + \int_{4\pi} R(\vec{x}', \vec{\omega}_o, \vec{\omega}_i) \int_{\vec{x}'}^{\vec{x}'_{f\vec{\omega}_i}} I(\vec{x}'', -\vec{\omega}_i) \mathrm{d}x'' \mathrm{d}\vec{\omega}_i. \tag{3.4}$$

In this equation, $\vec{x}'_{f\vec{\omega}_i}$ is again the scene exit point in direction ω_i. Since $I(\vec{x}, \vec{\omega}_o)$ is used, the inner integral relies on the radiance transport calculation of the surrounding medium and is therefore recursive. Because Equation 3.4 can only be solved incrementally and is computationally very expensive to evaluate correctly, this level of realism is usually not supported in applications aimed at interactive use. For some phenomena that are due to multiple scattering, statistical approximations exist that evaluate light transport in a local neighborhood around a given location in the scene while being subject to some restrictions that simplify the situation. *Subsurface scattering* is one of these special cases which describes how light interacts with semi-translucent materials like wax or skin. After light has first entered the medium at one location, it is scattered back towards the surface and exits the medium within a certain radius around the point of entry, depending on the material. A technique to simulate subsurface scattering in DVR is introduced in Chapter 5.

Dependency on Direct Volume Rendering Paradigm

While many volume illumination techniques, e.g., [42, 74, 71], can be used independently of the implementation of the DVR pipeline, some approaches, like [37, 78, 89], are designed to work in conjunction with 3D texture slicing. Using ray casting in DVR has the benefit of an independent parallel process being executed for each

pixel, which allows optimizations such as early ray termination. However, a ray of light modeled by such a process usually cannot influence neighboring processes and rays during the generation of the image, since rays are not synchronized and have no means of communication on typical graphics hardware. In 3D texture slicing, however, a whole slice of the volume is completed before continuing with the next slice one step further away from the viewer, thereby essentially advancing each pixel-ray one step at the same time. When the next slice is rendered, the previously accumulated result can be made available in the current rendering pass to access lighting information from neighboring parts in the medium that were traversed earlier. The techniques introduced in Section 3.2 depend on this approach to simulate global lighting phenomena.

Light Source Constraints

Volume illumination techniques differ in the number and configuration of virtual light sources that they can simulate. Most techniques, like [42, 74, 37], model the behavior of a point light source located at one specific point in space, since this kind of light source is easy to describe mathematically. Due to their simplicity, local illumination models like the Phong model [63] support multiple point light sources, while most advanced models which include attenuation of light intensity limit themselves to a single point light source. Some advanced models like directional occlusion slicing ([78], see Section 3.2.2) further constrain the possible location of the light source, e.g., to a hemisphere around the viewpoint of the observer. Finally, some models like spherical harmonic lighting ([71], see Section 3.3.3) do not support point lights but area light sources, which allow the simulation of natural lighting phenomena like sunlight.

Requirement of Data Preprocessing

Some volume illumination models for DVR, e.g., [71, 75], require a preprocessing in which a given dataset is analyzed and an auxiliary structure is generated before actual rendering is performed. These models differ in the complexity of their preprocessing and therefore in the time necessary for the computation. Usually, a separate volume texture with equal or lower size is used to store precomputed lighting information, which is filled with values during the preprocessing. Some volume illumination models, like [75], require the preprocessing only once for a dataset. For other models, like [71, 74], the preprocessing has to be repeated as soon as the transfer function and subsequently the transparency within the dataset change, which may interfere with an interactive transfer function design for large datasets.

Locality Restrictions

In principle, any region within a volume rendering scene can influence the illumination of a point in the scene by attenuating the intensity of light passing through the region on its way to that point. However, some volume illumination models such as dynamic ambient occlusion ([75], see Section 3.3.4) weaken this requirement and consider only the local neighborhood of a voxel as a possible source for attenuation and scattering, which is based on the assumption that regions distant from the current point will only have little influence on its illumination. Therefore, illumination models with a local neighborhood restriction trade physical correctness of the light transport for a reduced complexity of the preprocessing and possibly faster evaluation of the model during rendering.

Frequency of Simulated Phenomena

Most volume illumination models that simulate single scattering with attenuation produce shadows that are cast across the volume rendering scene. These shadows differ in their quality between models. For instance, Spherical Harmonic Lighting [71] is based on the convergence of an infinite series of weighted basis functions. This convergence is rather slow, and sharp peaks in the domain of the original function will not be visible in the band-limited reproduction. This results in very soft shadowing effects. Other models like half angle slicing ([37], see Section 3.2.1) produce very sharp shadows across the entire scene, even if the shadow is cast by small high-opacity regions. This property is commonly referred to as the frequency of simulated illumination phenomena. Its influence on the user's perception will be investigated in Chapter 6.

Requirement of Surface Normals

Finally, some illumination models, like Phong lighting ([42], see Section 3.3.1), require the presence of surface normals to compute how exactly light is reflected at a given point in the medium. However, surface normals are not naturally given in DVR. As mentioned in Section 2.2.1, the DVR pipeline can be extended by an optional step which computes the intensity gradient at the current sampling point, which can be used as a substitute for a surface normal. Depending on the required quality of the gradient, this step may introduce significant computation costs. Furthermore, a meaningful gradient may not always exist in regions of a dataset with highly varying values. Some illumination models, like [75, 74], therefore allow the optional use of gradients so that the model may be adapted to the currently visualized data.

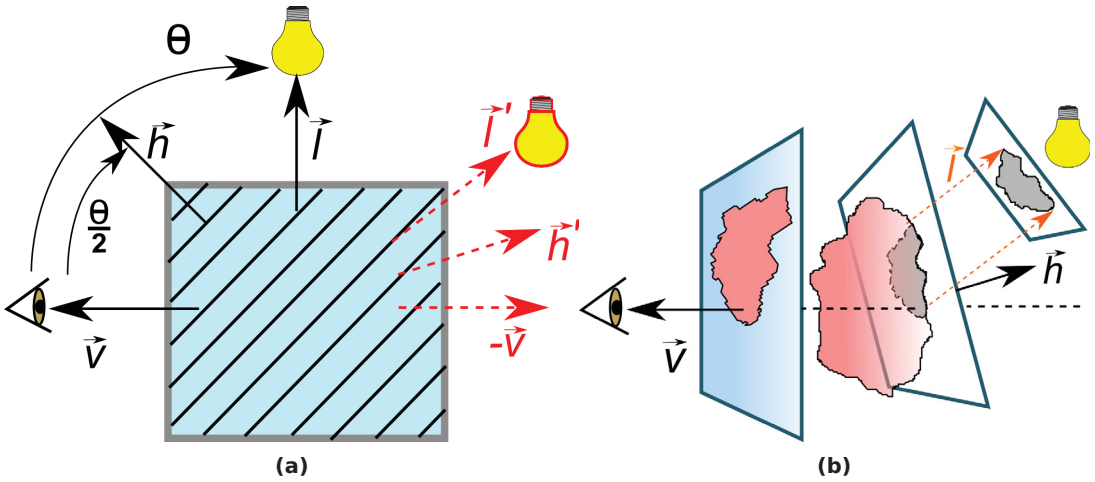


Figure 3.2: For half angle slicing, the normal of the slice geometries is chosen as the vector $\vec{h}$ which is halfway between the viewing vector $\vec{v}$ and the light vector $\vec{l}$. If the angle θ between $\vec{v}$ and $\vec{l}$ becomes greater than 90°, $-\vec{v}$ is considered instead, as indicated by the vectors highlighted in red (a). During rendering, an additional rendering pass using the light source as viewpoint is performed to accumulate illumination information (b).

3.2 Slice-Based Techniques

In this section, some models that build on the slicing rendering paradigm are introduced. All of them use a second slice buffer and a two-pass rendering algorithm to transfer temporary illumination information between the current and the next slice. One advantage of this approach is that no preprocessing step is necessary, as the two-buffer scheme allows incremental updates of the accumulated illumination while the volume is rendered. Additionally, none of the techniques in this section depend on intensity gradients.

3.2.1 Half Angle Slicing

Half angle slicing was introduced by Kniss et al. [37]. The name of this technique is derived from its use of the *halfway vector* $\vec{h}$, which is the normalized vector that lies halfway between the normalized viewing vector $\vec{v}$ pointing from the center of the dataset towards the virtual camera and the normalized vector $\vec{l}$ pointing from the

center of the dataset towards the light source (see Figure 3.2):

$$\vec{h} = \frac{\vec{v} + \vec{l}}{|\vec{v} + \vec{l}|} \tag{3.5}$$

In half angle slicing, $\vec{h}$ is used as the normal of the slice geometries, contrary to the usual approach of slices that are perpendicular to the viewing vector. If the angle θ between $\vec{v}$ and $\vec{l}$ is larger than 90°, $\vec{v}$ is replaced in Equation 3.5 by $-\vec{v}$ to ensure that the angle between the slice normal and the viewing vector as well as the angle between the slice normal and the light vector is at most 45°. This has the benefit that each slice is well visible both from the point of view of the observer as well as from the point of view of the light source. If $-\vec{v}$ was considered, the direction in which the slice geometries are traversed is reversed, so that the rendering process always starts closest to the light source.

During rendering, two passes are performed for each slice, first one from the viewpoint of the observer and then one from that of the light source. The result of the second rendering pass is saved in a separate light buffer, in which the opacity calculated in the previous lighting pass is updated for each pixel on the slice. When the first pass for the following slice is executed, the current light buffer is bound as a texture. The location within the light buffer, which corresponds to the current sampling point, is calculated and used to look up the accumulated occlusion from previous slices to attenuate the voxel color.

This process results in hard shadows in the final image which are cast over the entire scene. A modified version of this scheme is also used by Kniss et al. to simulate multiple scattering. Due to the dependency on the slicing direction, half angle slicing only supports a single light source, which can be placed in an arbitrary position around the dataset. Furthermore, the technique may produce artifacts under certain conditions due to the altered slice orientation. If the angle between $\vec{h}$ and $\vec{l}$ approaches its maximum of 45°, the viewer may be able to see the gaps between slice geometries if the slice distance is chosen too large.

3.2.2 Directional and Multidirectional Occlusion Slicing

Directional occlusion shading was introduced by Schott et al. [78]. As another slice-based technique, it takes advantage of the front-to-back rendering of the view aligned textures. This technique does not allow a free placement of the light source, but instead assumes that the light vector $\vec{l}$ is always identical to the viewing vector $\vec{v}$ (see Figure 3.3(a)). This simplification has the advantage that the slicing direction does

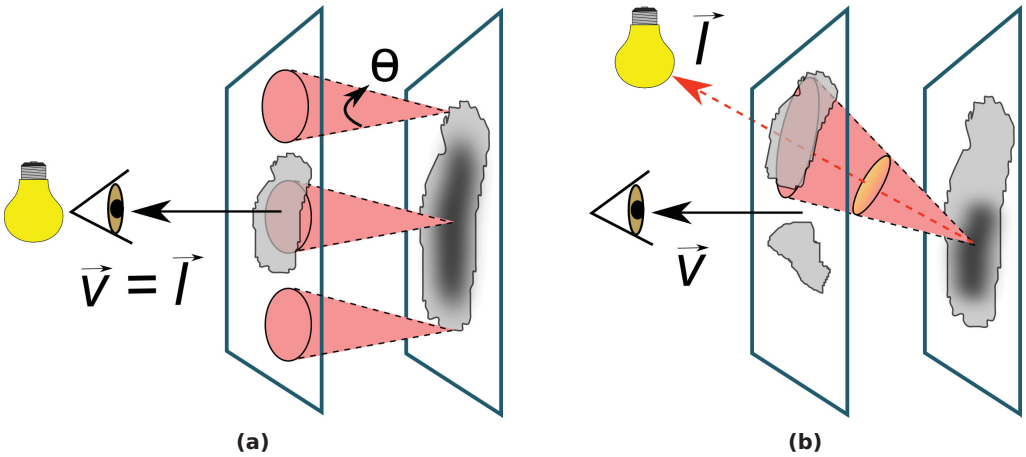


Figure 3.3: In directional occlusion shading, occlusion information is blurred along the negative viewing vector $\vec{v}$ which coincides with the light vector $\vec{l}$ by sampling an occlusion buffer obtained from the previous slice within a cone with opening angle θ (a). To allow a somewhat independent placement of the light source, multidirectional occlusion shading employs a potentially tilted and shifted ellipse as the occlusion sampling area instead of a simple radial blur (b).

not have to be adapted like in half angle slicing.

To propagate occlusion information from one slice to the next, a separate occlusion buffer is employed. In a second rendering pass for each slice, the accumulated occlusion from the previous slice is sampled within the base of a cone originating from the current sample which opens towards the camera. The number of occlusion samples as well as the opening angle θ of the cone can be controlled by the user. The occlusion buffer is updated with the average opacity obtained from the cone samples, which essentially blurs the opacity of surrounding pixels on the previous slice onto the current sample. In the rendering pass of the next slice, the updated occlusion buffer can be used to look up the current occlusion value of the current sample. This occlusion value can then be used to attenuate the transfer function color. The resulting shadows are soft and are located halo-like around high opacity features due to the fixed position of the light source at the virtual camera.

Multidirectional occlusion shading, introduced by Šoltészová et al. [89], extends the directional occlusion technique to allow for a more flexible light source placement. This is achieved by replacing the radial blurring used in directional occlusion shading by a freely orientable, shifted cone. This is achieved by employing an auxiliary 2D texture containing an ellipse that is produced by projecting the tilted cone onto

the plane defined by the slice orientation (see Figure 3.3(b)). The texture is then convoluted with the previous occlusion buffer during the second rendering pass to update the occlusion information. This approach allows the user to control direction and distance of the light source, resulting in soft directional shadows. However, the angle between $\vec{l}$ and $\vec{v}$ cannot be larger than 90°using this technique, essentially limiting the light source position to a hemisphere centered around the viewer.

3.3 General Techniques

In this section, some techniques that can be applied to both 3D texture slicing or ray casting are introduced. The advanced models in this section all use a precomputed auxiliary volume dataset containing illumination information, which could be accessed during rendering using either one of these DVR paradigms. However, all implementations of the techniques in this section that were used as part of this thesis employ ray casting.

3.3.1 Phong Lighting

Phong lighting [63] is an established technique for local surface illumination. It supports the simulation of single scattering from multiple point light sources towards the viewer by adding three different terms which represent one type of light reflection each: diffuse reflection, specular reflection and ambient reflection. Each term is weighted according to the material that is supposed to be simulated at the current point by the material coefficients k_d, k_s and k_a, as well as by light coefficients I_d, I_s and I_a which specify the proportion of light reflection in the scene due to each term. Originally, the technique was introduced for the illumination of polygonal surfaces. It therefore uses the surface normal $\vec{n}$ at the illuminated point in the scene to calculate the diffuse term which depends on the angle between the normal and the normalized vector $\vec{l}$ pointing towards the light source. This term describes the general ability of the surface material to reflect light diffusely, independent of the location of the viewer. Furthermore, the normal is used to calculate the normalized vector of perfect light reflection $\vec{r}$, depending on $\vec{l}$. The angle between $\vec{r}$ and the normalized vector $\vec{v}$ pointing towards the viewer is then used to calculate the amount of specular reflection at the current point in the scene. The specular reflection term describes the material's ability to directly reflect light, similar to a mirror. An additional material shininess parameter α is used to control the size of the area of high specular reflection. Finally, all indirect light reflections are simply summarized in the constant ambient term which is independent of the locations of the viewer or the light source. Given

the mentioned variables at a point $\vec{x}$ in the scene, the Phong lighting model can be summarized in the following equation for a single light source:

$$I(\vec{x}) = I_a \cdot k_{a_{\vec{x}}} + \max(0, \vec{l}_{\vec{x}} \cdot \vec{n}_{\vec{x}}) \cdot I_d \cdot k_{d_{\vec{x}}} + (\max(0, \vec{r}_{\vec{x}} \cdot \vec{v}_{\vec{x}}))^{\alpha} \cdot I_s \cdot k_{s_{\vec{x}}} \tag{3.6}$$

For multiple light sources, the diffuse and specular terms from other light sources are subsequently summed up to yield a final light intensity value.

One notable variation of this lighting approach is the *Blinn-Phong* model [4]. It replaces the reflection vector $\vec{r}$ in Equation 3.6 with the halfway vector $\vec{h}$ which has already been introduced for half angle slicing in Section 3.2.1. This has a computational advantage as the halfway vector is constant for the entire rendering process if the light source is located at an infinite distance from the scene, as $\vec{l}$ and $\vec{v}$ will not depend on $\vec{x}$ in this case. Furthermore, the resulting images have been experimentally verified to better approximate reality compared to the use of the reflection vector [57].

The Blinn-Phong lighting model was first applied to volume rendering by Levoy [42], who used voxel gradients as surface normals to calculate local lighting effects. The model has since become an established part of the DVR pipeline. As long as the dataset provides meaningful intensity gradients, the Blinn-Phong model can also be combined with advanced techniques by first shading the transfer function color with this model before further attenuating the shaded color with scattering or shadow effects due to the advanced model.

3.3.2 Shadow Volume Propagation

Shadow volume propagation was introduced by Ropinski et al [74]. It is a technique that approximates the propagation of radiance along the light direction within the dataset using a preprocessing step in order to subsequently allow efficient rendering. During the preprocessing, the current position of the light source relative to the dataset is used to select that side of the dataset bounding box which would receive the largest amount of incoming radiance. Starting with the slice S_0 that is adjacent to the selected bounding box face F_0, light information is then propagated slice by slice into an auxiliary dataset along the corresponding major axis of the bounding box (see Fig. 3.4). Each voxel on the current slice is influenced by certain close voxels on the previous slice. This is realized by utilizing a virtual cone originating from each voxel which has a user defined opening angle θ and opens towards the light source. Illumination information for all voxels on the previous slice within the cone will influence the current voxel, either by blocking light due to a high opacity or by passing on color as determined by the current transfer function, thereby simulating scattering. The auxiliary illumination volume has to be updated as soon as either the

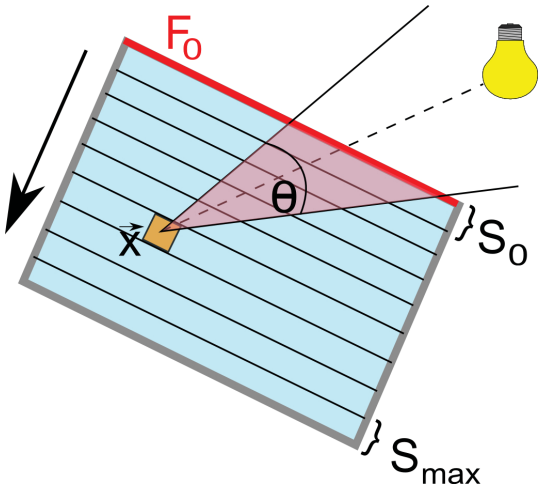


Figure 3.4: The figure illustrates the principle of the shadow volume propagation preprocessing step. Starting from the slice S_0 closest to the bounding box face F_0 which receives the most radiance from the light source, illumination information is propagated through an auxiliary volume dataset along the major axis indicated by the arrow. A cone with opening angle θ is used to determine which voxels from the previous slices contribute to the illumination of the current voxel $\vec{x}$.

light source position or the transfer function change. However, this preprocessing can be implemented on modern GPUs quite efficiently, so that an interactive change of the parameters is possible.

During the DVR process, the auxiliary light volume is accessed to look up the remaining luminance value at the current voxel and to attenuate the transfer function color, producing high frequency shadows. Optionally, the cone-based scattering color can be added to the current color to include a scattering effect. The technique supports a single point light source which can be placed arbitrarily around the dataset. The use of intensity gradients is optional and can be employed for additional specular highlights emphasizing surface-like regions within the dataset.

3.3.3 Spherical Harmonic Lighting

Spherical Harmonics (SHs) are a mathematical tool to efficiently describe functions defined on a sphere surface. Their use as part of an illumination technique in computer graphics was originally introduced by Sloan et al. [82].

Spherical Harmonics

Providing a detailed derivation of the concepts behind SHs would be beyond the scope of this chapter. Instead, we discuss the main properties exploited by the material techniques presented in Chapter 5 as well as by the lighting technique that is part of the study presented in Chapter 6, and refer the reader for a more detailed explanation to Green [18].

SH basis functions allow the representation of a continuous function given on the surface of a unit sphere, where each point can be uniquely identified by its polar angle ϑ and its azimuth angle φ. Similar to a Fourier series, an infinite number of SH basis functions $Y_l^m(\vartheta, \varphi)$ of varying order l and degree m can be weighted with SH coefficients c_{ij} to represent arbitrarily complex functions $f(\vartheta, \varphi)$ defined over the sphere surface:

$$f(\vartheta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_{lm} Y_l^m(\vartheta, \varphi).$$

Intuitively, the functions $Y_l^m(\vartheta, \varphi)$ can be thought of as having two independent components to span the sphere, represented by the two indexes m and l. Complex exponential functions with a single argument of period 2π are used to define the variations along the latitude, while associated Legendre polynomials are used to capture variations along the longitudes. To allow an efficient computation and representation of f, the representation can be bandlimited:

$$\tilde{f}(\vartheta, \varphi) = \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{lm} Y_l^m(\vartheta, \varphi)$$

In this bandlimited version, the number of summands is equal to $1 + 3 + ... + (2L + 1) = (L + 1)^2$. Thus, the chosen number of bands L directly determines the number of resulting coefficients c_{ij}. Using more coefficients will result in a better representation of the original function, but also increase the amount of memory per projected function.

To project f and compute the c_{ij}, a least square projection is used as an approach to minimize the squared error E between the approximation and the original function f:

$$E = \int_{4\pi} \left(\sum_{l=0}^{L} \sum_{m=-l}^{l} c_{lm} Y_l^m(\vartheta_s, \varphi_s) - f(\vartheta_s, \varphi_s) \right)^2 ds\,, \tag{3.7}$$

From this approach, it follows by searching for a minimum of E that the SH coefficients c_{lm} can be computed as an integral of a product of the corresponding basis

function and f over the surface S of the unit sphere:

$$c_{lm} = \int_{4\pi} Y_l^m(\vartheta_s, \varphi_s) f(\vartheta_s, \varphi_s)\, ds \tag{3.8}$$

This integral is usually solved in a discrete manner for the required number of coefficients using a Monte Carlo integration with random samples distributed uniformly over the unit sphere, for which the product of the two functions is evaluated. Due to the orthonormality of the SH basis functions, it can be shown that the integration of the product of two SH-projected functions $\tilde{f}$ and $\tilde{g}$ over the sphere surface is equal to the sum of the products of all their coefficients c_i and d_i:

$$\int_{4\pi} \tilde{f} \cdot \tilde{g}\, \mathrm{d}s = \sum_i^n c_i d_i \tag{3.9}$$

Terms like the one on the left of Equation 3.9 occur often in illumination calculations. The presented property of SHs makes solving these integrations very efficient, since when interpreting c and d as vectors, the integration can be easily computed on the GPU by using a dot product. Thus, instead of reconstructing and integrating the original functions, the calculation of a dot product is sufficient.

Another important SH property is rotational invariance. When a rotation is desired, it can be directly applied to the SH coefficients. This is very helpful when using SH representations for lighting, where the light source may rotate around an object. Due to its practicability and the thus easy realization on the GPU, the technique presented by Křivánek et al. [40] is used in this thesis, which splits up the coefficient rotation into Euler angles.

Usage of Spherical Harmonics in Volume Illumination

Spherical harmonic lighting was originally used in volume rendering by Beason et al. [2] for isosurfaces. Ritschel suggested the use of spherical harmonics for DVR with ray casting [71]. In a preprocessing, occlusion information per voxel is projected into spherical harmonic space. From each voxel, rays are traced towards a random set of sample directions that are distributed uniformly over the sphere. For each ray, the occlusion of light caused by the medium between the current voxel and the intersection of the ray with the border of the dataset is calculated, depending on the transparency of the chosen transfer function. This process results for each voxel in a spherical function which determines visibility values between 0 and 1 for each direction. This function is projected into SH space and saved as a set of coefficients in auxiliary datasets.

During rendering, the spherical integral over the product of a SH-projected area light source and the projected voxel occlusion is efficiently computed using the precomputed visibility information, thereby matching the amount of light arriving from each direction with the potential of a voxel to receive light from that direction. The result is then used to attenuate the local voxel color, resulting in very soft shadows. For practical reasons we use at most $L = 3$ bands throughout this thesis, resulting in $(3+1)^2 = 16$ coefficients per projected function. This set of coefficients can be conveniently saved for each voxel in four additional 3D RGBA textures.

3.3.4 Dynamic Ambient Occlusion

The term *ambient occlusion* is used for a group of illumination techniques which focus on how accessible a point in a scene is by outer influences such as light. The original approach was presented by Zhukov et al. [98] for polygonal graphics. In general, a hemisphere around the surface normal $\vec{n}$ of a given point $\vec{x}$ is searched in all directions $\vec{\omega}$ for occluding features in order to calculate an overall visibility value $V(\vec{x})$ for that point:

$$V(\vec{x}) = \frac{1}{2\pi} \int_{2\pi} (\vec{n} \cdot \vec{\omega}) v_{\vec{x}}(\vec{\omega}) \mathrm{d}\vec{\omega} \tag{3.10}$$

In this equation, $v_{\vec{x}}(\vec{\omega})$ denotes the visibility function which defines how much light is visible at $\vec{x}$ when looking in the direction $\vec{\omega}$. The resulting value $V(\vec{x})$ is independent of a light source position. It simply expresses how much darker the given point would appear in the scene due to the fact that surrounding features in the scene block incoming light. The integral in Equation 3.10 can be precomputed in a straightforward manner for static scenes using Monte Carlo integration and casting occlusion rays towards randomized sample directions. The lighting effects produced by ambient occlusion are rather subtle and limited in range, but add an additional layer of realism to the scene.

Dynamic ambient occlusion has been proposed by Ropinski et al. [75]. It focuses on the adaptation of the ambient occlusion value of a voxel in a volume dataset during interactive changes of the transfer function. To avoid costly re-computations during transfer function changes, an expensive preprocessing, which is necessary only once per dataset, is executed. During this preprocessing, a histogram of voxel values in a neighborhood within a given distance of each voxel is constructed. To save memory, the resulting set of histograms is condensed into a number of histogram clusters which are saved in a 2D texture. An auxiliary volume dataset is used to save the corresponding surrounding histogram ID for each voxel. During rendering, the associated cluster histogram serves as an approximation of the dataset configuration

around the current voxel. The histogram can be accessed together with the current transfer function to calculate an approximate value of the current neighborhood occlusion $V_{\vec{x}}$ of each voxel due to the transfer function opacity values. The resulting value is then used to attenuate the color obtained from the DVR classification step. The local histogram can also be used to simulate additional lighting phenomena, such as color bleeding and scattering. However, since nothing but the transfer function values within a limited radius around each voxel are considered, dynamic ambient occlusion can only yield a rough approximation of natural global illumination.

3.4 Techniques Based on Screen Space

In this section, a few techniques are introduced that employ a completely different approach to illumination than those that were previously mentioned. Instead of incorporating illumination information into the image during rendering, techniques that are based on *screen space* add illumination to an image *after* the rendering process is finished. Techniques in this category use the *depth buffer* to estimate global illumination effects. This buffer contains the depth information of objects in a scene, i.e., the distance between the first object that is projected onto a pixel and the virtual camera. By considering neighboring depth values, an ambient occlusion term for the current pixel can be estimated. Therefore, the effect produced by these techniques is often referred to as *screen space ambient occlusion* (see Figure 3.5). Illumination techniques based on screen space are relatively inexpensive to apply compared to techniques in other categories. They do not rely on the complexity of the scene, and do not require any preprocessing since the depth buffer is usually a byproduct of the regular rendering process.

However, these techniques are by their nature strongly dependent on the position of the virtual camera and may produce non-continuous illumination effects if the depth buffer changes significantly due to camera movement. Furthermore, since the depth buffer contains only the single front-most value for each pixel, illumination effects due to transparency cannot be reproduced. This makes screen-based techniques applicable to volume rendering only for DVR scenes which feature highly opaque transfer functions. In general, since the depth buffer can only supply limited information about a scene, these techniques are but a coarse approximation of true global illumination.

Luft et al. [50] introduced *depth darkening* which uses an unsharp masking of the depth buffer to generate a low-pass filtered copy which essentially blurs the local depth value with neighboring depth values. The content of the original depth buffer and the content of the filtered copy are then compared to obtain a difference value, which is used to augment the pixel color with a depth-darkening effect. Mittring [56]

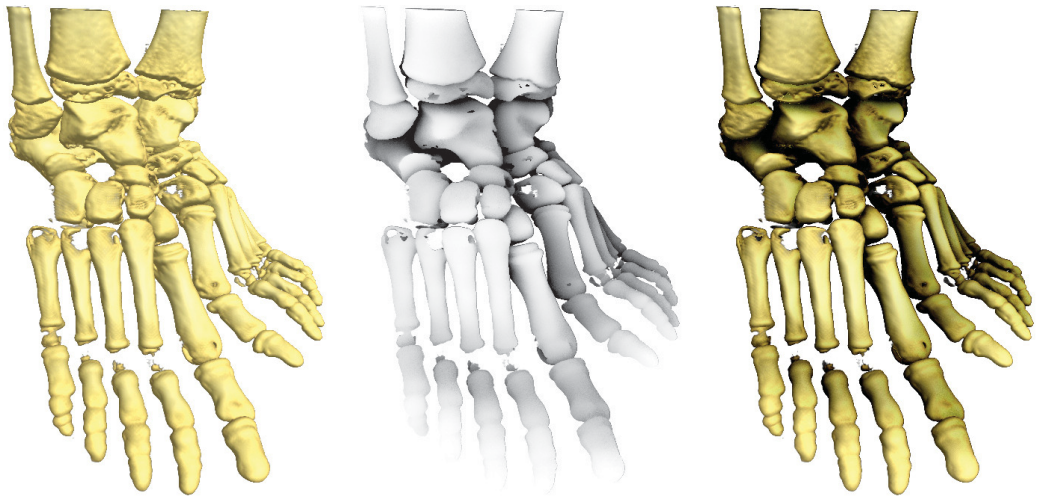


Figure 3.5: Screen space ambient occlusion. On the left, we see a ray casting result obtained from a CT scan of human feet, shaded with the Phong model (a). By analyzing the depth buffer, an attenuation factor is calculated (b) and used to weight the previous image. This results in a modified image with an added ambient occlusion effect (c).

and Shanmugam and Arikan [80] analyzed the depth buffer by randomly accessing a number of values in the depth buffer in a region around the current pixel. The obtained depth values are used to approximate the integral in Equation 3.10. Their approaches have been extended by Ritschel et al. [72] to include directional occlusion as well as diffuse indirect bounces of light, introducing a color bleeding effect.

Diaz et al. introduced the concept of *summed area tables* to volume rendering [13]. Summed area tables allow a fast computation and lookup of the sum of a rectangular subset of values in a regular grid, such as a 2D texture. Diaz et al. use a summed up area table computation to analyze the depth buffer. The resulting table can then be used to rapidly compute the average depth value around a given pixel to determine if the pixel is surrounded by scene features in front of it rather than behind it. If the average neighborhood depth is significantly smaller than the local depth value at the current pixel, this pixel is considered to be occluded and the color is augmented accordingly. This produces a locally restricted ambient occlusion effect in the processed image.

3.5 Further Techniques

This chapter focuses mainly on those volume illumination techniques which are examined and compared in Chapter 6 or were otherwise extended or employed as part of this thesis. However, many other approaches to volume illumination exist which should be mentioned. An in-depth overview of many of these techniques can be found in the survey by Jönsson et al. [33].

There are several other publications which aim to use ambient occlusion in volume rendering. Stewart [84] was the first to extend this approach to volume rendering by proposing vicinity shading, which works only for displaying the structures belonging to one isovalue. Similarly, Wyman et al. [94] as well as Penner and Mitchell [62] have proposed techniques which work for isosurfaces only. However, there have been a number of DVR approaches realizing more general ambient occlusion in recent years. Hernell et al. [25] integrate the opacity of each voxel in a surrounding sphere. These values are then used to approximate the visibility of the voxel and the amount of incoming light. Hernell et al. [26] further improved this technique by reusing ambient occlusion results to efficiently compute global shadows and first order scattering effects with piecewise linear integration. Ljung et al. [48] obtain interactivity by exploiting a multiresolution framework, even when refining the transfer function. Ruiz et al. [76] use a framework of obscurances to produce illustrative renderings and saliency maps. Diaz et al. use 3D summed area tables to precompute volume densities in a voxel neighborhood and achieve ambient occlusion effects [13].

Schlegel et al. [77] extended a 3D summed area table to produce light source dependent directional soft shadows by using this data structure to precompute and store radiance extinction for cuboid areas in the dataset. By using a cone based algorithm similar to that employed by shadow volume propagation, the summed area table is used during rendering to quickly extrapolate occlusion and scattering information within the dataset from the light source to the current sampling point. Hadwiger et al. [22] adapted the deep shadow map approach to DVR with appealing results. In this approach, visibility functions are precomputed and compressed for each texel from the point of view of the light source. Older techniques by Nulkar and Mueller [59] and Zhang and Crawfis [97] use splatting to produce shadows in volume rendering.

Sunden et al. [86] introduced image plane sweep volume illumination, which builds on the ray casting process and can produce soft shadow and scattering effects. By controlling the order in which per-pixel rays are cast by starting with the pixel row closest to the light source in image space and progressing row by row away from the light source, a 2D illumination buffer which stores illumination information on

the plane spanned by the rays starting from the previous row is updated along with the regular ray casting process. At the same time, the current sample color can be attenuated by accessing the previous plane buffer. Jönsson et al. [32] presented an illumination technique that uses photon mapping which simulates the spreading of photons from a light source into the dataset, before collecting those photons visible from the virtual camera. To allow for interactive transfer function upgrades while using this computationally complex approach, so-called historygrams are stored in a binary data structure which contain previous light transport results. When the transfer function is changed, the historygrams can be used to update only the photon mapping for those parts of the scene for which the illumination has actually changed.

Chapter 4

Voreen - The Volume Rendering Engine

In this chapter, the Voreen framework is introduced. Its basic concepts are presented, as well as the integration of advanced volume illumination techniques into the framework.

The *Volume Rendering Engine* (Voreen) is a C++-based software framework aimed at the visualization of volumetric data [55, 54]. The project is based on a core library which provides many built-in volume visualizations such as ray casting implemented using the Open Graphics Library (OpenGL) and the OpenGL Shading Language (GLSL). However, many non-rendering related functions such as volume processing are implemented as well. Using this core library, the *Voreen Visualization Environment* (VoreenVE) provides a user interface which allows the interactive combination of core functions to build potentially complex visualization networks.

4.1 Concepts

The visualization of volumetric data can require numerous intermediate processing steps. For instance, a given dataset may be too large for interactive rendering and therefore has to be reduced in resolution in a preprocessing step. Once the final rendering is generated, the user may wish to overlay text or other additional visual information before displaying the image on the screen. In these scenarios, the visual programming paradigm applied by Voreen allows the quick extension of a basic process such as ray casting. Towards this end, Voreen uses a network concept in which data flows from a data source through several processing steps until a network leaf node is reached. Each of these steps is realized by a Voreen *processor*. If necessary, this extremely modular approach allows the quick exchange or addition of processors.

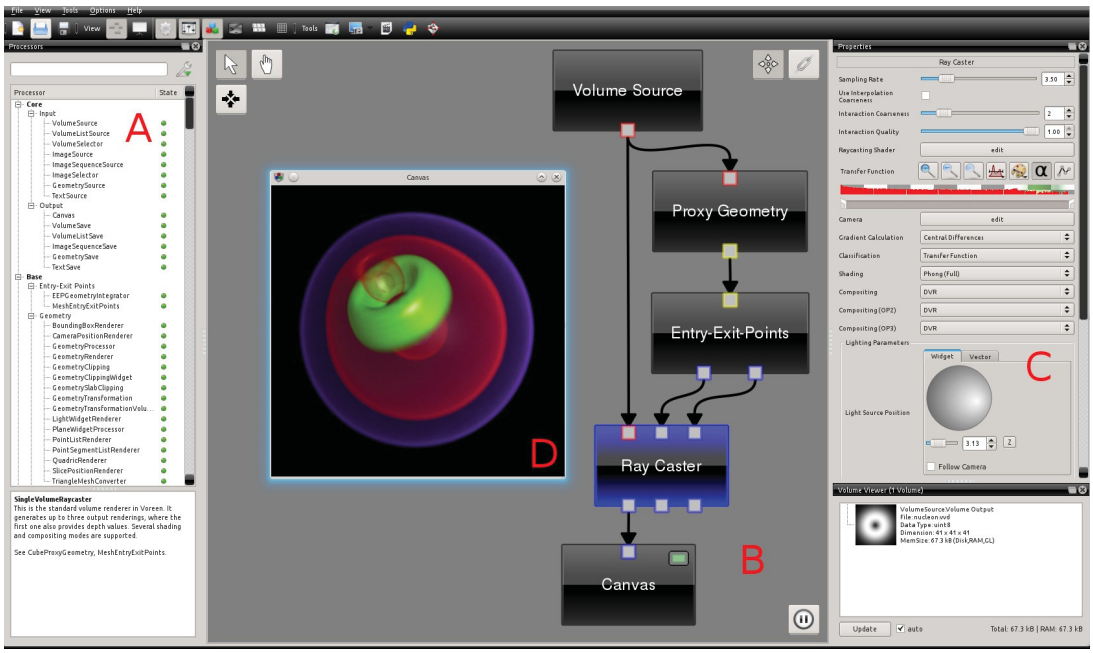


Figure 4.1: The VoreenVE application. The processor list (A) can be used to select and add new processors to the network editor (B). On the right, the properties of the currently selected processor are displayed (C). The final rendering produced by the network is displayed in a separate window (D) provided by the canvas renderer.

VoreenVE provides a list containing a large number of standard processors (Figure 4.1 A) which can be placed in the network using the VoreenVE network editor (Figure 4.1 B) to augment or change its functionality.

The behavior of each processor can be controlled by *properties* which are exposed to the user in a list to the right of the network (Figure 4.1 C). For instance, the step length and transfer function used during ray casting can be controlled by the properties of the ray casting processor.

The transfer of data from one processor to the next is realized using connected *ports* which differ in the type of data they support. Processors have a list of *inports* to receive and a list of *outports* to pass on data. Figure 4.1 provides an example for the data flow in a standard ray casting network used in Voreen. The volume source processor at the top of the network has only one *volume outport*. After loading the dataset, this port is used to transfer the volume data to the proxy geometry processor. This processor generates a proxy geometry and transfers it to the entry-exit-points processor using its *geometry outport*. Subsequently, this processor passes the generated

color-coded entry-exit-points to the ray casting processor using *render ports*. Finally, the ray casting processor uses its volume inport to receive the original data along with its render inports to receive the entry-exit-points. This enables the processor to render an image and pass it to the canvas processor, which uses an OpenGL canvas widget to display the result (Figure 4.1 D).

Intermediate rendering results like the entry-exit-points are hidden from the user by employing OpenGL *frame buffer objects* (FBOs). Instead of rendering directly to the screen, this functionality allows the application to write pixel values into a 2D texture object which can then be used as a regular texture to generate further renderings. This enables Voreen to merge partial rendering results of different branches of the network into a final rendering by combining or extending the contents of FBOs produced by previous processors. In most cases, after binding a texture containing a previous result, a screen-aligned rectangle that fills the entire view port is rendered which causes the execution of a fragment shader for each pixel. The shader is then used to look up intermediate results from the last rendering pass at the current pixel location, which can be potentially altered or replaced by the pixel result of the current processor.

4.2 Integration of Volume Illumination Techniques

An extension of Voreen by custom components is easily possible. This allows an efficient implementation of new rendering techniques, as basic parts of the rendering process (e.g., loading of data or calculation of entry-exit-points) can be re-used while only exchanging, for instance, the standard ray casting processor by the user's own version. As part of this thesis, those illumination models introduced in Chapter 3 that had not yet been implemented in Voreen were added: Spherical Harmonic Lighting, Directional Occlusion Shading, Multidirectional Occlusion Shading and Half Angle Slicing.

The Voreen source code is divided into *modules* to group related parts of the framework. Before compilation of the Voreen framework, the user selects the modules which should be added to the Voreen library using the CMake build system [51]. As a result, compilation time and binary size are reduced by avoiding components which users do not need for their work with Voreen. To add additional illumination techniques to the project, we therefore inherited the `VoreenModule` class to add a separate `AdvancedLightingModule` to the project. When starting VoreenVE, the constructor of this class will cause the processors implemented in the module to be added to the list of available processors using a factory pattern.

As mentioned in Chapter 3, some illumination techniques depend on the use

of slice-based rendering, while others can be used independently of the employed DVR paradigm. Those independent techniques were implemented in Voreen using ray casting due to its flexibility and performance. Furthermore, a few techniques are not directly part of the volume rendering pipeline at all, but require only a finished rendering and the associated depth values of the image to simulate lighting phenomena. The following sections describe how each of these three groups of techniques were integrated into Voreen.

4.2.1 Integration of Illumination Techniques Based on Ray Casting

As described in Section 2.2.3, volume ray casting is a process which first generates entry and exit points for rays from a proxy geometry. The volume is then traversed along a number of parallel rays, where each ray is defined by its respective entry and exit point. Voreen separates the extraction of the proxy geometry as well as the creation of the color coded entry-exit-points from the rendering process itself by employing specialized processors (see Figure 4.1). Since the advanced illumination techniques implemented in Voreen using ray casting only require standard entry-exit-points, they can each be realized with a single additional processor which employs the results of the regular entry-exit-point generating processors.

All of the techniques in this category require a preprocessing in which an additional auxiliary volume texture is generated. In the case of dynamic ambient occlusion, this procedure is rather lengthy and therefore handled by an external tool. This tool employs the Compute Unified Device Architecture (CUDA) developed by NVIDIA to speed up the massive computation that is necessary to complete the ambient occlusion histogram calculation and clustering for each voxel in a dataset (see Section 3.3.4). In the case of shadow volume propagation and spherical harmonic lighting, the preprocessing is handled in the `process()` method during the first evaluation of the visualization network, and afterwards each time the transfer function or the volume dataset have changed. To fill the auxiliary volume required for these techniques, the OpenGL capability to render directly into 3D textures slice by slice is exploited (see Figure 4.2). This is an extension of the regular render-to-texture functionality which Voreen already employs to pass rendering results between processors (see Section 4.1). In this approach, the FBO size is restricted to the size of a single slice of the target volume. For each slice along a chosen axis of the given dataset, a regular 2D OpenGL FBO is bound as the render target. The OpenGL view port is resized to the resolution of a slice in the target volume. Then, a screen-aligned, screen-filling rectangle which is congruent with the current slice is rendered. This has the effect of the active fragment shader being essentially executed for each voxel on the slice.

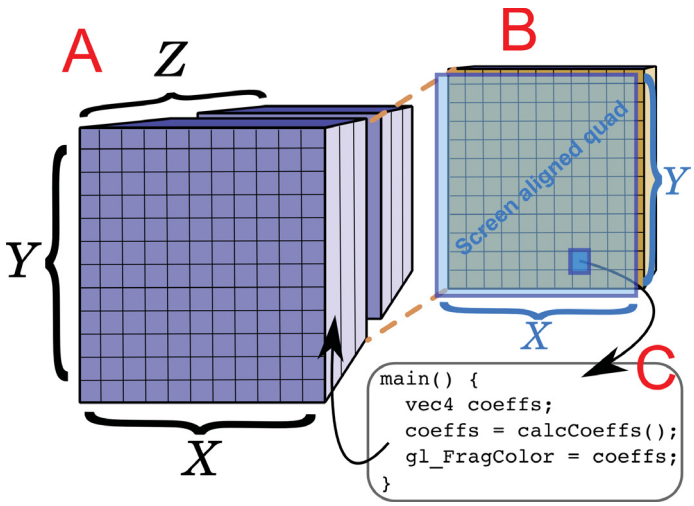


Figure 4.2: Rendering into a volume texture. First, an empty 3D texture in the required resolution $X \times Y \times Z$ is generated (A). The OpenGL view port is resized to $X \times Y$. For all slices along Z, render a screen aligned quad (B). This causes a fragment program to be executed for each pixel of the screen, which corresponds to a voxel in the volume texture (C). The resulting values containing illumination information are written into the voxel.

The resulting value is written into the associated voxel position of the dataset. By repeating this process for each slice, the auxiliary 3D texture is gradually filled with the illumination information that is calculated according to the respective technique. Since this approach exploits the parallel fragment computation power of the GPU, the preprocessing can usually be completed within the order of seconds for average sized volumes.

During the subsequent ray casting pass, the auxiliary volume textures can be accessed to augment the regular ray casting result by including the precomputed illumination information, depending on the chosen technique.

4.2.2 Integration of Slice-Based Illumination Techniques

Like ray casting, 3D texture slicing is part of the core functionality of Voreen. The process of calculating the necessary slice geometries is identical for the basic 3D slicing processor as well as the advanced illumination processors based on 3D slicing. The algorithm for slice extraction is based on the work by Rezk-Salama and Kolb [66] and available in the `VolumeSlicer` base class which all slicing based processors inherit. It employs a vertex shader to calculate the intersections of each slice plane

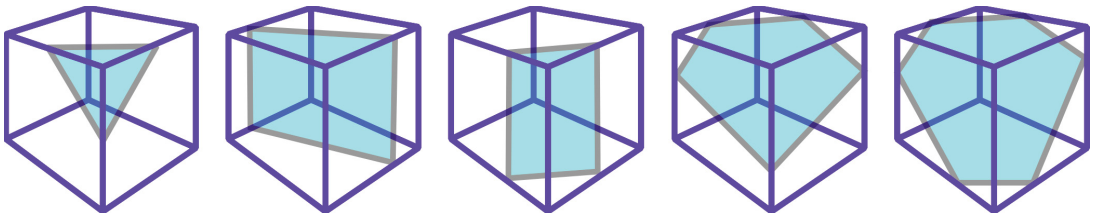


Figure 4.3: For 3D slicing, only six possible configurations for plane-box-intersection exist, not considering symmetries.

with the bounding box of the dataset on the fly. The algorithm exploits the fact that a box-plane intersection can produce polygons with at most six vertices located on the box edges (see Figure 4.3). Therefore, to calculate the slice geometries, six vertices in arbitrary positions are generated which are passed to the vertex shader. The edges of the dataset bounding box are searched for intersections with the slice plane, which is defined by the distance of the current slice to the virtual camera and the required orientation of the slice. If an intersection is found, one of the six incoming vertices is moved to the intersection position. If the intersection has less then six vertices after all edges have been searched, the location of the remaining unused vertices is moved to coincide with one of the actual intersection vertices, thereby producing degenerated triangles which will not produce any fragments during rasterization. The calculated slice geometry is then passed to the fragment shader in the form of 3D texture coordinates of the slice vertices on the bounding box edges. These polygon vertices are automatically interpolated by OpenGL for the rasterization stage, providing the fragment shader with a 3D texture position on the slice plane associated with the current fragment location. This coordinate can then be used to access the volume dataset and continue with the DVR pipeline. After all fragments have been processed, the rendering of the current slice is completed. This process is repeated for all slices in a loop, from front to back. The current slice is composited with the previously accumulated result in a FBO, yielding a final image at the end of the loop.

This basic approach is extended by the slicing based advanced lighting processors. Although the vertex calculation and the vertex shader remain the same, the fragment shader is updated to calculate additional illumination information for each slice. All advanced lighting processors in this category use a separate FBO to track illumination information. For each slice, a separate second rendering pass is performed to render illumination information into this auxiliary FBO, which will be used in the first rendering pass of the next slice to augment the rendering result.

4.2.3 Integration of Screen Space Illumination Techniques

Voreen provides a number of so-called *image processors*, which inherit from the class `ImageProcessor`. This set of processors usually has exactly one render inport and one render outport. A screen-aligned quad is rendered and causes the invocation a fragment shader for each pixel of the incoming image. The original image is accessed within the shader. For each pixel, the original color and transparency values are modified, depending on the functionality of the processor. For example, the `Grayscale` processor weights the incoming RGB values of each incoming pixel to calculate a grayscale value, which is then written to the resulting outgoing render target. Similarly, a Gaussian blur can be implemented by accessing and weighting the surrounding neighbors of each pixel.

The screen space illumination techniques implemented in Voreen depend on the fact that besides color and transparency, a depth buffer can be passed along between render targets that are connected within the network. Most images generated by Voreen use a screen aligned, screen filling quad to assign color values within the fragment shader. Without adaptation, this would lead to uniform depth values being produced by the conventional graphics pipeline, as all fragments originally lie on the same plane. By manually calculating the screen space depth of the current first hit point and writing it to the variable `gl_FragDepth` provided by GLSL in the fragment shader, depth values similar to those that would result from polygonal rendering can be added to a DVR result. These values are saved into the separate depth buffer of the render target, which can subsequently be passed on to the `DepthDarkening` or `ScreenSpaceAmbientOcclusion` processors along with the color buffer to calculate the difference in depth between neighboring pixels. This difference value is then used to approximate the amount of shadowing and, in the case of screen space ambient occlusion, color bleeding from surrounding pixels onto the pixel that is currently active in the fragment shader. The resulting modified pixel color is then written to the outgoing render target, resulting in an image with a rough approximation of global illumination.

Chapter 5

Advanced Light Material Interaction for Direct Volume Rendering

In this chapter we present a heuristic approach for simulating advanced light material interactions in the context of interactive volume rendering. In contrast to previous work, we are able to incorporate complex material functions, which allow to simulate reflectance and scattering. We exploit a common representation of these material properties based on spherical harmonic basis functions to combine the achieved reflectance and scattering effects with natural lighting conditions, i. e., incorporating colored area light sources. To achieve these goals, we introduce a modified spherical harmonic projection technique, which is not just tailored to a single material category, but adapts to the present material. Thus, reflecting and scattering materials as assigned by the transfer function can be captured in a unified approach. We will describe the required extensions to the standard volume rendering integral and present an approximation which allows us to realize the material effects while achieving interactive frame rates. By exploiting a combination of CPU and GPU processing, we are able to modify material properties and change the illumination conditions interactively. We will demonstrate the outcome of the proposed approach based on renderings of real-world data sets and report the achieved computation times.

5.1 Introduction

In recent years, sophisticated volumetric illumination models for DVR have been proposed. Interactive techniques are available, that support ambient occlusion [75, 78,

48], shadows [22] as well as scattering [38, 65]. While all these techniques brought realism in the area of volume graphics to a new level, an essential part needed to allow even more realistic image synthesis has only attracted little attention in the area of DVR so far: light material interaction. The appearance of an illuminated object depends on the combination of material properties, lighting properties as well as the light transport. While these three factors are often considered when illuminating polygonal objects, volume rendering techniques so far mainly focus on modeling the light transport. Regarding the material and the lighting properties only little work has been done, i. e., materials are usually reduced to assigning an emissive color through the transfer function and a point light source is used to illuminate the scene. However, only by considering complex reflectance functions in combination with area light sources, realistic illumination effects can be simulated. By not considering advanced material properties as well as area light sources, the quality of DVR images is rather limited compared to polygonal computer graphics.

Besides the material properties, light material interactions depend on the light source geometry, i. e., its extent and direction, and in some cases also the viewing direction. Therefore, changing the camera or the light source requires a re-computation. In this chapter we introduce an approach for integrating different material types into the volume rendering process and illuminating them using area light sources. With the presented approach, it becomes possible to simulate realistic materials of different kinds interactively. We will demonstrate how to incorporate complex reflectance and phase functions as well as area light sources. By transforming all these properties into a common basis representation, we are able to perform all computations required for rendering on the GPU and to achieve interactive frame rates, but still allowing to change the light properties and to a certain degree also the material properties. The key idea is to exploit a SH basis function representation for the material properties as well as the light sources. This allows us to integrate colored area light sources and to simulate their interaction with advanced materials interactively, i. e., the light direction as well as the camera position can be changed. To our knowledge, this is the first attempt which incorporates colored area light sources into DVR. Furthermore, we are not aware of other techniques dealing with advanced material properties in the area of interactive volume rendering in the sense that complex reflectance functions as well as scattering functions can be used. Additionally, we exploit an adapted SH projection approach to handle different material categories, while allowing an easy integration of conventional SH lighting [82].

5.2 Related Work

In the past, several interactive volumetric illumination models have been proposed. Many of these techniques are based on the optical models initially derived by Max [53]. One important approach in this area is the scattering technique presented by Kniss et al. [38]. By restricting themselves to a forward scattering phase function, the authors are able to achieve convincing results at interactive frame rates. The idea of constraining the phase function has been picked up by other researchers. Schott et al. extend this idea in order to integrate directional occlusion effects into a slice-based volume renderer [78]. Patel et al. have adapted the concept to not only incorporate occlusion information, but to also simulate light emitting structures [61]. However, since the techniques by Schott et al. as well as Patel et al. are based on a slice-wise front-to-back compositing, only headlights are possible. Ropinski et al. also use cone-shaped phase functions in order to simulate scattering effects together with an interactive volume processing technique, in which light is swept through an illumination volume [74]. While all these techniques are capable to produce convincing results, they are rather rough approximations, since scattering is assumed to be directed along one major axis, and not across the whole sphere. This is especially problematic when dealing with area light sources having multiple spatial components. An alternative approach has been introduced by Rezk-Salama [65]. He proposes the use of Monte-Carlo integration to allow scattering effects on a limited set of surfaces.

Besides the integration of scattering effects, researchers have also targeted the simulation of diffuse interreflections. As with the related work regarding scattering, we cover only those publications targeting interactive volume rendering. Ropinski et al. use a set of local histograms in order to capture the vicinity of a voxel [75]. While their technique is based on a rather expensive histogram preprocessing, Ljung et al. exploit a ray casting approach to determine the ambient illumination [48]. Both of these approaches can be applied to DVR as well as isosurface rendering. In contrast, Wyman et al. [94] and Beason et al. [2] focus on a preprocessing which allows to render isosurfaces under static illumination conditions. More recently, Banks and Beason have demonstrated how to decouple illumination sampling from isosurface generation in order to achieve sophisticated isosurface illumination results [1].

SH basis functions have been used in many areas of polygonal rendering. They allow to approximate spherical functions by exploiting a set of SH basis functions. Among other applications, this concept has been exploited to capture bidirectional reflectance distribution function (BRDFs) [92], as well as to allow the integration of low frequency shadowing effects [82]. It could also be shown, that SH basis functions can be used for other rendering effects, such as caustics or scattering [34].

Ritschel has applied SHs in the field of volume rendering to simulate low-frequency shadowing [71]. Similar as in this chapter, a GPU-based approach is exploited in order to compute the required SH coefficients. The ray traversal with increasing step size as proposed by Ritschel has also been integrated into our implementation.

As can be seen, all cited volume rendering publications rather target light transport than light material interaction. However, we believe that this area needs to be covered in order to generate more convincing renderings. To our knowledge only the style transfer function approach for illustrative visualization by Bruckner and Groeller [6] deals with advanced material properties in the area of interactive volume rendering so far.

5.3 Light Material Interaction

In the following, we will incorporate advanced material properties as well as colored area light sources into interactive volume rendering. We will refer to reflectance and phase functions as material functions, and will call materials having significant differences as belonging to other material categories. When considering material and lighting properties together with light transport, the radiance $L(\vec{x}, \vec{\omega_o})$ which leaves from position $\vec{x}$ inside a volume in direction $\vec{\omega_o}$ can be defined as:

$$L(\vec{x}, \vec{\omega_o}) = L_e(\vec{x}, \vec{\omega_o}) + \int_{4\pi} M(\vec{x}, \vec{\omega_i}, \vec{\omega_o}) \cdot L_i(\vec{x}, \vec{\omega_i}) d\vec{\omega_i}$$

Here, $L(\vec{x}, \vec{\omega_o})$ is the sum of the emission $L_e(\vec{x}, \vec{\omega_o})$ and the local light material interaction, which is expressed by the integral over the unit sphere. To compute the local light material interaction, the material function $M(\vec{x}, \vec{\omega_i}, \vec{\omega_o})$ is modulated by the incident intensity $L_i(\vec{x}, \vec{\omega_i})$ coming from direction $\vec{\omega_i}$. Since we deal with volume data, where the notion of a surface does not inherently exist, we do not weight this modulation by the clamped cosine term of the surface normal and ω_i. Given the original volume intensity $f(\vec{x})$, a surface normal could be approximated by considering the intensity gradient $\bigtriangledown\tau(f(\vec{x}))$. However, this approach would not comply with the definition of the material function $M(\vec{x}, \vec{\omega_i}, \vec{\omega_o})$, which is in contrast to surface reflectance functions not defined over the hemisphere, but over the whole sphere. So far, only local effects, such as emission and local light material interactions, i. e., out-scattering or reflectance, have been incorporated. To integrate environmental effects, as in-scattering or color bleeding, the unmodified incident intensity $L_i(\vec{x}, \vec{\omega_i})$ has to be modulated with $L_{mi}(\vec{x}, \vec{\omega_i})$, which is based on the environment of $\vec{x}$:

$$L(\vec{x}, \vec{\omega_o}) = \int_{4\pi} M(\vec{x}, \vec{\omega_i}, \vec{\omega_o}) \cdot L_{mi}(\vec{x}, \vec{\omega_i}) \cdot L_i(\vec{\omega_i}) d\vec{\omega_i} \tag{5.1}$$

In Section 5.4 we will describe how to compute L_{mi} in order to integrate diffuse interreflections as well as in-scattering. Please note that we use a slightly modified notion for L as compared to L_i and L_{mi}. While L is defined based on the direction $\vec{\omega_o}$ going out from $\vec{x}$, L_i and L_{mi} are defined based on the incoming direction $\vec{\omega_i}$ arriving at $\vec{x}$. This means that these terms represent all light arriving at $\vec{x}$ from $\vec{\omega_i}$, regardless of the position of the viewer. This unmodified light intensity is then weighted by the material term $M(\vec{x}, \vec{\omega_i}, \vec{\omega_o})$ which potentially modulates the intensity due to reflection.

With the simplified illumination model specified in Equation 5.1, we are able to incorporate complex material functions specified through M. M, L_{mi} as well as the incident light L_i are defined over the unit sphere. Obviously, the integration over the sphere involves a significant computing effort and cannot be made for each frame. However, when assuming that the functions are continuous, we can exploit SH basis functions in order to approximate Equation 5.1 and achieve interactive frame rates. A brief introduction to SHs and their properties can be found in Section 3.3.3.

5.4 Realizing Advanced Material Effects

While most SH-based material representations for polygonal data are tailored to support one specific material category, i. e., highly reflective or scattering, in volume rendering the material category may change just by modifying the transfer function. When for instance a lower opacity is assigned through the transfer function, stronger scattering can be expected. Therefore, a specialized SH compression method is needed in order to support different material categories.

In our approach, the SH projection adapts to the material function M at each sample position $\vec{x}$. In particular, we consider the transparency $\tau(f(\vec{x}))$ assigned through the transfer function, as well as the gradient length $|\bigtriangledown \tau(f(\vec{x}))|$, depicting the heterogeneity of the medium. To illustrate our approach, we would like to emphasize the two existing extreme cases. First, when $\tau(f(\vec{x}))$ represents high transparency and $|\bigtriangledown \tau(f(\vec{x}))|$ is small, we can assume that the whole unit sphere centered at $\vec{x}$ affects its illumination. In contrast, when $\tau(f(\vec{x}))$ depicts a rather opaque medium and $|\bigtriangledown \tau(f(\vec{x}))|$ is large, only the front facing hemisphere affects the illumination of $\vec{x}$, since $\vec{x}$ can be considered as a part of a boundary surface [42]. The other possible combinations of the extrema of $\tau(f(\vec{x}))$ and $|\bigtriangledown \tau(f(\vec{x}))|$ lie somewhere between

on this scale. So loosely speaking, we consider the whole contribution of the unit sphere, when no surface-like structure can be assumed depending on $\tau(f(\vec{x}))$ and $|\bigtriangledown \tau(f(\vec{x}))|$, and just the hemisphere, when a surface-like structure is present.

For a low-frequency spherical function $g(s)$, it can be shown that Monte-Carlo integration over a sufficiently high number of samples is sufficient to compute Equation 3.8 and project $g(s)$ into SH space. This is achieved by using uniform distributions of samples over the unit sphere to compute $g(s)$. However, in the case where we deal with a surface-like structure, we need to sample the front facing hemisphere only, since the back facing hemisphere is not assumed to contribute to the illumination of $\vec{x}$. Therefore, we have chosen to integrate an adaptive sampling in order to perform the SH projection of the environmental lighting. By treating the direction s as a random variable associated with a probability density function $p(s)$, the SH coefficient calculation given in Equation 3.8 can be reformulated so that it equals the expected value E of a new function $h(s)$:

$$c_j = \int_{4\pi} Y_j(s)g(s)ds = \int_{4\pi} \frac{Y_j(s)g(s)}{p(s)}p(s)ds = \int_{4\pi} h(s)p(s)ds = E(h(s)) \tag{5.2}$$

This reformulation allows us to use the Monte-Carlo approach to approximate the integral discretely with a finite sum over a set S of random samples which are chosen according to $p(s)$:

$$E(h(s)) \approx \frac{1}{|S|}\sum_{i=1}^{|S|} h(s_i) \tag{5.3}$$

To obtain $h(s)$ and evaluate this equation, the probability-density function $p(s)$ needs to be defined. In the spherical case, it has to fulfill the requirement $\int_{4\pi} p(s)ds = 1$, which can be achieved by setting $p(s)$ to $\frac{1}{4\pi}$. In our case, we have to incorporate both a hemispherical or spherical region in order to deal with translucent and surface-like structure properties. Since these properties are continuous, we introduce a below-surface cut-off angle $\alpha_{\vec{x}}$, which is proportional to the transparency of the voxel at $\vec{x}$:

$$\alpha_{\vec{x}} = \tau(f(\vec{x})) \cdot \frac{\pi}{2}$$

For a completely transparent voxel, this angle is equal to $\frac{\pi}{2}$, and for a completely opaque voxel this angle is equal to 0. During Monte-Carlo sampling, only rays are considered if their angle with $\bigtriangledown\tau(f(\vec{x}))$ is less or equal to $\frac{\pi}{2} + \alpha_{\vec{x}}$. To fulfill the requirements of the probability density, 2π is used as a base value for the hemisphere, and the additional ring below the surface given by $\alpha_{\vec{x}}$ is included as $2\pi + 2\pi \cdot \sin\alpha_{\vec{x}} = 2\pi(\sin\alpha_{\vec{x}} + 1)$. This term is equal to 2π for $\alpha_{\vec{x}} = 0$ and to 4π for

$\alpha_{\vec{x}} = \frac{\pi}{2}$, which corresponds to the whole sphere. Using these considerations to define

$$p(s) = \frac{1}{2\pi(\sin\alpha_{\vec{x}} + 1)}$$

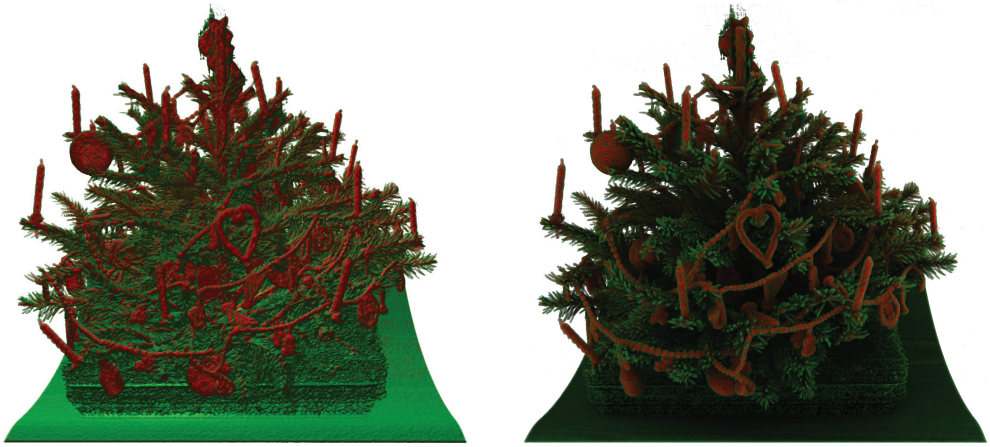
for a given location $\vec{x}$ and combining Equations 5.2 and 5.3, we have obtained the final Monte-Carlo integrator to approximate each SH coefficient:

$$c_j \approx \frac{2\pi(\sin\alpha_{\vec{x}} + 1)}{|S|} \sum_{i=1}^{|S|} Y_j(s_i) g(s_i) \tag{5.4}$$

We will use this integrator to project various material- and lighting-related spherical functions $g(s)$ into SH space.

5.4.1 Color Bleeding Effects

By using the modified Monte-Carlo integration, the computation of the color bleeding effects can be done easily by performing SH projection. This is a straightforward extension to the diffuse shadowed radiance transfer. However, we omit a weighting with the clamped cosine of the angle between $\bigtriangledown\tau(f(\vec{x}))$ and the light vector, since a similar behavior is already incorporated in our modified sampling approach described above. Thus, from each voxel, rays are cast in random directions ω_i based on our modified sampling scheme. The color and intensity changes, which occur to the ray while traveling through the medium, are modeled by evaluating the front-to-back compositing volume rendering integral in order to obtain $L_{mi}(\vec{x}, \vec{\omega}_i)$. This is different from the approach by Ritschel, where only absorption and thus no chromaticity effects are considered [71]. After the ray casting for the current sample has been completed, each of the accumulated color channel values are used as a weight for the values $Y_l^m(\vartheta_s, \varphi_s)$ of the current sample, which are added to each coefficient sum of the SH projection defined in Equation 5.4. When substituting the thus computed L_{mi} in Equation 5.1, renderings as shown in Figure 5.1 (b) can be generated. When comparing the outcome with conventional gradient-based shading (see Figure 5.1 (a)), it can be seen that a better depth separation is achieved. In this approach, we can use two different ways of incorporating material properties. When evaluating the material function M for each sample on the ray, it directly influences the bleeding color. However, due to the fact that M is considered in the SH projection, it cannot be exchanged without performing a new SH projection. Alternatively, M could be neglected during the ray-marching and instead the transfer function color could be taken into account. In this case, material functions could be exchanged during



(a) (b)

Figure 5.1: The Christmas tree data set, with gradient-based lighting (a) and with a color bleeding contribution compressed using SH basis functions (b).

rendering interactively, as long as they have a similar hue.

5.4.2 Scattering Effects

With the approach described in the previous subsection, we are able to incorporate indirect illumination effects by exploiting the computation of L_{mi}. However, to further increase the level of realism for certain materials, scattering effects need to be incorporated. Hao and Varshney could show in the context of subsurface scattering, that although scattering is in principle a global effect, its influence is rather local due to the exponential falloff of light [23]. We have adopted this insight, and exploit an additional scattering pass. Since we have already computed and SH projected the L_{mi} for all $\vec{x}$ as described in the previous subsection, we can exploit this information when computing the scattering contribution. While previous interactive approaches have constrained scattering just to a cone angle [38, 74], we are thus able to perform the scattering over the whole unit sphere, or the hemisphere depending on the adapted Monte-Carlo sampling. Assuming that multiple scattering dominates, the in-scattering reaching $\vec{x}$ from its neighborhood can be computed as follows:

$$L'_{mi}(\vec{x},\vec{\omega}_i) = \sum_{\vec{x}_i \in N(\vec{x})} \frac{1}{\pi} R_d(|\vec{x}_i - \vec{x}|) \cdot L_{mi}(\vec{x}_i,\vec{\omega}_i) \cdot \frac{1}{|N(\vec{x})|} \tag{5.5}$$

In this equation, a sum over the previously computed weighted SH coefficients

of voxels within a neighborhood $N(\vec{x})$ around the current voxel at $\vec{x}$ is calculated. $R_d(r)$ specifies a so-called dipole factor, which depends only on the distance of the current voxel to the respective neighboring voxel $\vec{x}_i$ and the material that the current voxel is supposed to represent. This factor describes the diffuse reflectance obtained by a single dipole approximation for multiple scattering as described by Jensen et al. [31]. We used the material parameters as specified in their work to define $R_d(r)$ and to generate the images showing subsurface scattering in this chapter. Jensen et al. showed that this approximation achieves a comparable accuracy as using the diffusion approximation with a volumetric source distribution. Although the parameters have been originally acquired to simulate subsurface scattering, we were able to achieve compelling scattering effects by using these parameters. Thus, Equation 5.5 enables us to compute the scattering contribution for each voxel $\vec{x}$ and a given light direction $\vec{\omega}_i$. We have specified this equation in its discrete form to depict that this additional scattering pass is computed based on the already present SH coefficient volumes, containing the coefficients for L_{mi}. Since these volumes are by nature given in a discrete data structure, the $\vec{x}_i$ in Equation 5.5 refer to actual voxel positions instead of sample positions.

Equation 5.5 is inspired by the subsurface scattering equation introduced by Hao and Varshney [23]. Since we are not interested in subsurface scattering, but true volumetric scattering, we have neglected the cosine weighting from the original equation. Furthermore, we have dropped the Fresnel term, and modified the weighting to comply with the volumetric scattering neighborhood $N(\vec{x})$. Notice that the color bleeding contribution is also incorporated in Equation 5.5, since we use L_{mi} as derived in the previous subsection. In our implementation, the neighborhood size $|N(\vec{x})|$ is assumed to be constant within the entire volume. Obviously, the resolution and the average size of structures contained in the data set should be taken into account in order to specify $|N(\vec{x})|$. However, in our case we could always achieve good results, when setting $|N(\vec{x})|$ to 10% of the maximum volume dimension. An alternative approach would be to adapt it based on the material properties of the current region around $\vec{x}$.

The main benefit of our method is the fact that we can again use SH projection in order to efficiently store the L'_{mi} as the results of this scattering pass. Thus, as with conventional SH lighting, the light properties can be changed interactively. Figure 5.2 shows the application of Equation 5.1, when we substitute L_{mi} with L'_{mi} and use different parameters for R_d.

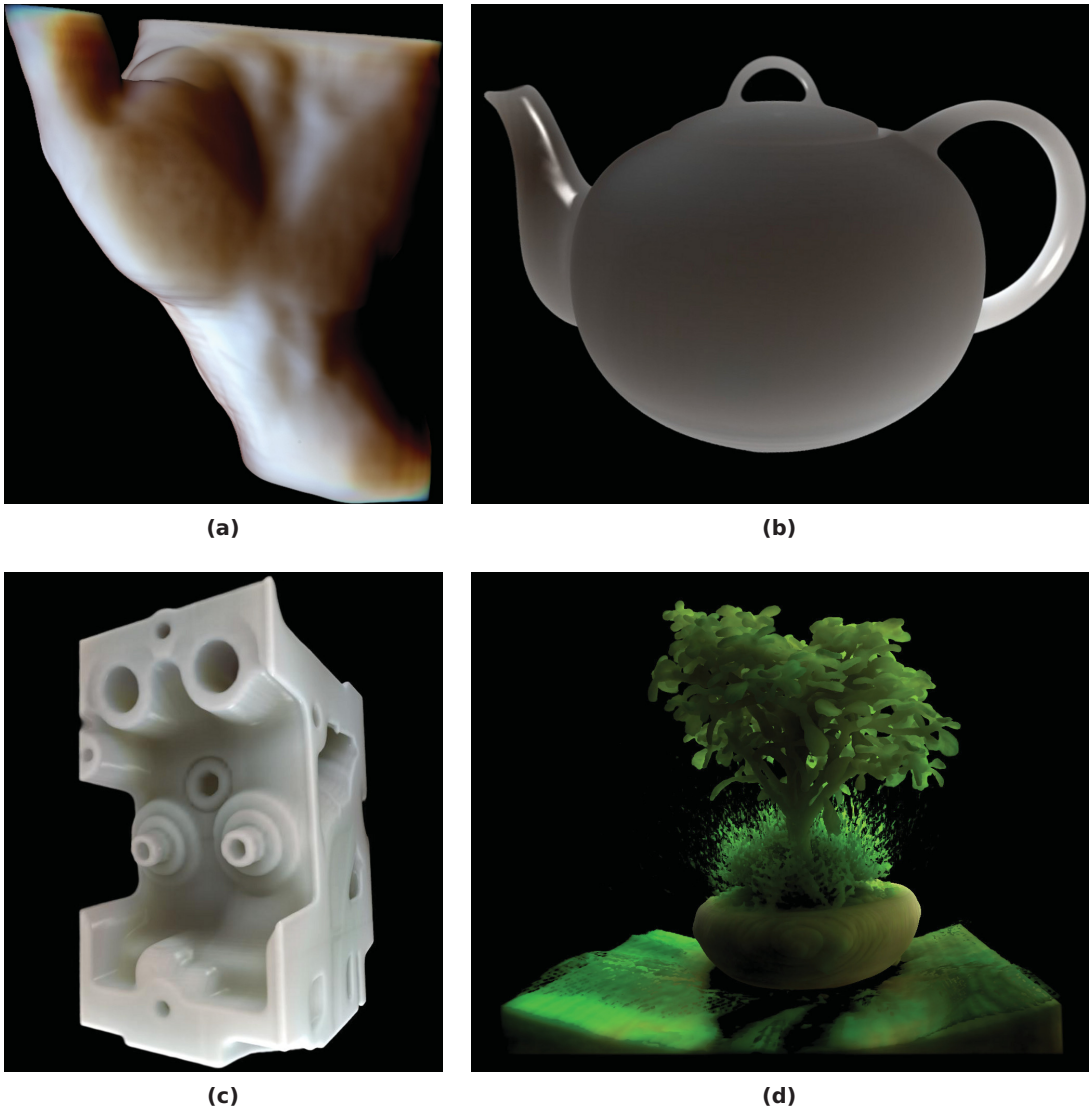
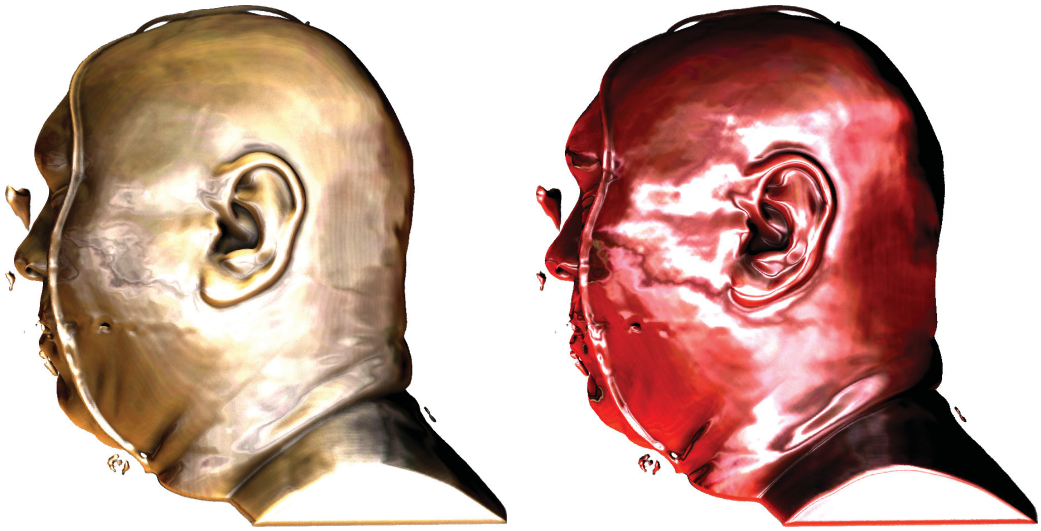


Figure 5.2: Application of the interactive scattering technique to different volume data sets: For the hand data set, we have set R_d to allow a realistic skin appearance due to subsurface scattering (a). For the teapot data set, R_d has been set to mimic a porcelain-like appearance (b). For the engine an R_d was selected that simulates the behavior of marble (c). The bonsai data set shows the occurrence of brown and green color bleeding effects (d).



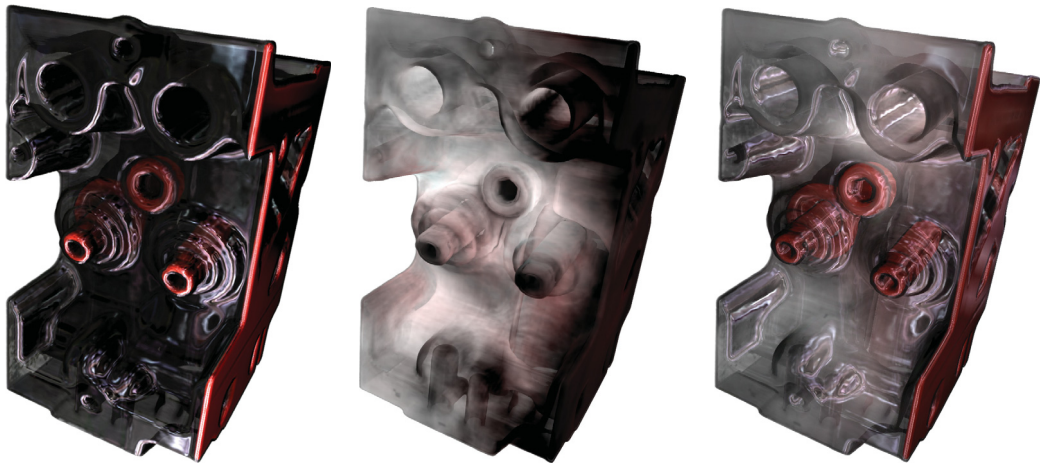
(a) (b)

Figure 5.3: In this example we use BRDFs as material functions M. The BRDFs have been SH projected by using 1250 samples distributed over the hemisphere. In (a), an aluminum bronze material is simulated, while (b) shows a specular phenolic material.

5.4.3 Local Material Effects

Now that we are able to compute the environmental illumination contribution, i. e., the in-scattering reaching $\vec{x}$, we can focus on the realization of the local material effects. As seen in Equation 5.1, this is specified by the material function $M(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)$. Since M is also a spherical function, we can compress it using standard SH projection techniques. Therefore, we cast rays from the center of a sphere representing the material to the outside, and project the hit values on the sphere's surface. Since this can be done once in a preprocessing for each material added to the system, performance is not an issue regarding this step. However, to simplify the rendering process, we have decided to use the same number of SH bands as in the other SH projections. At the same time, by increasing the number of samples, we can improve the quality of the approximation. Examples where we have used 1250 samples distributed over the hemisphere are shown in Figure 5.3. In this case, we have used data from the MERL BRDF library [52] to describe the material function M for surface-like structures having a high degree of reflectance. The actual representation in our implementation is described in Section 5.5.

We have used the same projection technique to represent light probes in order



(a) (b) (c)

Figure 5.4: Different effects demonstrated to the engine data set: the application of $M \cdot L_i$ (a), the environmental light contribution $L_{mi} \cdot L_i$ (b), and the simultaneous application of both effects (c).

to integrate natural lighting conditions depicted as L_i. Thus, we have all necessary parts to compute Equation 5.1 and can compute the illumination at $\vec{x}$ by evaluating Equation 3.9. However, while SH integration can be done easily for two functions (see Equation 3.9), dealing with three functions, i. e., M, L'_{mi} and L_i, becomes significantly more complex. While existing techniques allow the realization of the integration of three functions [83], we have decided to use a simpler technique due to performance issues. Therefore, we approximate the integration by using the following equation, which is not physically correct but led to convincing visual results:

$$L(\vec{x}, \omega_o) = \int_{4\pi} M(\vec{x}, \vec{\omega}_i, \vec{\omega}_o) L_i(\vec{x}, \vec{\omega}_i) d\vec{\omega}_i \cdot \int_{4\pi} L'_{mi}(\vec{x}, \vec{\omega}_i) L_i(\vec{x}, \vec{\omega}_i) d\vec{\omega}_i. \tag{5.6}$$

Intuitively, we simplify the triple product to two successive double products. First, we integrate over M and L_i to obtain the local material effects. The result of this integration is modulated by the environmental illumination described by the second integral. Applications of the introduced rendering technique are shown in Figure 5.4. Figure 5.4 (a) shows the engine dataset when applying only the left integral of Equation 5.6 using an area light source and a glass-like BRDF, while Figure 5.4 (b) shows the color bleeding contribution due to the right integral. Figure 5.4 (c) shows the combination of both effects as described in Equation 5.6.

5.5 Implementation

The concepts described in this paper have been integrated into a GPU-based volume ray caster implemented in C++ by using OpenGL in combination with GLSL. The coefficients for the SH projections of L_{mi} and L'_{mi} are stored in 3D coefficient volumes, whereby those representing L_{mi} can be discarded, after L'_{mi} has been computed, since they are not accessed during rendering. To generate the SH coefficient volumes, we exploit FBOs in conjunction with multiple render targets. The actual ray marching is performed in a fragment shader, while rendering slice by slice into the 3D coefficient texture. While the GLSL specification describes functionality for generating random numbers, this feature is not implemented yet. Therefore, we have decided to generate the random samples for the Monte-Carlo sampling on the CPU in form of normalized Cartesian vectors and upload them to the shader as a 2D texture. The same is done for the values of $Y_l^m(\vartheta_s, \varphi_s)$ for each of the sample directions. By using the random samples as directional vectors, rays are cast within the shader. After a ray has been terminated, its contribution is multiplied with the values $Y_l^m(\vartheta_s, \varphi_s)$ obtained from the second 2D texture and added to the total integration sum. Thus, we obtain all $(L+1)^2$ SH-coefficients for the current voxel, which are distributed to the attached render targets.

To speed up the computation, we have integrated early ray-termination based on a ray opacity threshold of 95%. Furthermore, since the ray casting is performed directly in volume space $[0...1]^3$, a sample ray is terminated if one of its position-coordinates reaches 0 or 1.

Currently multiple render target functionality is limited to 8 simultaneous output targets. Therefore, we have chosen $L = 3$ as the highest band, yielding 16 coefficients per color. We generate these 16 coefficients by rendering into 4 slices with 4 channels simultaneously. However, to reduce memory requirements, we exploit the fact, that human beings are less sensitive to chromaticity variations as compared to variations in luminance [47]. Therefore, we have chosen to use only a 2-band projection for the chromaticity values while keeping the 4-band projection for the luminance. Thus, the three color-coefficient sets sum up to only $3 \cdot 4 = 12$ coefficients altogether, which can be handled using three textures attached to a FBO during the projection, as well as three 3D volume textures during rendering. Even when using chromaticity and luminance, this adds up to $4 + 3 = 7$ render targets, which makes this operation possible in a single pass.

While the SH coefficient volumes are generated on the GPU, the projection of the material functions and the light probes is less time critical and can be performed on the CPU, where we generate a partitioned 2D texture representation to be passed

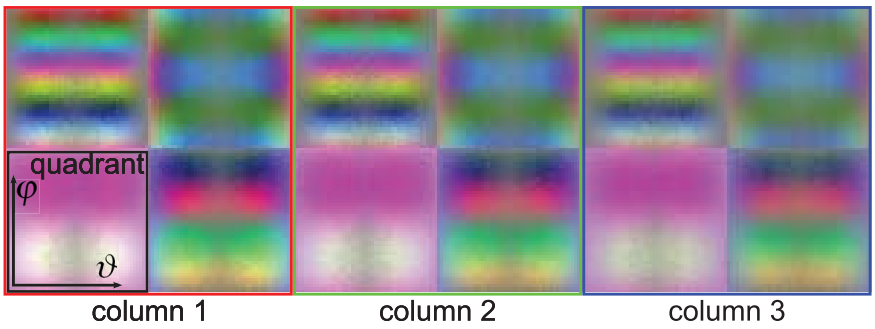


Figure 5.5: The BRDF coefficient texture for the aluminum-bronze material. One quadrant holds 4 coefficients for each ω_o. The x-axis spans the values $\vartheta = 0...\pi$ and the y-axis the values $\varphi = 0...2\pi$.

to the shader during rendering. When using a 4-band compression, the resulting texture is partitioned as shown in Figure 5.5. For each ω_o, a combination of texels represents the 16 SH coefficients needed for the ω_i. The texture consists of three columns, having 4 quadrants of size $N \times N$ texels, when using N^2 random samples. The first 4 coefficients (bands 0 and 1) are saved in the lower left, the next 4 in the lower right (the first 4 coefficients of band 2), the next 4 in the upper left (one coefficient of band 2 and 3 coefficients of band 3) and the final 4 in the upper right (the remaining 4 coefficients of band 3). The x-axis of each quadrant represents ϑ of ω_o, ranging from 0 to π, and the y-axis represents φ of ω_o, ranging from 0 to 2π. Thus, we have all needed coefficients in a GPU-friendly format and the integration as described in Subsection 5.4.3, can be performed during rendering in a fragment shader by using matrix and vector operations.

5.6 Performance Results

Table 5.1 shows a comparison of the performance of the SH projection process for different parameters. The tests were performed with three different data sets, having a resolution of $128 \times 128 \times 128$, $256 \times 256 \times 128$ and $256 \times 256 \times 256$ voxel. All tests have been conducted on a Intel Core2 Quad CPU Q9450, running at 2.66 GHz, with 4 GB of main RAM and an nVidia GeForce 9800 GTX with 512 MB RAM. In the columns the SH projection times as well as the frame rates are shown for shadows only, scattered shadows and scattered shadows and color bleeding. During all tests as well as to generate the renderings shown in the paper, we have used downsampled coefficient volumes of about 33% of the original size, while discarding voxels with an opacity of zero. Assuming that the radiance transfer is of low-frequency, the

	Shadows	+Scattering	+Bleeding
128^3	0.9s / 24 fps	21.3s / 17 fps	58.9s / 13 fps
$256^2 \times 128$	2.6s / 20 fps	70.2s / 14 fps	180s / 5 fps
256^3	3.4s / 6 fps	149s / 3 fps	388s / 1 fps

Table 5.1: Timings for SH projection and rendering with data sets of different voxel sizes. From left to right: shadows only, scattered shadows, scattered shadows with color bleeding.

downsampling is a simple and effective optimization. Furthermore, we have used increasing ray step size [71]. The windows resolution used to compute the given frame rates was 400×400.

As it can be seen in Table 5.1, for data sets of $128 \times 128 \times 128$ voxels, we achieve interactive frame rates for all compared SH techniques. This allows the user to inspect the data set for a given transfer function interactively. Furthermore, for the shadowing technique, the SH projection is still fast enough to allow an interactive change of the transfer function. Obviously, the scattering pass results in an additional performance penalty, such that the transfer function cannot be changed without waiting for the SH projection. However, since the reprojection has to be performed only when the transfer function is changed, the actual rendering can still be explored at $14 - 17$ frames, i. e., camera and light parameters can be changed freely. As described in Subsection 5.4.1, the material function M can only be exchanged interactively when it is not considered during the SH projection. In fact, in many application areas of medicine or computational fluid dynamics, predefined transfer functions are used, which could also involve M. Table 5.1 also shows that when additionally incorporating the color bleeding during the shadowing pass, both SH projection as well as rendering take significantly longer. This delay is due to the fact that three additional SH coefficient volumes have to be generated and accessed during rendering.

5.7 Conclusions and Future Work

In this paper we have demonstrated how to integrate realistic light material interaction effects into interactive volume rendering. With the proposed techniques we are able to exploit material functions of reasonable frequency. By using a modified SH projection which has been adapted to the needs of volume rendering we are able to represent these effects using a common representation. As we have shown, this also allows the seamless integration of conventional SH lighting effects. To our knowledge, this is the

first application of non cone-shaped phase functions in the area of interactive volume rendering. To achieve this, we have explained how to incorporate the desired effects within the volume rendering integral, and have described our hybrid CPU/GPU implementation. We believe that the proposed concepts bring the realism of volume rendered images to a new level.

While we were able to produce high-quality imagery at interactive frame rates, there are still several open issues, which could potentially be addressed in the future. The main drawback of SH-based techniques is the restriction to approximate low-frequency functions only. While this is sufficient for most phase functions, for some materials having sharp specular highlights this might not be sufficient and Gibbs ringing may occur. Therefore, alternative concepts should be investigated for these scenarios, such as the SH approximation method by Zafar et al. [96], or wavelet compression techniques, which have been proved useful in the area of polygonal rendering [85] and might be also used in DVR. Another issue with the proposed technique is the availability of appropriate material functions. While several simplified models exist, having a true volumetric capturing would be of great interest. Finally, since the presented technique can be considered as approximative, it should be evaluated with respect to perceptual requirements.

Chapter 6

About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering

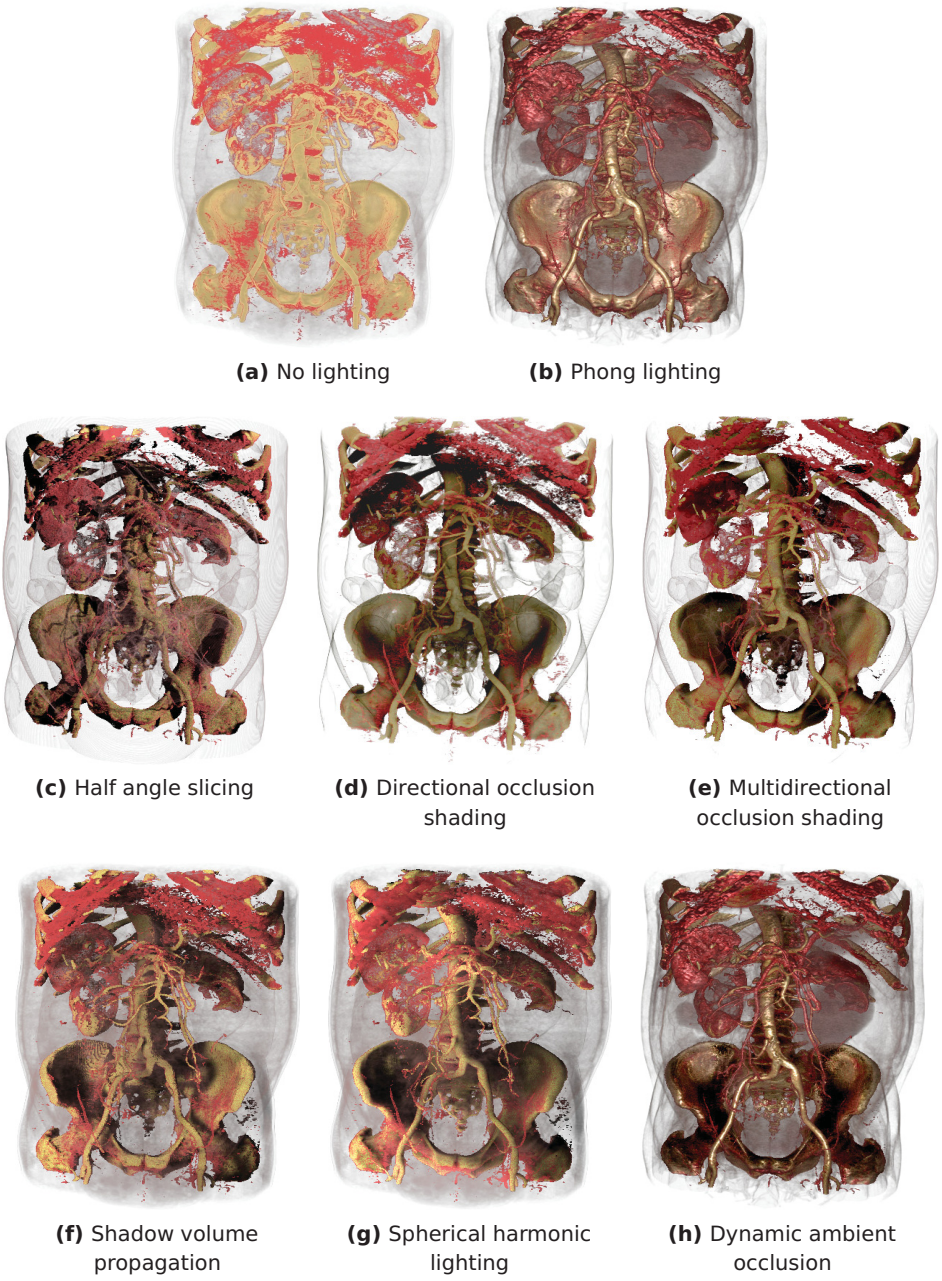
In this chapter, we present a user study in which we have investigated the influence of seven state-of-the-art volumetric illumination models on the spatial perception of volume rendered images. Within the study, we have compared gradient-based shading with half angle slicing, directional occlusion shading, multidirectional occlusion shading, shadow volume propagation, spherical harmonic lighting as well as dynamic ambient occlusion. To evaluate these models, users had to solve three tasks relying on correct depth as well as size perception. Our motivation for these three tasks was to find relations between the used illumination model, user accuracy and the elapsed time. In an additional task, users had to subjectively judge the output of the tested models. We discovered statistically significant differences in the testing performance of the techniques. Based on these findings, we have analyzed the models and extracted those features which are possibly relevant for the improved spatial comprehension in a relational task. We believe that a combination of these distinctive features could pave the way for a novel illumination model, which would be optimized based on our findings.

6.1 Introduction

In recent years, the development and improvement of off-the-shelve GPUs has led to the proposition of several interactive advanced volumetric illumination models.

While most researchers claim an improved perception of volume data using their technique, little work has been done verifying these claims and comparing the different approaches. In this chapter, we present a user study which evaluates the perceptual qualities of seven state-of-the-art volumetric illumination models. The evaluated models have been chosen based on their novelty, the number of citations as well as actual spread in real-world applications. To ensure comparability, we focused solely on DVR techniques. Besides the conventional gradient-based Phong illumination model [42], which we have selected due to its pervasiveness, we selected three models based on slice compositing and three models which can be combined with ray casting based techniques. The slice-based techniques consist of the established half angle slicing [37] and two newer models, directional occlusion shading [78] and multidirectional occlusion shading [89]. Among the techniques not exclusively based on slicing, we have decided to evaluate the recently proposed shadow volume propagation technique [74], spherical harmonic lighting [71] and dynamic ambient occlusion [75]. Fig. 6.1 shows an image comparison of these seven techniques.

Since this is the first study comparing advanced volumetric illumination models, we decided to minimize potential errors by reducing the study complexity. We focused on static images instead of dynamic images or interactive applications since they occur in many everyday tasks. This also implies that no stereoscopic perception cues were available. Therefore, the layman users who participated in our study had to rely on the visual cues of monoscopic perception such as occlusion and cast shadows. For the design of the conducted tests, we have considered the perceptual processes relevant for scene comprehension. First, depth perception is crucial to comprehend the arrangement of a scene [91], as it allows the observer to judge the distance between an object and himself as well as other structures in the image. A conceptually more complex perceptual task required to judge scene arrangements is size perception, which strongly depends on depth perception [17]. Consequently, both depth perception as well as size perception are essential in order to judge the spatial relationships within a scene. Therefore, in the first two of our exercises, we tested the participants' ability to recognize depth in a volume rendered image, while the third task involved the cognition of the size of different features. Our goal was to reveal the perceptual advantages and disadvantages of the evaluated models based on these tests. Besides this analysis of quantitative measurements, we also examined which techniques were subjectively preferred by the users when given the choice between two images. After describing all conducted tests in detail, we will discuss our results and distill them to identify perceptually relevant features of volumetric illumination models. Finally, we will present guidelines to select the right illumination model for a given task.



(a) No lighting **(b)** Phong lighting

(c) Half angle slicing **(d)** Directional occlusion shading **(e)** Multidirectional occlusion shading

(f) Shadow volume propagation **(g)** Spherical harmonic lighting **(h)** Dynamic ambient occlusion

Figure 6.1: An overview of the seven volume illumination techniques which we implemented and tested. The renderings of a CT scan of a human body differ only with respect to the illumination method, as other parameters remain constant. On the upper left, a rendering with no illumination at all is displayed for reference.

6.2 Related Work

The influence of lighting on the perception of a rendered scene in general has been the subject of extensive research. Due to the similarity with the qualities of the tested illumination models, we mainly focus on those studies investigating shadows or similar phenomena. Wanger et al. found that the existence of shadows significantly improved the recognition of spatial features as well as object size [91]. Wanger conducted a further study on the influence of shadow quality on the recognition of spatial relationships, object size and object shape in computer generated images [90]. In his tests, users were subjected to images of objects with no shadows, soft shadows and hard shadows. His results showed no significant relation between spatial comprehension and the softness of shadows. However, recognition of the shape of objects improved with hard shadows, but even decreased when using soft shadows as compared to using no shadows at all.

Hubona et al. [29] performed tests in which the subjects had to complete a symmetric arrangement of 3D shapes by either repositioning or resizing an object under different circumstances. Among others, tested variables included again the presence of shadows as well as the number of light sources. It was found that in the repositioning test, shadows increased accuracy, but caused longer response times, while in the resizing test, no significant improvement in accuracy could be found when shadows were present. In both tests, the addition of a second light source and the resulting shadows decreased the performance of the subjects.

Langer and Bülthoff [41] conducted tests on the efficiency of surface shading under diffuse lighting in tasks related to the perception of depth and brightness. They found that the subjects performed generally as good when using a diffuse light source as when using a point light source positioned at different locations. Their results further support the hypothesis that humans benefit from illumination models in which low luminance is correlated with depth ("dark means deep") [58]. However, based on the very good performance of test subjects, they come to the conclusion that this rather simplistic model is only a rough approximation of the complex mechanism of human depth perception, which seems to employ more sophisticated procedures.

Hu et al. [28] performed two tests in which users had to judge the distance between a block of wood and a table in a virtual environment. In the first test, users had to place the block themselves as close to the table as possible without touching it. In the second test, users watched as the block was lowered towards the table up to a certain point, then had to estimate the remaining distance between the two objects. The authors did not only focus on shadows, but also included stereoscopic viewing as well as reflections on the table as visual cues. However, their results confirm that

the addition of shadows had a positive effect in all cases.

With respect to volume rendering, perceptual studies are rather scarce. Boucheny et al. [5] tested the depth perception of DVR images containing transparent objects. They found that the perception was poor when transparency was the only depth cue. It improved when the test subjects had to detect the direction of rotation of a cylinder, but only when carefully choosing luminance and opacity of the cylinder.

Chan et al. [9] focused on image saturation, shape enhancement and transparency as visual cues. They have presented a framework to automatically optimize transfer function parameters for a more effective user perception of DVR images. Wu et al. [93] have presented quantitative effectiveness measures for different volume rendering challenges such as the clarity of image contours or depth coherence. They allow the user to receive feedback about the overall effectiveness of a DVR image and provide the ability to further optimize rendering parameters.

Ropinski et al. [74] conducted a user study in which they evaluated their shadow volume propagation technique. They compared their approach to conventional gradient based shading, and found that their global lighting approximation significantly improved user speed and accuracy of depth perception.

6.3 Evaluated Volumetric Illumination Models

6.3.1 Selection Process

In recent years, several advanced illumination models for volumetric data have been proposed [67]. However, due to reasons of time and in order to keep the number of test images manageable for the user, we had to limit ourselves to a subset of these. Therefore, we have included mainly representatives from each group of DVR techniques, and those techniques that were proposed within a space of time of three years before the study was conducted. The only exception is half angle slicing which has been proposed as early as 2002 [37]. It was included as it is a popular, often-used model with a high citation count which produces the highest-frequency shadowing effects of the presented techniques. Another slicing based method, directional occlusion shading [78], was included because we were interested in the potential of this rather simple, efficient approach compared to more complex models, such as multidirectional occlusion shading [89]. This technique is based on directional occlusion shading, but has less restrictions concerning the position of the light source. It was included in our study as it allows the direct analysis of the influence of the light source position on the user performance when compared to directional occlusion shading.

Of the techniques that do not require slicing, spherical harmonic lighting [71] was included since it provides the possibility to test the effect of area light sources and more natural images on the user, in contrast to the hard shadows of techniques like half angle slicing. Shadow volume propagation [74] was added to the survey since it is one of the few advanced shading models that are slicing independent while not requiring a noticeable preprocessing. Dynamic ambient occlusion [75] was chosen for our tests since it belongs to a group of techniques that produce very subtle and subdued lighting effects, which may not be prominent enough to have a notable effect on user performance. Finally, Phong lighting [63, 42] which has long been established as a part of the DVR pipeline was included in the user study since it serves as the default technique which is compared to more sophisticated approaches.

We did not include unshaded DVR images in the study. While this case has its own benefits as it allows intensity quantification, our focus was on the improvement of perception of depth and size due to volume illumination.

The tested techniques are explained in more detail in Chapter 3. For a summary of the features of these models, refer to Table 6.1. While the list of used techniques is not exhaustive, we compiled it with the intention to present a cross sectional study that includes a balanced selection of state-of-the-art volume illumination models. An overview of the omitted techniques, which are also well worth mentioning, is available in Section 3.5.

6.3.2 Comparing Techniques

As seen in Chapter 3 and in Table 6.1, some of the tested techniques are fundamentally different from others, which makes some considerations about a fair comparison between them necessary. As mentioned before, three of the tested techniques were based on texture slicing [7]. All techniques not based on slicing were implemented using ray casting [65]. To be able to compare these two different rendering paradigms, we adjusted the number of slices in the slice-based techniques and the sampling rate in the ray casting techniques so that a comparable image quality was reached.

Another basic difference in the introduced techniques is whether or not gradient information is incorporated in the illumination computation. Gradients can be helpful when considering the relation between the orientation of a surface contained in the dataset and the location of the light source. However, in noisy datasets (such as ultrasound datasets) the gradients may not be clearly defined and lead to artifacts and false lighting effects. Thus, the use of an illumination model not depending on gradients can be an advantage. For this reason, we avoided the use of gradients in all techniques if the option was available. For Phong lighting, we used an on-the-fly

Table 6.1: A summary of the features of the surveyed illumination models.

Model	Rendering paradigm	Uses gradients	Light source location	Preprocessing required	Simulated global lighting phenomena	Whole dataset affects lighting	Shadow frequency
Half angle slicing	slice-based	no	no constraints	no	shadows, scattering	yes	very high
Directional occlusion shading	slice-based	no	headlight only	no	shadows	yes	low
Multidir. occlusion shading	slice-based	no	user hemisphere	no	shadows	yes	low
Dynamic ambient occlusion	independent	optional	no constraints	long, but only once per dataset	shadows, color bleeding	only voxel neighbourhood	low
Shadow volume propagation	independent	optional	no constraints	yes, but hardly noticeable	shadows, scattering	yes	high
Spherical harmonic lighting	independent	optional	freely rotatable area light source	for every TF change, noticeable	shadows, color bleeding, scattering	yes	very low
Phong lighting	independent	yes	no constraints	no	none	no	-

central-difference approach to compute gradients.

The introduced illumination models also vary in the quality of the produced shadows. Usually, the terms high frequency and low frequency are used to categorize illumination model features such as shadows. When using a high frequency illumination model, even small opaque features produce clearly defined, sharp shadows which are cast over the whole extent of the scene. Low frequency shadows are softer and more subtle, but may appear more natural than high frequency shadows due to the inclusion of penumbral regions. Researchers concerned with the influence of shadows have also differed between self-shadowing of objects onto themselves and cast shadows of objects onto other objects [95]. We neglected this difference since the division of a scene into objects cannot be applied in volume rendering, as a volume dataset represents a single object as well as the whole scene.

While most of the techniques approximating global lighting focus on shadowing, some of them are able to produce additional visual cues such as scattering effects. However, as shadowing is the only feature which all the tested advanced illumination models have in common, our main focus was on the influence of shadows as a visual cue. Furthermore, a separate evaluation of different lighting phenomena would have introduced too much complexity to complete the study effectively.

6.4 User Study

The study's goal was to test the influence of the techniques on the ability of an average layman user to deal with the challenge of understanding volume rendered images. We were also interested in the potential of the different models under varying surrounding conditions and monitor setups which better resemble real-world working space conditions. To do so, it was important to include as many users as possible in order to reach a representative subset of the population, as well as to perform the study within their everyday working environments. Therefore, we decided to realize the testing procedure by using a Java applet which users could start in an internet browser on their office or home computer. The applet started with a short questionnaire in which participants were asked for some information about themselves, including age, gender, profession and quality of eyesight. This was followed by the four tasks. Each of them consisted of a screen with instructions, followed by a first easy to solve training problem and a subsequent series of between 21 and 42 images, depending on the task. After completion, the results were transmitted back online. No medical or other expert knowledge was required, as the exercises were kept simple. We received a total number of 61 test results. Six of them were discarded as the participants had either not solved the training problems correctly or had needed a large amount of time to complete the session. The remaining 55 participants had diverse backgrounds, with their age ranging from 17 to 62 years with an average of 27.3 years. 56% of the participants were male. All had normal or corrected to normal vision, with 53% wearing glasses or contact lenses. Most of the users had limited 3D graphics experience or were complete laymen.

The applet used a within-subjects design as all participants took part in all four tasks. The users did not receive any feedback on their performance during or after the trials. The images for every task were presented in random order to prevent an ordering bias. They were generated using CT, MR, ultrasound and synthetic datasets which were illuminated using the different illumination models. Each dataset was used two times at most per task to allow for a greater variety of data. If a dataset appeared twice in one of the first three tasks which were aimed at depth and size perception, the lighting model, transfer function, camera position and light source position were changed between images to prevent a learning effect. All images were rendered using perspective projection. The transfer function specifications ranged from more transparent setups to surface-like representations, in order to capture a wide range of possible images. Some of the datasets contained entities which layman users could recognize, such as scans of body parts. However, we also included some datasets not familiar to layman users to increase diversity, such as scans of an

aneurysm or a mouse heart.

Except for spherical harmonic lighting for which we used a circular bright area light source, a white point light source was applied in all trials. For each image, a light source position in the front-facing hemisphere from the point of view of the observer was chosen. This allowed us to keep the position of the light source between techniques as consistent as the illumination models restrictions allow, while also complying with directional occlusion shading where the light position is not a changeable parameter (see Section 6.3). The participants were only informed about the fact that different illumination models were used. Detailed information about the illumination models themselves was not given. The users were advised to give their answers as quickly as possible and not to think too long about the solution. One run of the whole session typically lasted about 12 minutes.

After the tests had been completed, the average performance per technique was calculated for every task and test subject. We investigated connections between illumination model, correctness of the users' answers and the time the users took to solve each trial. The results were first evaluated using a one-way analysis of variance (ANOVA) to reject the null hypothesis that all correctness means were equal between techniques. If a confidence level of $p = 0.05$ was reached, a Bonferroni post-hoc test with the same acceptance level was performed to reveal differences between the individual techniques. This test was chosen as it compensates for error inflation caused by the rather large amount of pairwise comparisons between the seven lighting models. To test for a linear correlation between correctness and time, we used Pearson's r statistic. We will continue with a discussion of each of the four tasks as well as their respective outcomes, before drawing conclusions.

6.4.1 Relative Depth Perception

Material and Methods

For this task, we prepared 42 volume rendered images which were generated with the different illumination models. Each illumination model appeared 6 times. In each image, two circular markers were added which were located somewhere on the image region covered by the dataset. The positions of the markers were selected carefully for each used dataset and camera position to keep the difficulty level as consistent as possible between techniques. The users' task was to select with the mouse the marker which highlighted the region of the dataset that seemed closer to them along the viewing axis (see Fig. 6.2). Measured variables were the correctness of the choice as well as the time the users took to process each trial.

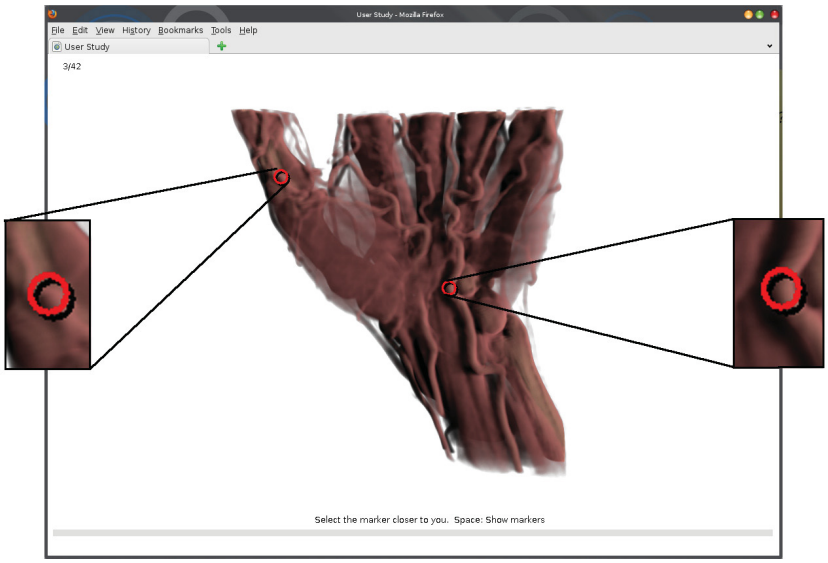


Figure 6.2: The evaluation applet used for the study during the relative depth perception task. The user had to select the marker covering the structure closer to the camera.

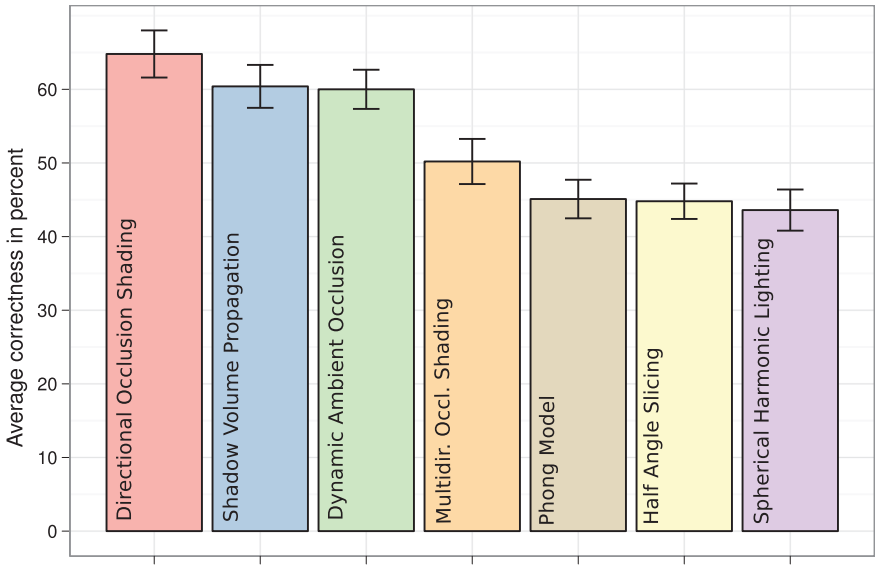


Figure 6.3: Relative depth perception results with standard error. The bars indicate the average percentage of correctly selected markers per participant when the respective technique was used.

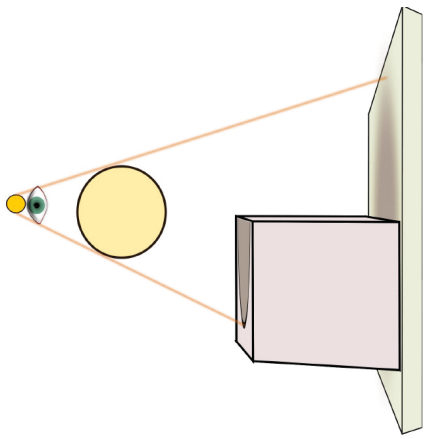


Figure 6.4: When using directional occlusion shading, the blurriness and the extent of shadows indicates relative depth, which helps to distinguish between dataset features.

Results

The ANOVA test showed a significant difference in the average correctness for the different techniques ($F(6, 378) = 9.809, p < 0.001$). The Bonferroni post-hoc test showed that directional occlusion shading with an average correctness of 64.8% performed significantly better that multidirectional occlusion shading (50.2%), Phong lighting (45.1%), half angle slicing (44.8%) and spherical harmonic lighting (43.6%). Shadow volume propagation (60.3%) and dynamic ambient occlusion (60.0%) performed significantly better than Phong lighting, half angle slicing and spherical harmonic lighting (see Fig. 6.3). There were no significant differences between techniques with respect to the elapsed testing time ($F(6, 378) = 0.512, p = 0.799$). Furthermore, there was no clear relation between time and correctness of the answers (Pearson's $r = -0.32$). Users took an average of 3.0 seconds for each trial.

Discussion

In general, the performance of users in this task was rather poor, as even with the technique that had the best result, over one third of the trials was solved falsely. This may be due to the fact that most users were not familiar with volume rendering or the presented style of data. The relatively good performance of directional occlusion shading may indicate that rather simple shadows which introduce a natural front-to-back hierarchy to dataset features can be beneficial when one has to distinguish the depth of different features in a volume rendered scene. This hierarchy effect may be due to the fact that the blurriness as well as the extent of a shadow from directional

occlusion shading clearly depict the relation of the fore- and background object to the viewer. In Fig. 6.4, a headlight causes an object to cast a shadow onto a box as well as a farther wall. Due to the shadow, the observer is able to distinguish the depth of the box surface from the depth of the wall as the shadow on the wall is bigger and softer. A similar effect is also exploited by the depth darkening technique first described by Luft et al. [50] who use blurring of a depth buffer. Thus, both techniques comply with the "dark means deep" paradigm [58]. The relatively weak performance of multidirectional occlusion shading (of which directional occlusion shading can be considered a subset) could mean that, in contrast to what is stated by Šoltészová et al. [89], the possibility to freely place the light source is not always an advantage. The separation of light source position and point of view of the observer leads to an increasing amount of visible shadows which might confuse users rather than help them, especially when the features that the user has to compare are not located next to each other.

Spherical harmonic lighting and half angle slicing had the worst performance, although they differ strongly in the frequency of the produced effects. This may indicate that the softness of shadows does not directly influence user performance in this task. Therefore, we performed an additional ANOVA test in which we compared the average correctness per user for soft shadow versus hard shadow techniques, without including the Phong lighting model. The test did not yield a result at a statistically significant level ($F(1, 108) = 1.107, p = 0.295$). This may indeed indicate that shadow frequency is not as important for this task as other illumination attributes (such as shadow direction).

6.4.2 Absolute Depth Perception

Material and Methods

This task subjected users to 42 volume rendered images, with every technique appearing 6 times. Each trial displayed the image with a single circular marker along with a horizontal slider at the bottom (see Fig. 6.5). Again, each marker was positioned with caution depending on dataset and camera position to allow for a consistently fair challenge. The users were asked to approximate the absolute depth of the marked region on the slider in percent, in relation to the maximum distance between the closest and the farthest point on the dataset from their point of view. After making their choice, the users had to confirm their selection by clicking the OK button. We measured the error between their choice and the actual absolute depth of the marked region as well as the time the users took to complete each trial.

Results

The ANOVA test revealed a significant difference in the average discrepancy of the users' guesses ($F(6,378) = 8.819, p < 0.001$). The Bonferroni post-hoc test showed that half angle slicing (18.5%), shadow volume propagation (20.3%), directional occlusion shading (21.0%) and multidirectional occlusion shading (21.0%) performed significantly better than Phong lighting (25.7%) and dynamic ambient occlusion (26.0%). Additionally, spherical harmonic lighting at 21.8% average discrepancy performed significantly better than dynamic ambient occlusion (see Fig. 6.6). No significant differences in time were measured between techniques ($F(6,378) = 0.415, p = 0.869$). There was no clear relation between time and correctness of the answers (Pearson's $r = 0.109$). Users took an average of 6.7 seconds for each trial.

Discussion

The results showed hardly any differences between the advanced techniques. The presence of global shadows seemed to help absolute depth perception, as Phong lighting and dynamic ambient occlusion had the worst performance. The presence of shadows on a bounding shadow receiver (such as the walls of the synthetic dataset in Fig. 6.7) may have helped subjects to better judge depth in this task. This resource is missing when using dynamic ambient occlusion, as this technique does not capture occlusion from objects which are not in the neighborhood. Such an effect may be of advantage when working with CT data, as they potentially contain less noise and thus produce less, but better defined shadows. Therefore, we evaluated the results for this modality only. The ANOVA test revealed a significant difference in means of correctness ($F(6,378) = 11.397, p < 0.001$), but not of elapsed time ($F(6,378) = 0.272, p = 0.95$). The post-hoc test showed that the correctness for half angle slicing was significantly better than for spherical harmonic lighting, directional occlusion shading, Phong lighting and dynamic ambient occlusion for this modality. The other techniques did not significantly improve their performance compared to their results for all modalities. This tendency towards a technique with very hard shadows may be explained by the fact that these shadows are cast across the whole extent of the scene. They may have the effect of a natural ruler when cast onto a homogeneous region of the dataset, which are more often contained in high quality datasets such as CT. Wanger [90] reported no significant differences between soft and hard shadows for an absolute depth perception task using non-volume rendered images. To compare our results to his, we performed the previous additional ANOVA test in which we compared the average discrepancy per user for soft shadow versus hard shadow techniques, without including the Phong illumination model. The result

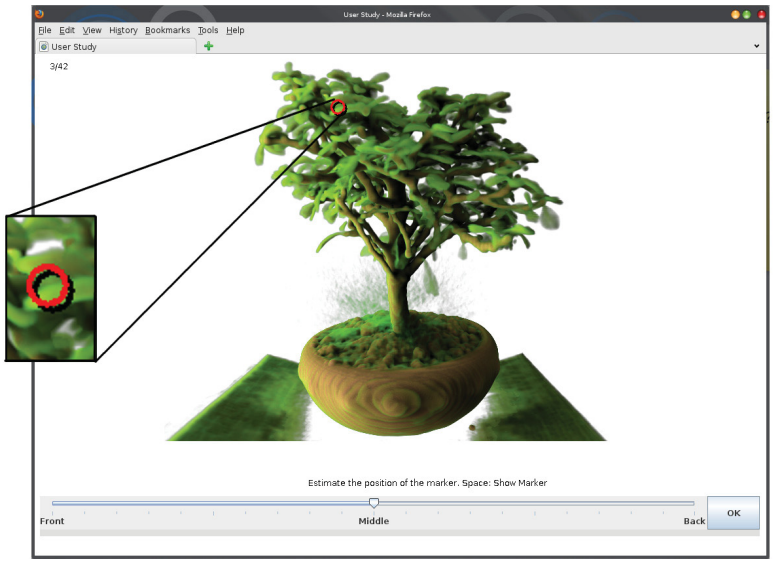


Figure 6.5: The evaluation applet used for the study during the absolute depth perception task. The user had to estimate the depth of the marked structure in relation to the total extent of the scene.

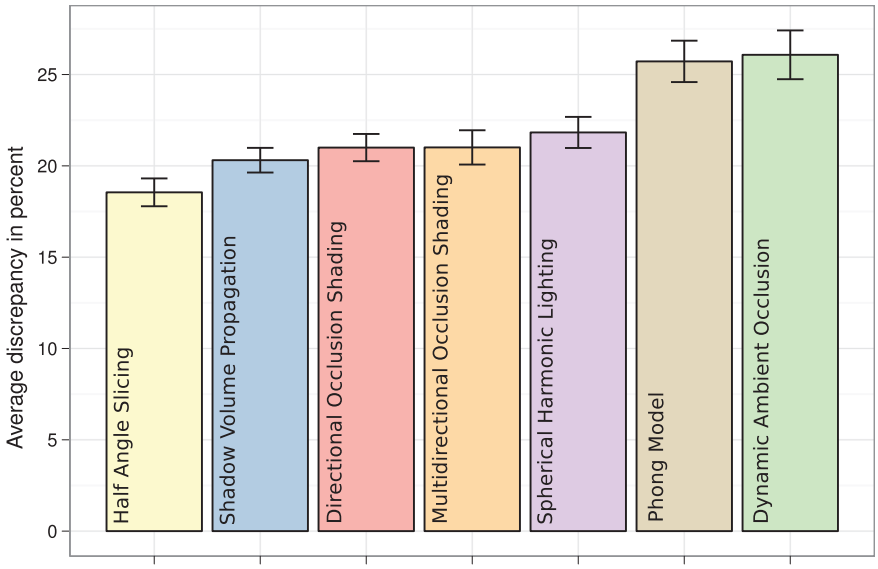


Figure 6.6: Absolute depth perception results with standard error. The bars indicate the average discrepancy from the true depth of the marker per participant when the respective technique was used.

showed that there is a small, but significant difference in favor of hard shadow techniques ($F(1,108) = 9.403, p = 0.003$), with an average discrepancy of 19.4% for hard shadows versus 21.9% for soft shadows. This may indicate that (at least for volume rendered images), hard shadows provide a better frame of reference when judging the absolute depth of features.

6.4.3 Relative Size Perception

Material and Methods

In this task, 21 volume rendered images were shown. Every illumination model appeared 3 times. The participants had to sort features contained in the datasets by volume size, such as the different bones in a CT scan of a human hand. The users had to sort the objects by size with the help of drop-down combo-boxes at the bottom of the image, which were labeled with the letter associated with the object. They had to confirm their choice with an OK button before proceeding. We selected and segmented different features which were labeled in the images with letters and highlighted with different colors so the users knew which parts of the dataset they were supposed to compare (see Fig. 6.7). Ultrasound datasets were not used for this task as they proved to contain too much data noise to achieve a meaningful segmentation. Measured values were again the correctness of the ranking given by the users, as well as the time they took for each trial. A trial was evaluated as correct only if all objects were assigned the correct size rank.

Results

Again, the differences in average correctness per user were statistically significant between illumination models according to the ANOVA test ($F(6,378) = 33.88, p < 0.001$). The post-hoc tests revealed that directional occlusion shading performed significantly better than all other techniques with an average correctness of 86.4%. Shadow volume propagation (63.2%) performed significantly better than multidirectional occlusion shading and dynamic ambient occlusion (both 48.4%), half angle slicing (42.7%) and Phong lighting (33.3%). Spherical harmonic lighting (56.9%) performed significantly better than half angle slicing and Phong lighting, while multidirectional occlusion shading and dynamic ambient occlusion only delivered significantly better results than Phong lighting (see Fig. 6.8). Furthermore, a significant difference in timing could be measured ($F(6,378) = 4.332, p < 0.001$). The Bonferroni post-hoc test showed that on average, trials with directional occlusion shading were answered significantly faster at 4.8 seconds than trials with dynamic ambient occlusion (8.5

seconds) and shadow volume propagation (9.2 seconds). Additionally, spherical harmonic lighting trials were solved significantly faster at an average of 5.6 seconds than shadow volume propagation trials. There was no clear relation between time and correctness of the answers (Pearson's $r = -0.04$). Users took an average of 6.4 seconds for each trial.

Discussion

The results of this task are similar to those of the relative depth perception task. The use of directional occlusion shading leads to a success rate of over 85%, followed with some distance by shadow volume propagation. This is in line with previous research which indicates that comprehension of depth is an important prerequisite for comprehension of size [17], as objects farther away from the viewer appear smaller while having a potentially larger volume. As directional occlusion shading was the top-performing technique for relative depth perception, it may have helped users to avoid false judgments based on perspective. Furthermore, as the light direction is congruent with the viewing direction using this technique, a shadow cast by an object close to the viewer was in most cases projected onto a shadow receiver, since the camera was pointed at the center of the dataset in almost all of our trials. Less information was lost since the shadows did not "leave" the dataset.

Half angle slicing and multidirectional occlusion shading achieved a similar ranking in the first task. These techniques do not seem to yield a considerable difference compared to the simple Phong illumination model in this task. Spherical harmonic lighting, however, yields considerably better results than in the first task as it improves user performance compared to the local lighting model. While very soft shadows seem to be confusing rather than helpful when pin-pointing a single point of depth in a scene, they could be better suited to approximate size. This may be due to the fact that single small features, which were also often marked in the first task, have almost nonexistent shadows when illuminated by a low-frequency illumination model, while their projection is still clearly visible with high-frequency techniques as half angle slicing. The size of the relatively large objects in this task, however, can be possibly estimated rather well even with low frequency techniques because they produce natural, volumetric soft shadows to which users can relate. Since shape and volume of an object are closely related, this conclusion may seem to be in conflict with the results of Wanger [90], who reported that diffuse shadowing was detrimental to the subjects' ability to perceive the shape of an object. However, we did not ask the users to assign a shape to a single object, but rather to find a ranking in relative size between several objects. This seems to be a different process in which individual

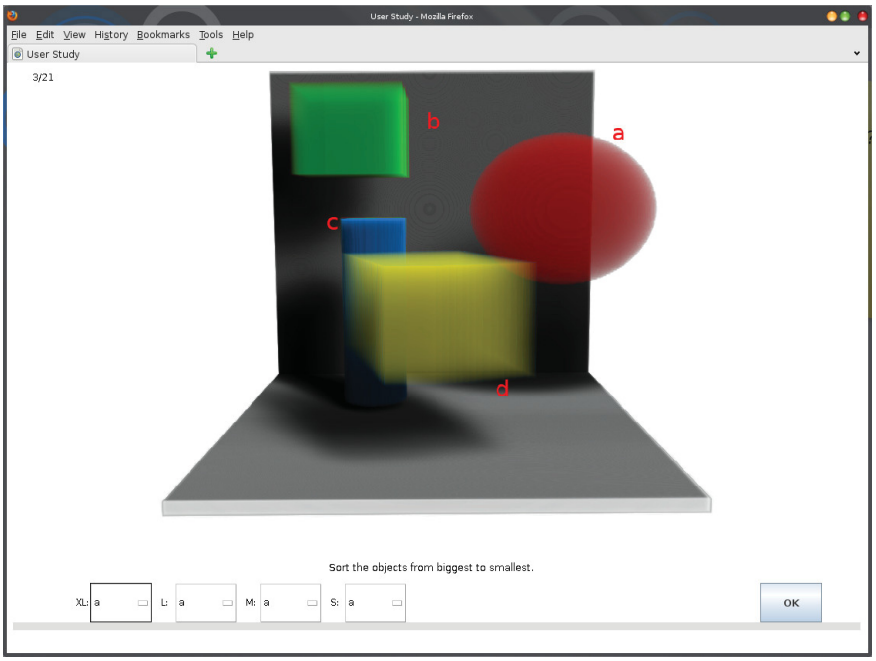


Figure 6.7: The evaluation applet used for the study during the relative size perception task. The user had to sort the labeled objects by size, from biggest to smallest.

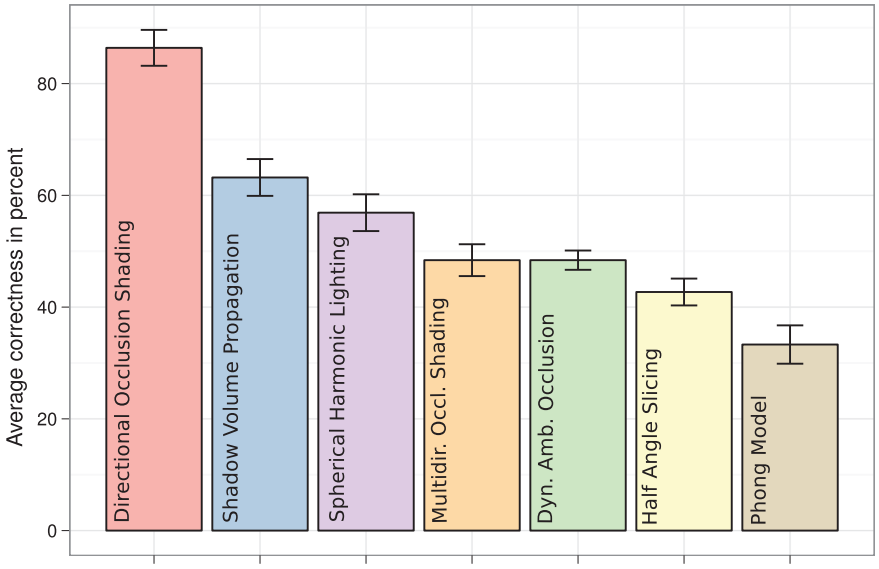


Figure 6.8: Relative size perception results with standard error. The bars indicate the average percentage of correctly ranked objects per participant when the respective technique was used.

shape recognition may not be the most important tool, but rather the shadowing of the scene as a whole.

Dynamic ambient occlusion, another technique with rather low-frequency effects, failed to reach a similar success rate as in the previously described relative depth task. This is probably due to the fact that objects which are located relatively far from surrounding features do not produce shadows at all with this technique, since they are not included in the local voxel neighborhood considered during the preprocessing (see Section 6.3). Therefore, low-frequency lighting techniques which still include the whole dataset in the occlusion calculation seem to be a good choice in this context. This is also supported by the fact that directional occlusion shading and spherical harmonic lighting trials were answered faster than trials involving other techniques, while retaining a relatively high rate of correctness.

As the results from the previous absolute depth recognition task became less ambiguous when considering only images from high quality datasets, we again conducted an analysis of images from CT data only. There was a significant difference in the average correctness ($F(6, 378) = 24.615, p < 0.001$). The success rate was generally higher for this modality than when considering all modalities together, except for Phong lighting with a success rate of only 7.3%. The post-hoc test showed that even while considering only high quality CT data, directional occlusion shading at a success rate of 83.6% still performed significantly better than dynamic ambient occlusion, half angle slicing and multidirectional occlusion shading, while spherical harmonic lighting performed significantly better than dynamic ambient occlusion. Additionally, all techniques performed significantly better than Phong lighting. The ANOVA test for timing results was also significant for CT datasets ($F(6, 378) = 3.357, p = 0.003$). Directional occlusion shading performed significantly better in this regard at an average time of 4.7 seconds than dynamic ambient occlusion (8.5 seconds), Phong lighting (8.6 seconds) and multidirectional occlusion shading (8.7 seconds). This result emphasizes the potential of directional occlusion shading and of advanced lighting in general for this task.

6.4.4 Subjective Evaluation

Material and Methods

In the final task, users were presented with 26 pairs of images which were each rendered with one volume dataset, but using two different illumination models. The image pairs differed only in the used illumination model (see Fig. 6.9), and the users were asked to select that image which they subjectively liked better. All 21 possible combinations of illumination models were included in the series. We measured the

percentage of times each illumination model was preferred.

Results

The ANOVA test showed a significant difference in illumination model preference ($F(6,378) = 61.655, p < 0.001$). The Bonferroni post-hoc test revealed that Phong lighting was favored significantly more often by users than all other techniques at a rate of 81.2%, followed by half angle slicing at 68.0% being favored significantly more often than the remaining approaches. The middle ranks were occupied by shadow volume propagation (54.0%) and spherical harmonic lighting (52.9%), which performed significantly better than the subjacent techniques. Directional occlusion shading (37.9%) and dynamic ambient occlusion (37.3%) were preferred significantly more often than multidirectional occlusion shading at a preference rate of 21.8% (see Fig. 6.10).

Discussion

It is rather surprising that the Phong illumination model performed so well in this task, compared to the previous results. There are no shadows present, and the resulting images have a more artificial appearance than those rendered with the other techniques. On the other hand, spherical harmonic lighting, which could be considered the most realistic of the used illumination models due to a natural area light source and diffuse shadows, delivered a rather average result. It is worth noting that we did not specify what quality of the image the users were supposed to judge (such as "Which image appears more realistic to you?"), but asked only which image appealed more to them aesthetically. Phong images are generally brighter and more colorful due to the missing shadows and have a more illustrative quality. This may have a positive effect on the user if such an image is placed next to one including shadows. The participants also seemed to prefer high-frequency hard shadows over low-frequency techniques, with the half angle slicing and shadow volume propagation illumination models reaching considerably better results than the other advanced illumination models. This was confirmed by an additional ANOVA test of soft shadowing versus hard shadowing techniques without considering the cases involving the Phong model, which showed a significant difference in preference rates ($F(1,108) = 101.14, p < 0.001$). Altogether, techniques with hard shadows were preferred in 65.6% of those trials, while techniques with soft shadows had a preference rate of only 34.3%. Hard shadows with no penumbral region do not appear often in daily life since a point light source with an otherwise dark environment would be necessary to produce them. This might indicate a general tendency of users

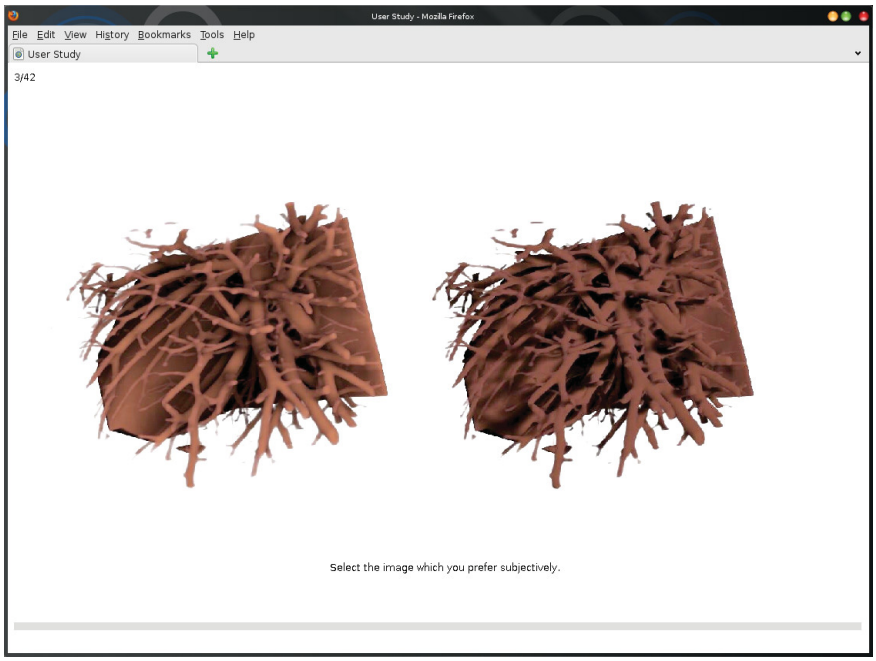


Figure 6.9: The evaluation applet used for the study during the subjective preference task. The user had to select the image which he subjectively preferred.

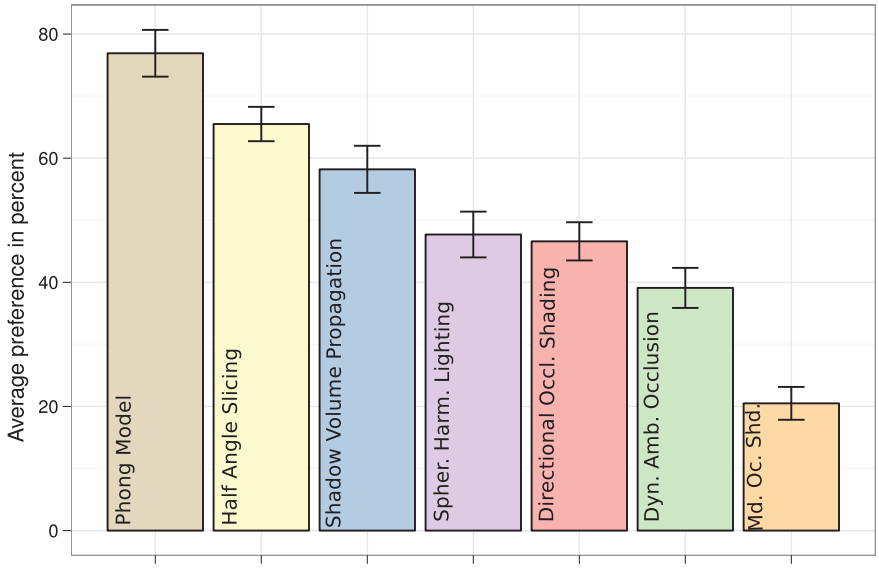


Figure 6.10: Subjective preference results with standard error. The bars indicate the average percentage of times per participant in which the respective technique was preferred over the other shown technique.

to prefer rather artificial, illustrative representations of a scene over more natural, realistic approximations.

6.5 Results and Discussion

Our user study shows that depth and size perception in volume rendering is significantly improved by exploiting advanced lighting approximations, since the local technique (Phong) delivers worse results than almost all global models in all related tasks. This result is backed up by previous studies with a similar outcome in polygonal rendering [91, 29]. Therefore, it seems to be preferable to use an illumination model which provides additional cues such as shadows when conducting volume data analysis. Of course, an interesting result is the discrepancy between effectiveness and appeal of the illumination models, as the Phong illumination model was most often favored by the test subjects. Apparently, there is an inverse relationship between realism and preferableness of an illumination model, similar to the uncanny valley phenomenon. It may be caused by the presence of too much visual information in the images with shadows, in contrast to the relative simplicity of the Phong model images. This effect may also be connected to the familiarity of rendered images that were generated with Phong lighting, since it can be considered as a long-established standard used in many computer-based scenarios. This could be exploited when not analysis, but rather presentation of data is important. For instance, the Phong illumination model (or a similar basic shading method) could be used to illustrate medical information during a doctor / patient communication.

The choice of the applied advanced illumination model should depend on the required task as well as the quality of the volumetric data. With respect to relative depth recognition, it seems that positioning the light source at the point of view of the user is beneficial in the tested cases. Directional occlusion shading seems to help the user best in distinguishing the depth of dataset features from another, because the muted shadowing effects produced by this technique introduce a hierarchical order. It seems that cast shadows from the shadow volume propagation, half angle slicing and multidirectional occlusion shading techniques can in same cases be confusing rather than helpful if the shadow direction deviates from the viewing axis. This is especially interesting as medical illustrators prefer a placement of the light source at the top left corner of the field of vision to enhance image perception [89]. Our results are generally not in line with this assumption, which becomes clear in particular when comparing the performance of directional occlusion shading versus multidirectional occlusion shading. Here, the difference lies not in the quality of the shadows, but in their direction and amount. Thus, while shadows are beneficial for relative depth

perception in general, our findings indicate that one should use them in a rather subtle manner when confronted with this kind of task to reach the best results. The relatively good performance of dynamic ambient occlusion in the first task also supports this claim. The shadows of this technique do not depend on a light source position at all, but only reflect the local scene geometry around each voxel. Of course, in all of the presented techniques the light source could be placed at (or around) the point of view of the observer to prevent excessive shadowing, which may be a point of interest for future studies. Using directional occlusion shading to achieve a better relative feature perception has the additional advantage that this technique is independent of gradients and does not require a noticeable preprocessing. For these reasons, we recommend directional occlusion shading for tasks related to relative depth perception in combination with all modalities.

In turn, if the classification of the absolute depth of a dataset feature is required, our results suggest that an illumination model should be preferred which produces shadows that are not aligned with the viewing axis. Half angle slicing, a global lighting approximation with hard shadows, may have reached better results than others, since it provides a frame of reference by casting clearly outlined shadows onto other dataset features. This effect, however, is not as noticeable as the apparent positive influence of a head light on feature ranking tasks. It seems to strongly depend on the quality of the given data, as the test results were less ambiguous when considering CT data only. The addition of a reference ruler indicating the total extent of a dataset in addition to an advanced lighting model could improve user performance. In the absolute depth task of Wanger [90] in which he included a ruler, test subjects reached far better results than in our version of this task. However, considering the higher complexity of volume rendered images used by us compared to Wanger's images of simple non-volumetric objects, the potential benefit of this device is at least doubtful. The tests should be repeated in the future to determine if such a tool would improve user accuracy and yield a clearer result even for noisy data. Until this has been investigated, we recommend half angle slicing for CT and other clearly structured volumetric data, as it reached the best test result and requires no long preprocessing.

Finally, the recognition of relative feature size seems to benefit from soft lighting effects as well as shadows which are cast over the whole scene and not just towards neighborhood features. This is indicated by the good performance of shadow volume propagation and spherical harmonic lighting as well as the very good performance of directional occlusion shading, which combines both attributes. In addition, the recognition of relative size is a more complex procedure which takes more time to complete than the other experiments. This may have led to a more distinct difference

in timing results than in the other tasks. Directional occlusion shading also performed well in this regard, as users needed significantly less time to complete the task with this technique compared to several other models. Therefore, we recommend directional occlusion shading for all modalities for this kind of task, since it combines the best testing results for the quality and the timing of the participants' answers.

Based on our results, we recommend the following guidelines for the application of volume illumination techniques to improve image comprehension:

- In general, keep lighting as subtle as possible - the principle "less is more" seems to apply.
- A lighting model that does not dominate the appearance of the scene should be used as a default. The image should be enhanced, not distorted.
- To enable the user to understand the relation between all structures in a scene, the lighting model should ideally include the whole dataset when producing lighting effects.
- We recommend an initial light source position at the point of view of the observer, as the produced visual cues are less likely to be obtrusive. If this default configuration does not provide enough information, an optimal model should enable the user to produce more noticeable effects by changing parameters like light source position and shadow frequency.

Directional occlusion shading complies with most of these guidelines and could be used as a starting point. We believe that multidirectional occlusion shading is a valuable first step to enhance this technique. Other properties such as the frequency of effects could be added as a changeable parameter in the future to allow for the adjustment of lighting properties according to the given circumstances.

6.6 Conclusions and Future Work

We have presented a study in which we tested and compared the effectiveness of standard as well as advanced illumination models for DVR. Based on the results, we derived guidelines on the choice of an illumination model. Our survey has shown that global illumination models make a difference when it is necessary to assess depth or size in images. Although the time the users took to complete the individual trials was hardly influenced by the use of a particular model, the level of correctness improved significantly. Furthermore, the position of the light source seems to play a

pivotal role. Depending on the situation, the direction of shadows can significantly influence the quality of depth perception.

We believe directional occlusion shading performed so well because the users were intuitively aware of the exact light source position at the point of observation, whereas an arbitrarily placed light source provides an additional factor of uncertainty. While modern hardware allows the inclusion of most of the tested illumination models in real-time, we think that there is a strong indication that at least a relatively simple illumination model as the directional occlusion shading should be employed in the professional analysis of scientific volume data.

In the future, additional studies are needed to further clarify the influence of different illumination factors. In the presented study, the light source position was not directly part of the investigated parameters. A more careful research of the influence of light source positioning should be conducted, as Hubona et al. [29] have also suggested. Finding out if a head light is beneficial in conjunction with all illumination models would be especially important. Another point of interest would be how a moving light source (resulting in moving shadows) could improve the perception of a volume rendering scene over static lighting. Research indicates that mobile shadows can be a much stronger visual cue [35]. Other visual cues such as interreflections, light scattering or color bleeding should be also included in future testing. If possible, each cue should be tested individually to find out exactly which ones should be part of an optimal lighting model. Finally, a task which tests the influence of volume illumination models on shape recognition could be an interesting addition to a prospective study, maybe also when dealing with dynamic data.

In our study only laymen with no familiarity with the presented datasets were asked to participate. Thus, we were able to minimize the influence of previous knowledge and expectations. Domain experts could not only benefit from visual cues, but also their experience and training when solving different tasks. As a consequence, a future domain expert study would be of interest, as it could reveal completely different results. This study should include task-driven trials to research the effectiveness of volume illumination models in real-life tasks, such as the identification of aneurysms in medical volume datasets.

As far as the development of new illumination models is concerned, we find that there is still room for improvement. Even untrained users should be able to reach better results than those demonstrated in our tests, as the respective error rates were rather high. An optimized illumination model as suggested in Section 6.5 would be as unambiguous and subtle as directional occlusion shading as the default state. This could be combined with the possibility to move the light source away from the observer's point of view temporarily to produce high frequency cast shadows, which

might help to relate the absolute depth of features to the whole extent of the dataset. Multidirectional occlusion shading might be a first valuable step into this direction. However, an additional study in which user interaction is included would have to be carried out to test this combination.

Chapter 7

Interactive Comparative Visualization of Multimodal Brain Tumor Segmentation Data

We present a visualization system for the analysis of segmentation data of brain tumors. As more and more algorithms for the efficient segmentation of brain tumors have become available, our system is designed to allow researchers and doctors a further investigation of the segmented tumor data beyond a quantitative assessment of size. This includes the efficient visual analysis of the shape and relative position of the different, often overlapping segmented data modalities, using high quality 3D renderings of the data. Furthermore, our system includes visualization methods to compare tumor segmentation volumes acquired at various points of time, which helps the user to explore changes in shape and size before and after treatment. We employ different linked 3D and 2D visualization methods, as well as specialized interactive diagrams which allow the user to quickly navigate and analyze overlapping tumor regions. All methods are assembled and linked in a multi-view framework. With the help of our system, the user is able to efficiently compare the data acquired with different kinds of brain imaging modalities visually as well as quantitatively. In the future, this could help researchers to better understand and compare the results of established and new, experimental imaging methods.

7.1 Introduction

Gliomas are the most common primary tumors of the central nervous system with an incidence ranging between 6-8/100.000. Despite multi-modal therapeutic regimens (surgery, radiation, chemotherapy, anti-angiogenic therapy), the prognosis of high

grade gliomas (World Health Organization Grade IV) is still poor with median survival being 1-2 years. In the past years, two major research areas have supported increased understanding of glioma biology: first, detailed molecular-genetic analysis revealing a step-wise accumulation of disease-specific genetic changes with some of them having prognostic relevance; second, development of multi-modal molecular and structural imaging based on PET and MRI. A combination of MRI and PET imaging is being used as primary diagnostic tool for tumor localization, grading, and delineation from functionally important neuronal tissue; thereafter, imaging is being used to quantify response to therapy, time of tumor recurrence and differentiation of biologically active tumor tissue from radiation necrosis [81]. One of the critical parameters influencing the disease's progression and outcome is the extent of the tumor in 3-dimensional space. The true tumor extent can be measured by a combination of various MRI- and PET-based parameters as they reveal complementary information on the tumor. The combined assessment of the tumor volume as measured by MRI and PET has direct impact on surgery, radiation therapy and clinical outcome [64] [20]. Our framework aims to facilitate this assessment before and during treatment.

For many patients, a set of two MRI scans (T1- and T2-weighted) and one PET scan is produced as part of the diagnosis. The resulting datasets are often compared by overlaying 2D slices, or by comparing 2D slices side by side. However, in recent years the segmentation of three-dimensional regions of interest such as tumor tissue has been greatly simplified by the introduction of automatic segmentation techniques such as the approach presented by Corso et al. [11]. By separating voxels within a 3D dataset having a high probability of belonging to a tumor region from voxels representing healthy tissue, a binary mask dataset is generated which has non-zero values only where the image indicates tumor tissue. We call these regions *segmented tumor regions* (STRs). The system has been designed with the objective to subsequently facilitate the understanding of the spatial extent of the generated three-dimensional STRs in relation to each other. By exploiting comparative visualization strategies, the three different standard imaging modalities can be compared to each other simultaneously to allow a clinical characterization of the tumor. Areas where two or even three STRs intersect can be easily identified, which may indicate the center of the actual tumor tissue.

Besides an initial diagnosis, an optimal application is required to assess the progress of the tumor over time to track development of the tumor disease, discover recidivisms, make treatment decisions and carry out therapy such as surgery or radiotherapy. Therefore, our framework also aims to visualize the extension or the shrinking of different tumor parts between different stages of treatment to determine potential recidivisms and tumor parts that are inactive. Surrounding anatomical structures are

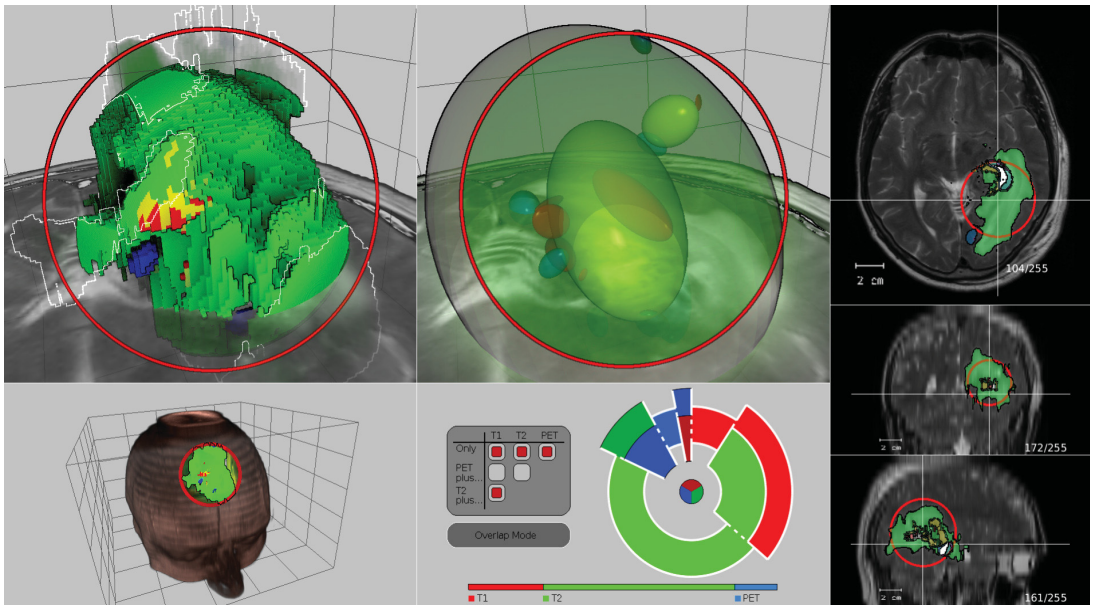


Figure 7.1: An overview of our framework. In the upper left, the tumor intersection closeup with the context view below. In the upper middle, the tumor ellipsoid diagram; in the lower middle, the volume overlap diagram. At the right, three linked 2D slice views showing the STRs and their overlaps, as well as the respective 2D projection of the tumor lense.

visualized to determine the direction in which the tumor is proliferating. It is also important that the position of the potential tumor volume can be easily located in the anatomical structure. Therefore, the introduced 3D visualization is linked with a traditional 2D slice representation.

This chapter is structured as follows: First, we will discuss previous work on multimodal brain imaging. We will then describe in detail the 3D and 2D visualization methods we chose. Afterwards, we will introduce two types of diagram that were developed specifically to ease the understanding of overlapping volumetric tumor regions. We will then briefly discuss some implementation details, before drawing a conclusion and describing possible future work on our system.

7.2 Related Work

In recent years, much work has been published concerning multimodal volume rendering of brain imaging data. However, most publications focus on pre-operation

planning, using context modalities such as fMRI to find optimal access paths instead of comparing the size and shape of multimodal segmentations of the same tumor for diagnosis and treatment planning. An early example of multimodal brain image rendering in 3D is given by Serra et al. [79], who simultaneously integrate CT, MRA and MRI modalities into a virtual workbench using a slicing approach. Beyer et al. [3] introduce a framework that supports planning of the optimal incision point by using a skull-peeling algorithm, as well as multimodal visualization of the underlying structures. Herghelegiu et al. [24] present a system for optimal planning of a brain tissue biopsy which helps to avoid hitting blood vessels or other critical areas. Jannin et al. [30] extract silhouettes of multimodal data to blend over a context volume in order to visualize multiple structures, which inspired the use of silhouettes and contours in this chapter as a 3D shading approach.

Rieder et al. [68] presented interaction methods to aid surgeons to simultaneously perceive multimodal data in order to help them with intervention planning. Rieder et al. [69] introduced a framework for blending multimodal MRI images which used specialized clipping and illumination to separate tumor tissue from surrounding brain tissue. Furthermore, Rieder et al. [70] introduced the tumor map to visualize the treatment success of liver tumors with percutaneous radiofrequency ablation. This map provides a pseudo-cylindrical representation of the tumor surface, which simplifies the visualization of the tumor structure.

Tietjen et al. [87] presented techniques to show the extent of multiple segmented objects within 2D slices by attaching vertical bars representing the length of each object next to a 2D slice of a context volume. Viola et al.[88] suppressed less important parts of the data with maximum importance projection, essentially mapping importance of the data to visibility. A similar approach is used for the tumor lense introduced in this chapter.

7.3 3D Visualization

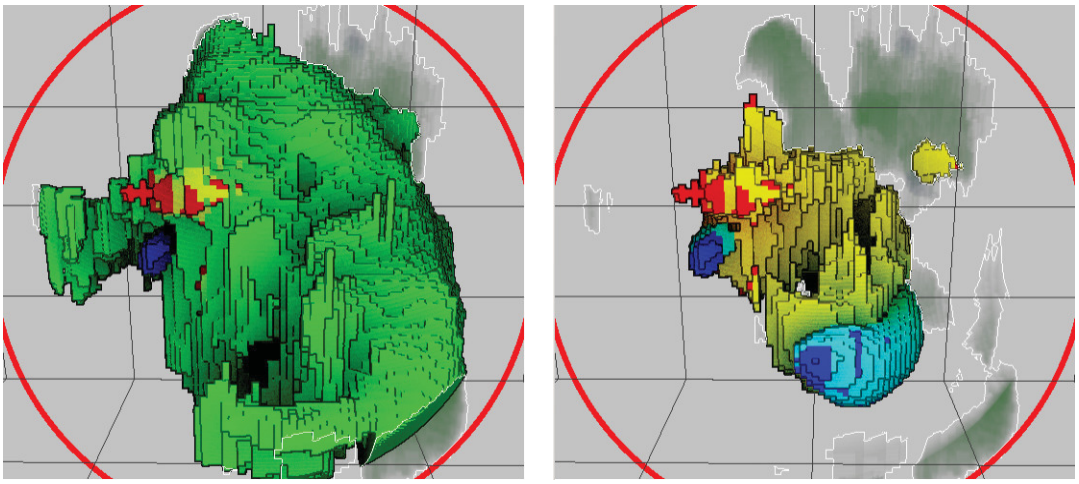
As mentioned in Section 1, the goal of this work is to visually analyze and compare segmentations obtained from different brain imaging modalities, as well as their changes before, during and after treatment. The challenge lies in providing the user with a spatially comprehensible image, as large sections of the different modalities will overlap and therefore occlude each other. This section gives an overview of the 3D rendering methods which we use to achieve this goal. Our central 3D visualization of the tumor uses a modified GPU ray casting approach, where rays traverse a volume dataset in which the co-registered STR masks are bitwise encoded as integer data, the so called *tumor index*. Thus, for instance, a single 8 bit dataset is able to hold

information about three different tumor masks at two different points in time. The color scheme used during ray casting when hitting an STR is based on the RGB color space, where each of the three STRs from the standard image modalities is represented by the respective color channel. The T1-weighted MRI STR is associated with the red channel, the T2-weighted MRI STR with the green channel and the STR segmented from the PET modality with the blue channel. For overlaps between the STRs, an area of intersection of the first and the second STR would be assigned the color yellow, whereas a region where only the third mask was active would be assigned the color blue. This basic color scheme was chosen instead of the usual assignment of color by transfer functions during ray casting as we want to visualize a set of binary masks and their intersections, rather than a distribution of intensity values.

7.3.1 Surface Shading and Illumination Model

To perceive the surface, shape and size of objects, illumination is a valuable tool in volume rendering [43], especially when having to distinguish overlapping objects. However, depending on the segmentation method used as well as on the image modality, the different STRs are often very irregular and frayed, making an intensity gradient-based shading approach difficult to apply. Instead, we chose a volume illustration approach to shading, in which depth-based edge detection is used to generate and display silhouettes and contours of each STR. First, a ray casting image is generated using nearest instead of linear or even more complex filtering methods. We chose this simple filtering model as we try to visualize the segmentation masks themselves and not a reference volume with the help of segmentation masks. As a result, the rendering resembles the original STR data more closely. On the rendered tumor surface, neighboring voxels whose distance from the viewer differ even only by the extent of a single voxel will have depth values which differ by a small, but significant amount due to the simple, coarse filtering. A subsequent depth-based edge detection on the ray casting image using depth-based line thickness brings out these small differences as subtle contours, resulting in a ”building brick effect” (see Fig. 7.2). Silhouettes are emphasized with black lines. Inner contours are kept more subtle, using very thin lines in a light gray color.

Additionally, screen space ambient occlusion as originally suggested by Mittring [56] is used to further illustrate differences in relative depth by displaying shadow-like halos around STR areas in the foreground. Additionally, the computed ambient occlusion factor is used to add a subtle color bleeding from neighboring pixels. Our implementation of this technique relies only on depth values and is



(a) (b)

Figure 7.2: The tumor intersection closeup. The region where only the T2-weighted MRI modality is active (green) occludes other segments and overlaps (a). By deselecting it (b), the underlying STRs and their overlaps can be made visible. In this case, relatively large areas are displayed where the T2-weighted MRI and the T1-weighted MRI modality (yellow) as well as the T2-weighted MR and the PET modality (turquoise) are active. Note that the contours and screen space ambient occlusion convey the shape of the regions well.

therefore suitable for the non-smooth surface structure of the STRs. Furthermore, this method delivers illumination effects similar to directional occlusion shading [78] which yielded the best overall testing results for depth and size perception in the user study presented in Chapter 6. Since directional occlusion relies on 3D texture slicing and our framework is based on ray casting, and due to the fact that no transparency is involved in the rendering of the foreground STRs, screen space ambient occlusion is an adequate replacement. The result of the combination of this form of ambient occlusion with contour and silhouette lines is an unobtrusive, yet effective shading model conveying the size, shape and relative depth of visible regions and surfaces when applied to the coarse STR surfaces (see Fig. 7.2).

7.3.2 Tumor Context View

The lower of the two 3D views (see Fig. 7.1, left column) displays a context volume together with the tumor data, generated using ray casting with a user-defined transfer function for the context volume. We call this view the *tumor context view* (TCV). In

order to always be able to see the tumor region, the ray casting algorithm has been modified for this view. Before rendering, the axis-aligned bounding box of the combined STRs is calculated and saved in a separate proxy geometry. If a ray is going to hit this tumor box, the region which the ray traverses within the context volume before it hits the box is discarded. Within the tumor box, only tumor data is rendered with full opacity while the opacity of the context volume is set to 0. If the ray has not hit tumor tissue on its way through the tumor box, the ray continues through the context volume, now with the regular opacity given by the transfer function. Thus, the tumor box is always visible showing the entire union of all STRs, independently of the rotation of the context volume (see Fig. 7.1, bottom left).

7.3.3 Tumor Intersection Closeup in Single Time Step Mode

In single time step mode, the upper 3D view displays the combined STRs of the three standard image modalities recorded at the same time in the so-called *tumor intersection closeup* (TIC). For this view, we use a two level ray casting process [21] which has been modified to deal with the issue of overlapping STRs. We distinguish two regions: an inner spherical region of interest which we call the *tumor lense*, initially located around the center of mass of the union of all STRs, and the outer context region. As a ray traverses the volume, we check whether or not the current sample point lies within the spherical tumor lense. If not, a low opacity is used to visually suppress regions that are currently not of interest. If the ray sample lies within the tumor lense and tumor tissue is hit, all previously composited color from the outer region is discarded in order to provide an unobscured view of the lense content. However, the depth of the first hit point of the outer region is always saved separately in order to provide contour lines of context STRs which would currently occlude the tumor lense. These outer contour lines are overlaid in the final image as a subtle hint that there are currently occluded areas (see Fig. 7.2). Then, the local tumor index is looked up. The user can control which STRs should be visible within the spherical region by enabling a set of modalities or modality intersections in the tumor overlap diagram (see Section 7.5). If the retrieved tumor index is part of the current user selection, a high opacity is used, otherwise the found STR voxel remains invisible.

In order to better navigate this view, the camera view vector is synchronized with the one used in the TCV. When rotating the tumor in TIC view, the camera orientation (but not position) is synchronized to the TCV camera. This enables the user to track from which direction relative to the patient's head the tumor closeup is seen. For a further visual connection, a red ring is drawn around the region of interest representing the radius the user has currently selected for the tumor lense in both

TCV and TIC (see Fig. 7.1). This ring can be used to control the radius of the lense by grabbing and dragging it with the mouse cursor. The center of the tumor lense can be changed by the user by dragging the mouse within the 3D view while pressing an additional modifying key. This also changes the trackball center and camera focus of the TIC which always follows the center of the tumor lense, keeping the navigation unambiguous. The user is able to display measurements within the TIC by drawing a ruler line from an arbitrary point in the rendering. The first hit points at start and end of the line are looked up. Then, the corresponding distance is converted to real world values and overlaid over the rendering next to the mouse cursor.

7.3.4 Tumor Surface Closeup in Consecutive Time Steps

Instead of focusing on different STRs recorded within the same timeframe, the user may also select the same STR recorded at different points of time t_1 and t_2. The so-called *tumor surface closeup* can be used to analyze regions in which the tumor has changed due to treatment and to monitor how each imaging method displays these changes. This visualization uses a two step process in order to identify regions in which the selected STR has shrunk, grown or remains unchanged. During this process, the distance between the surface voxels of STR_{t_1} and STR_{t_2} is determined and subsequently visualized.

First, the intersection $STR_{\cap}$ of STR_{t_1} and STR_{t_2} is calculated. We then use the technique introduced by Cornea et al. [10] to compute a set of points STR_{skel} within $STR_{\cap}$, representing its inner curve-skeleton. This technique is based upon computing a repulsive force field over the voxel grid and using topological characteristics of the resulting vector field, such as critical points and critical curves, to extract the skeleton. The skeleton voxels will then be used to determine the direction of change from each surface voxel of STR_{t_1} and STR_{t_2}. Since the STRs can have complex elongated forms including concavities, computing the change in the direction of the center of mass, for instance, may not be sufficient. Furthermore, due to the fuzzyness of the MRI data, we did not want to rely directly on intensity gradients to calculate the direction of surface change of the STRs. The skeleton provides inner reference points without using gradients, while observing the potentially complex shape of the STR.

Next, we find the set of outer surface voxels STR_o of the union of STR_{t_1} and STR_{t_2}, i.e., all voxels belonging to a surface of either STR_{t_1} or STR_{t_2} where the respective other STR is inactive (see Fig. 7.3). For each point in STR_{skel}, we separately calculate the closest surface voxel belonging to STR_{t_1} and STR_{t_2}. Conversely, for each voxel in STR_o, we calculate the closest skeleton point in STR_{skel}. Afterwards, for each voxel v_o in STR_o, we look up the corresponding closest skeleton point and search

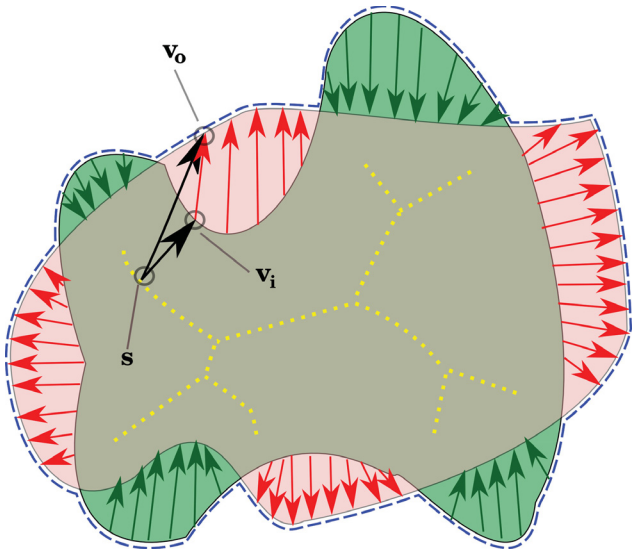
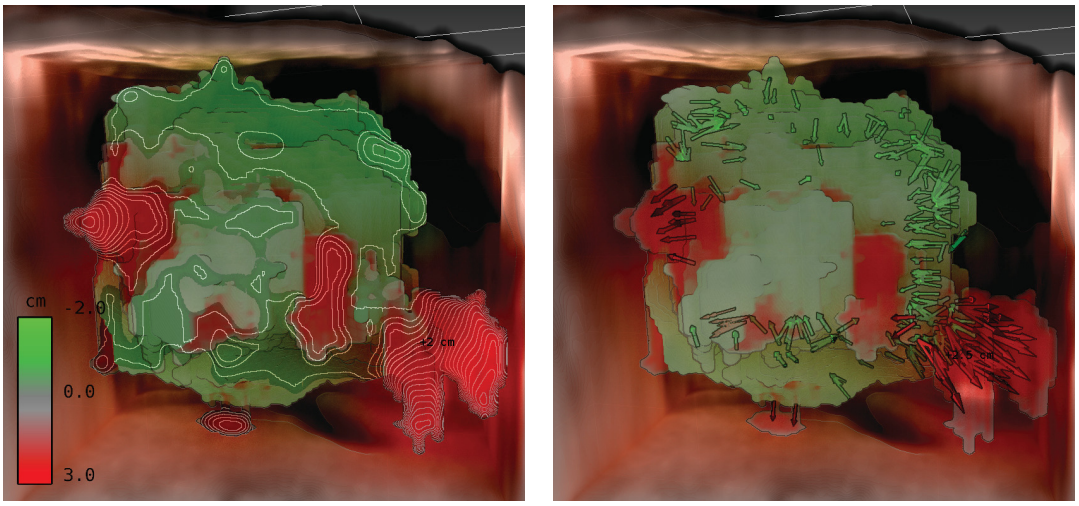


Figure 7.3: The image illustrates the method to find the changed surface of an STR between two points in time. For each voxel v_o on the outer surface of the union of the two STRs (blue), the closest of all skeleton points (yellow), s, is determined. Then, we consider the set of all voxels on the surface of the respective other STR to which s is the closest skeleton point. We search this set for the voxel v_i which is closest to v_o (see black arrows). The vector connecting the two represents the change of the tumor surface. The direction of this vector depends on whether of not v_o belonged to the more current time step.

the associated set of closest surface voxels from the respective other surface for the closest voxel v_i to v_o. Thus, we have found a vector between the two surfaces of the chronologically separated STRs representing the direction and amount of change towards or away from the skeleton of their intersection. An additional flag is saved to indicate whether v_o originally belonged to STR_{t_1} or STR_{t_2}, representing tumor recession or tumor growth, respectively.

In order to visualize the previously calculated surface changes, the user of our framework can choose from two visualization methods. Both use ray casting as a basis, with additional overlaid information. In the first method, for each image pixel, the ray's first hit point is used to look up the associated directional change information. The length of the surface change vector and the type of change (growth or shrinkage of the tumor) is then saved to a separate image target. After ray casting, a Gauss filter is applied to that target in order to smooth the resulting image. The smoothed image is then transformed to a height field image, with the color ranging from transparent



(a) (b)

Figure 7.4: The tumor surface closeup when comparing consecutive time steps. After skeletonization and the distance calculation, the user can view the resulting changes on a height map (a) which is projected onto the outer union of the two STRs. When moving the cursor over the height map, the corresponding surface difference value is displayed. In the case shown in this image, the STR has mainly shrunk (green areas), but has also developed two extensions which were not there before (red areas). As an alternative to the height field, the user can view the surface changes between time steps using arrows (b). The number of displayed arrows can be limited by specifying a minimal length threshold to avoid cluttering, while focusing on regions where the tumor has changed substantially.

to opaque green for regions of tumor shrinkage and from transparent to opaque red for regions of tumor growth. Regions where the STR has not changed remain transparent. The height field image is then blended over the original ray casting image (see Fig. 7.4(a)). By moving the mouse cursor over the resulting image, the unfiltered value of the length of the surface change vector is looked up from the ray casting result. The corresponding distance is displayed next to the cursor in order to reveal the exact local surface distance between the temporally differing STRs. Instead of using the height field, the user can also choose a geometrical representation of surface change which uses green or red arrows to represent changes of the tumor surface(see Fig. 7.4(b)). The arrows are generated on the GPU using geometry shaders to allow for interactive frame rates, even when displaying the associated arrow of each STR surface point. The number of shown arrows can be limited using length

thresholding to avoid cluttering and to focus on areas of large surface changes. Again, green arrows represent shrinkage, while red arrows represent growth of the tumor. As with the height map, the user can display the amount of surface change by hovering the mouse cursor over an arbitrary arrow.

Finally, the resulting STR surface rendering is embedded in a visualization of one of the datasets which were used to generate the STR. Using the same approach as in the tumor context view (see Section 7.3.2), this dataset containing surrounding anatomical structures of interest can be rendered simultaneously by exploiting a ray casting approach which differs between a context volume and the STRs (see Fig. 7.4(b)). This allows the user to track directly in which direction relative to the brain the tumor has extended or receded, and whether a possible tumor extension is close to sensitive brain regions which may make additional treatment necessary.

7.4 2D Visualization

In order to support the 3D visualization methods, our framework includes traditional 2D slice visualizations of each anatomical plane, showing linked transversal, coronal and sagittal slices of the context volume. In each 2D view, the STRs are overlaid using the same color scheme as in the 3D visualizations (see Fig. 7.1, right column). Additionally, the radius and center of the tumor lense (see Section 7.3.3) are projected onto each 2D view and displayed as a circle in single time step mode, similar to the one used in the corresponding TIC and TCV. By dragging the mouse over the TIC, the 2D views are updated to match the current 3D location of the cursor. This allows an interactive exploration of the exact outer structure of the overlapping STRs in 3D, while relying on the more unambiguous 2D views. To further visually link the 2D and 3D views, the transversal slice currently selected by the user can optionally be included in the TIC as well as the tumor overlap diagram (see Section 7.5) as a semi-transparent plane. This allows the user to connect anatomical structures visible in the slice to the more complex 3D representation of tumor tissue in those visualizations (see Fig. 7.1, upper left / middle).

7.5 Tumor Overlap Diagram

The *tumor overlap diagram* depicts the relative sizes of the three currently displayed STRs (see Fig. 7.5). It distinguishes between three kinds of regions:

- Only one modality is active (non-stacked ring segments)

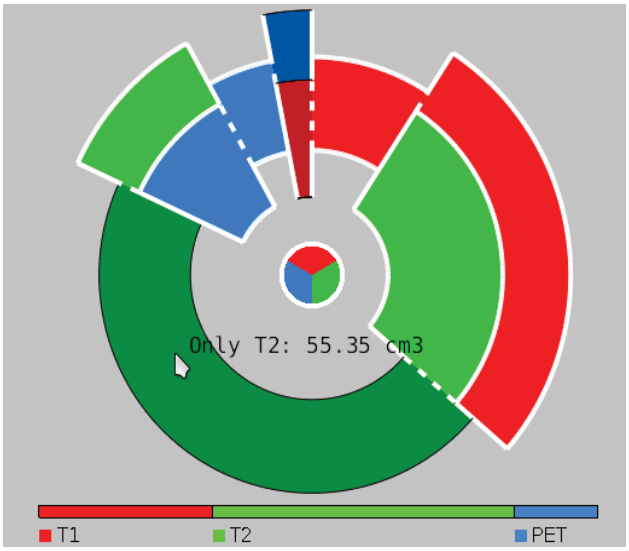
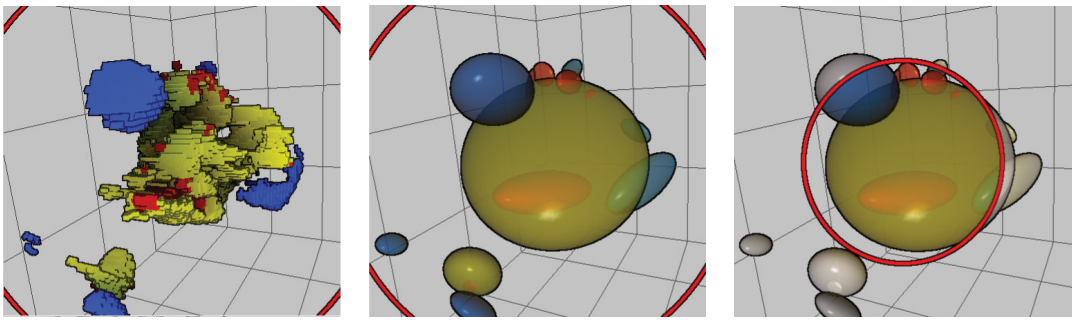


Figure 7.5: The tumor overlap diagram shows overlap sizes within the STRs as well as the original size of any STR compared to the others. Angle sizes represent the size of the overlaps, while areas of the same color represent the size of an STR without considering overlaps. Overlaps currently visible within the tumor lense are bordered in white. Stippled lines are used to indicate borders between segments which are part of the same STR.

- Two modalities are active (stacked ring segments)
- All modalities are active (circle in middle of diagram)

The diagram was designed to enable the user to compare the relative sizes of the STR overlaps using the opening angle of the segments. At the same time, the relative sizes of the STRs themselves are preserved by the color distribution within the diagram. For instance, by comparing the area of diagram regions colored in green with the other two colors in Fig. 7.5, the user can quickly determine that the T2 STR is roughly twice as large as the T1 STR and five times as large as the PET STR. However, judging by the radian measure of the non-stacked green ring segment, it is also obvious that a large area of the T2 STR is active where none of the other STRs show tumor tissue. Based on this observation, the user may want to mask this region in the 3D views and concentrate on those areas where two or more STRs coincide.

For this reason, the tumor overlap diagram is the central tool in our framework to select the currently active STRs and their overlaps. By selecting and deselecting segments of the diagram with the mouse, the user can specify an arbitrary configuration



(a) (b) (c)

Figure 7.6: The tumor ellipsoid diagram uses the closest fitting ellipsoids for each connected component of each currently active STR (a) to provide a simplified view of the tumor data (b). This diagram facilitates the spatial understanding of the STRs, as it avoids the visual cluttering and occlusions which may occur when applying ray casting on the original data. The tumor lense radius is visualized by graying out ellipsoid regions which are outside of the lense (c).

of active segments. When viewing a single time step, this will cause the tumor lense in the TIC to only show the active segments. For instance, this enables the user to only show the three-way overlap section along with the section where only the PET STR is active. This allows for a flexible comparison of all STR combinations. Additionally, when moving the mouse over each segment, the size of the corresponding segment is displayed in order to provide a quantitative assessment of the STRs.

7.6 Tumor Ellipsoid Diagram

As mentioned before, the structure of the STRs and the resulting overlaps is mostly irregular and not easily comprehensible when rendered in 3D due to the challenge of depth perception and occluded structures. While looking for a simplified representation of the STRs in 3D, we found that despite the irregularity of the tumor areas, their shape is generally still convex enough to be approximated by a closest fitting ellipsoid. Therefore, the *tumor ellipsoid diagram* provides a simplified view of the tumor area by replacing the original, arbitrarily shaped STRs with ellipsoids while following the same color scheme (see Fig. 7.6(a) and (b)). The result is similar to a 3D Venn diagram, which facilitates a quick understanding of the rough structure of the STRs. The point of view from which the diagram is seen is synchronized to the TIC, as well as the currently visible STRs or STR overlaps as specified by the user in the

tumor overlay diagram. Thus, the tumor ellipsoid diagram can be used as a point of reference to better comprehend the more detailed rendering provided by the TIC.

To compute the ellipses, a preprocessing step is necessary. The voxels belonging to each STR as well as all occurring overlapping areas between the individual STRs are first divided into their connected components. This division is necessary since a single ellipsoid is not sufficient to approximate STRs which are divided into several separate active regions, which occurs often within STRs derived from PET modalities. We then use the algorithm proposed by Khachiyan [36] to compute the closest fitting ellipsoid for each set of voxels within a connected component. After this process is complete, the resulting ellipsoid centers, radii and rotation matrices are stored together with the STR dataset.

To generate the diagram, the ellipsoids are rendered as geometric primitives, without the use of ray casting. The user can control the transparency of each visible ellipsoid. To guarantee correct order independent transparency, we employ an A-buffer as introduced by Carpenter [8] to store a list of fragments per pixel which is used after rendering to correctly sort the stored fragments based on depth and correctly blend individual ellipsoids over each other. The screen space ambient occlusion shading used in the TIC is also employed for this diagram, as it enables the user to better judge the size and relative depth of the different ellipsoids. Additionally, the ellipsoid surfaces are shaded using the Phong reflection model with the light source located at the point of view of the user to convey their 3D shape. The red ring which indicates the current radius of the tumor lense is also displayed in this diagram to support the visual connection between the views. Regions of ellipsoids which are outside of the lense are displayed in shades of gray to distinguish them from regions within the lense (see Fig. 7.6(c)).

7.7 Domain Expert Feedback

We demonstrated our framework to two domain experts familiar with the process of diagnosis and treatment of brain tumors. Both agreed that our framework could be an overall useful visual tool. One complimented the use of nearest filtering in the 3D ray casting views and the subsequent ability to exactly identify regions in the TIC with only few voxels of tumor tissue which might nevertheless require treatment.

Both experts agreed that the tumor overlap diagram required some adaptation to its approach of presenting information with circular elements as opposed to more traditional bar charts. However, after a short learning phase, one domain expert was able to judge within seconds which modalities and which overlaps were dominant in a given dataset. One of the experts suggested the implementation of an additional

interactive table to control the visibility of tumor overlaps, as it shows more clearly what the individual ring segments signify exactly (see Fig. 7.1, middle bottom). Another suggestion which we adopted was to include an optional non-overlap mode in the diagram which allows the user to select STRs without considering overlaps, which results in simplified visualizations in the TIC as well as the tumor ellipsoid diagram and provides the user with a less complex overview of the original data. Furthermore, a rectangular bar that displays the STR sizes without overlaps next to each other was added below the tumor overlap diagram in order to provide a more conventional point of reference (see Fig. 7.5).

The tumor ellipsoid diagram was described by both as useful for explaining a diagnosis to a patient rather than the diagnosis itself, since the convexity of the ellipsoids causes tumor overlaps to be exaggerated at times. As one of the experts noted that the shape and relative position of the ellipsoids could be indistinguishable depending on the camera position, we added a simple form of shading to the diagram (see Section 7.6).

Finally, the tumor surface closeup in consecutive time steps was commended by both experts as being useful to judge the change of a tumor over time. This task is challenging due to the potentially complex structure of brain tumors as well as the inhomogeneity of the changes a tumor may undergo. The experts welcomed our approach as it shows complex information of the whole tumor in one simple to understand visualization.

7.8 Implementation and Performance

The framework has been implemented in C++ using OpenGL and GLSL, as well as the Qt graphical user interface. The preprocessing necessary for the tumor ellipsoid diagram described in Section 7.6 was implemented on the CPU. Its runtime depends on the number of connected voxel components within the STR dataset and typically lasts up to two minutes on an Intel Core i7 CPU. The skeletonization and surface difference detection process described in Section 7.3.4 lasts a few seconds in the current CPU implementation. These processes could be implemented in a highly parallel framework like OpenCL to further speed them up, if necessary. Otherwise, the system is completely interactive. Using an Intel Core i7 CPU in combination with an nVidia GeForce GTX 480, our system typically achieves frame rates of around 20 fps when updating the 3D views. The Voreen network necessary to assemble the multiple parts of the framework as seen in Figure 7.1 is shown in Figure 7.7.

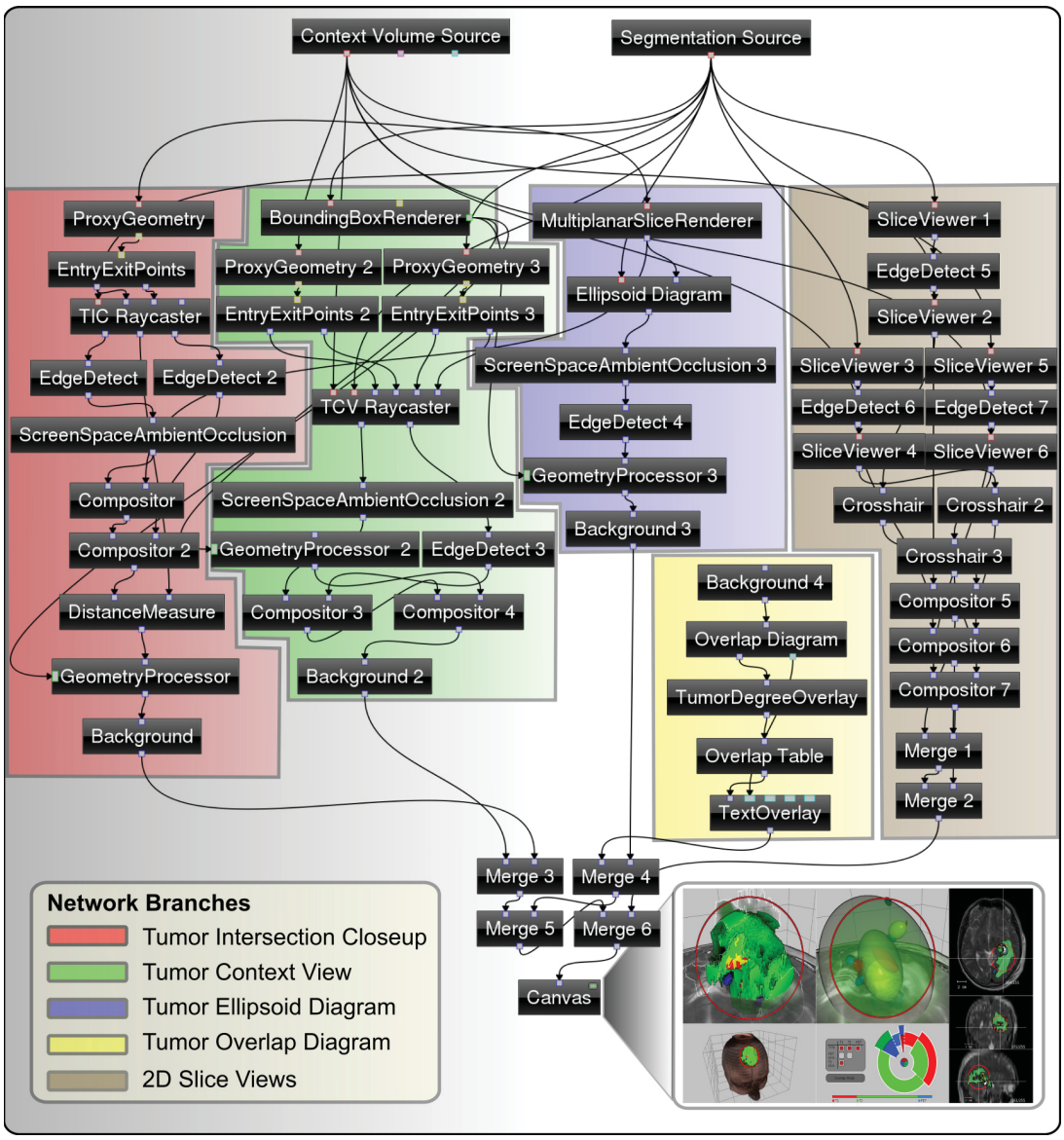


Figure 7.7: An overview of the Voreen network used to assemble the framework. At the top, the data source processors load the datasets and distribute them to the different branches of the network. Each branch serves the purpose of implementing one of the 3D views, one of the diagrams or the 2D slice views. The results of all five branches are then combined by using a set of merging processors. Afterwards, the final image is displayed on one final rendering canvas.

7.9 Conclusion and Future Work

We have presented an interactive framework for the exploration of tumor segmentations derived from multimodal brain images. Our framework provides the user with an exact understanding of the inner structure of the union of all STRs. The user can specify arbitrary subsets of overlapping areas or areas where only one STR is active. These subsets are directly visible since areas that are currently not of interest for the viewer are removed. Our shading approach supports a quantitative understanding and comparison of the shape and size of different kinds of imaging modalities. Exact size and distance values are available to the user in the different views due to information overlays displayed next to the mouse cursor. To provide an overview of the STR structure, the user can refer to the tumor ellipsoid diagram which reduces the spatial complexity by providing an approximate 3D Venn diagram.

The tumor overlap diagram provides an instant overview of the entire size of each STR as well as the size of overlaps with other segmented regions. As the area of a color in the diagram directly translates to the size of the associated STR, it could be a powerful tool when used as a kind of glyph to view many datasets next to each other, for example to visually determine relations between the size of the STR and the imaging method.

We demonstrated our framework to domain experts and used their feedback to improve our visualizations. By modifying our color scheme as well as extending the tumor overlap diagram, it may be possible to visualize more than three modalities at the same time. Our system could also be extended to include other kinds of modalities such as fMRI and be subsequently used for surgery planning.

Chapter 8

Conclusions

Volume illumination is a relatively new branch of volume graphics that is still rapidly growing. Nevertheless, much research focusing on the implementation of realistic illumination effects has already been presented by the visualization community. This development has supported the extension of volume rendering beyond the scientific domain towards its application in special effects for the film and gaming industry. For some scenarios, the representation of parts of a scene using volumetric representations is the most natural approach, e.g., regarding spatial atmospheric effects. Half angle slicing, for instance, is already used to efficiently model dynamic particle smoke effects [19] (see Figure 8.1). With increasing computation power and more memory efficient approaches, techniques like the subsurface scattering approach presented in Chapter 5 or the global illumination techniques that were reviewed in Chapter 6 could be applied to large dynamic scenes that contain participating media to obtain natural-looking renderings. Conversely, additional illumination phenomena that can already be interactively simulated using polygonal graphics could be transferred to volume graphics to produce even more realistic looking volume renderings in real time.

As far as the application of volume illumination in scientific visualizations is concerned, there is still much research ahead. The user study presented in Chapter 6 is just a starting point. Its conclusions could be used to carry out more specific studies, possibly with domain experts, to indicate how real-world applications could benefit from volume illumination. In particular, its influence on the user during interaction with the scene as opposed to viewing static images should be explored. The guidelines presented at the end of Chapter 6 could be used by software engineers to implement new lighting techniques or combine existing ones in order to develop an illumination model tailored to their application. In general, our results indicate that such a model should produce subdued lighting effects which are subject to some constraints (e.g., a fixed location of the light source relative to the viewer). This approach restricts the possible configurations of the lighting effects and avoids an

Figure 8.1: Smoke particles simulated using half angle slicing. The image was generated using the Nvidia GPU Computing SDK [60].

excess of information which would not support, but confuse the user. Overall, in the context of scientific visualizations, a volume illumination technique should only be as complex as necessary, and as simple as possible.

This principle was applied in the framework presented in Chapter 7. Screen space ambient occlusion was used to increase the depth perception and help the user to correctly perceive the complex structure of the segmented tumor data. This relatively simple technique was chosen since it produces subtle, locally restricted occlusion shadows without introducing an additional parameter like a light source location. It provides the right amount of lighting effects for this situation, in which photo-realistic images are not as important as providing additional information about the shape of the tumor masks. Furthermore, it requires no costly data preprocessing and can be easily integrated into an extensive rendering framework without major adaptations. This is but one of many possible examples for the adaptability of volume illumination in scientific applications. A further development of such applications with similar uses of volume illumination could support its acceptance as a valuable visualization tool in the future.

Bibliography

[1] D. C. Banks and K. Beason. Decoupling illumination from isosurface generation using 4d light transport. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1595–1602, 2009.

[2] K. M. Beason, J. Grant, D. C. Banks, B. Futch, and M. Y. Hussaini. Pre-computed illumination for isosurfaces. In *Visualization and Data Analysis '06 (SPIE Vol. 6060)*, pages 1–11, 2006.

[3] J. Beyer, M. Hadwiger, S. Wolfsberger, and K. Buhler. High-quality multimodal volume rendering for preoperative planning of neurosurgical interventions. *IEEE Transactions on Visualization and Computer Graphics*, 13:1696–1703, 2007.

[4] J. F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, July 1977.

[5] C. Boucheny, G.-P. Bonneau, J. Droulez, G. Thibault, and S. Ploix. A perceptive evaluation of volume rendering techniques. In *4th Symp. on Applied perception in graphics and visualization*, pages 83–90, 2007.

[6] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[7] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Symp. on Volume visualization*, pages 91–98, 1994.

[8] L. Carpenter. The A-buffer, an antialiased hidden surface method. *SIGGRAPH Comput. Graph.*, 18(3):103–108, January 1984.

[9] M.-Y. Chan, Y. Wu, W.-H. Mak, W. Chen, and H. Qu. Perception-based transparency optimization for direct volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 15(6):1283–1290, 2009.

[10] N. D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer*, 21, 2005.

[11] J. J. Corso, E. Sharon, S. Dube, S. El-saden, and U. Sinha. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions on*, 27(5):629 –640, may 2008.

[12] T. J. Cullip and U. Neumann. Accelerating volume reconstruction with 3d texture hardware. Technical report, University of North Carolina at Chapel Hill, 1993.

[13] J. Díaz, H. Yela, and P. Vázquez. Vicinity occlusion maps - enhanced depth perception of volumetric models. In *Computer Graphics Int.*, pages 56–63, 2008.

[14] S. Diepenbrock, J.-S. Praßni, F. Lindemann, H.-W. Bothe, and T. Ropinski. Preoperative planning of brain tumor resections. *IEEE Visualization Contest*, 2010.

[15] S. Diepenbrock, J.-S. Praßni, F. Lindemann, H.-W. Bothe, and T. Ropinski. 2010 ieee visualization contest winner: Interactive planning for brain tumor resections. *IEEE Computer Graphics and Applications*, Sept/Oct 2011.

[16] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-time volume graphics*. AK Peters, Limited, 2006.

[17] E. B. Goldstein. *Sensation and Perception*, pages 179–184. Wadsworth, 2007.

[18] R. Green. Spherical harmonic lighting: The gritty details. In *Game Deveopers Conference*, 2003.

[19] S. Green. Volumetric particle shadows. *Technical Report, Nvidia Developer Zone*, 2008.

[20] A. L. Grosu, W. A. Weber, M. Franz, S. Stärk, M. Piert, R. Thamm, H. Gumprecht, M. Schwaiger, M. Molls, and C. Nieder. Reirradiation of recurrent high-grade gliomas using amino acid PET(SPECT)/CT/MRI image fusion to determine gross tumor volume for stereotactic fractionated radiotherapy. *International Journal of Radiation Oncology*Biology*Physics*, 63(2):511–9, 2005.

[21] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 40–, Washington, DC, USA, 2003. IEEE Computer Society.

[22] M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler. GPU-accelerated deep shadow maps for direct volume rendering. In *ACM SIGGRAPH/EG Conference on Graphics Hardware*, pages 27–28, 2006.

[23] X. Hao and A. Varshney. Real-time rendering of translucent meshes. *ACM Transactions on Graphics*, 23(2):120–142, 2004.

[24] P. Herghelegiu, V. Manta, R. Perin, S. Bruckner, and M. E. Gröller. Biopsy planner - visual analysis for needle pathway planning in deep seated brain tumor biopsy. *Computer Graphics Forum*, 31(3):1085–1094, June 2012.

[25] F. Hernell, P. Ljung, and A. Ynnerman. Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *IEEE/EG Int. Symp. on Volume Graphics*, pages 1–8, 2007.

[26] F. Hernell, P. Ljung, and A. Ynnerman. Interactive global light propagation in direct volume rendering using local piecewise integration. In *IEEE/EG Int. Symp. on Volume and Point-Based Graphics*, pages 105–112, 2008.

[27] W. Hibbard and D. Santek. Interactivity is the key. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, VVS '89, pages 39–43, New York, NY, USA, 1989. ACM.

[28] H. H. Hu, A. A. Gooch, S. H. Creem-Regehr, and W. B. Thompson. Visual cues for perceiving distances from objects to surfaces. *Presence: Teleoper. Virtual Environ.*, 11(6):652–664, 2002.

[29] G. S. Hubona, P. N. Wheeler, G. W. Shirah, and M. Brandt. The relative contributions of stereo, lighting, and background scenes in promoting 3D depth visualization. *ACM Trans. Comput.-Hum. Interact.*, 6(3):214–242, 1999.

[30] P. Jannin, O. Fleig, E. Seigneuret, C. Grova, X. Morandi, and J. Scarabin. Multimodal and multi-informational neuronavigation. *Proc. of CARS-Computer Assisted Radiology and Surgery*, pages 167–172, 2000.

[31] H. Jensen, S. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *ACM SIGGRAPH*, pages 511–518, 2001.

[32] D. Jönsson, J. Kronander, T. Ropinski, and A. Ynnerman. Historygrams: Enabling interactive global illumination in direct volume rendering using photon mapping. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2364–2371, 2012.

[33] D. Jönsson, E. Sundén, A. Ynnerman, and T. Ropinski. A survey of volumetric illumination techniques for interactive volume rendering. In *Computer Graphics Forum*. Wiley Online Library, 2013.

[34] J. T. Kajiya and B. P. V. Herzen. Ray tracing volume densities. In *ACM SIGGRAPH*, pages 165–174, 1984.

[35] D. Kersten, P. Mamassian, and D. Knill. Moving cast shadows and the perception of relative depth. *Max-Planck-Institute for Biological Cybernetics Technical Report*, 1994.

[36] L. G. Khachiyan. Rounding of polytopes in the real number model of computation. *Math. Oper. Res.*, 21(2):307–320, May 1996.

[37] J. Kniss, S. Premoze, C. Hansen, and D. Ebert. Interactive translucent volume rendering and procedural modeling. In *IEEE Visualization*, pages 109–116, 2002.

[38] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, 2003.

[39] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings IEEE Visualization 2003*, 2003.

[40] J. Křivánek, J. Konttinen, S. Pattanaik, K. Bouatouch, and J. Žára. Fast approximation to spherical harmonics rotation. In *ACM SIGGRAPH Sketches*, page 154, 2006.

[41] M. S. Langer and H. H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29(6):649–660, 2000.

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

[43] F. Lindemann and T. Ropinski. About the influence of illumination models on image comprehension in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1922–1931, 2011.

[44] F. Lindemann, T. Brix, S. Diepenbrock, J.-S. Praßni, B. Hemmer, S. Linnemann, P. Weingardt, G. Wilde, and K. Hinrichs. Visual analysis of phase transitions and polarisation domains in barium titanate. *IEEE Visualization Contest*, 2012.

[45] F. Lindemann, K. R. Laukamp, A. Jacobs, and K. H. Hinrichs. Interactive comparative visualization of multimodal brain tumor segmentation data. In *Proceedings of the 18th International Fall Workshop on Vision, Modeling, and Visualization (VMV13)*, pages 105–112, 2013.

[46] F. Lindemann and T. Ropinski. Advanced light material interaction for direct volume rendering. In *IEEE/EG Int. Symp. on Volume Graphics*, pages 101–108, 2010.

[47] M. Livingston. *Vision and Art: The Biology of Seeing*. Harry N. Abrams, New York, 2002.

[48] P. Ljung, F. Hernell, and A. Ynnerman. Local ambient occlusion in direct volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 15(2), 2009.

[49] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987.

[50] T. Luft, C. Colditz, and O. Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph.*, 25:1206–1213, July 2006.

[51] K. Martin, B. Hoffman, A. Cedilnik, B. King, and A. Nuendorf. *Mastering CMake: A cross-platform build system*. Kitware Incorporated, 2003.

[52] W. Matusik, H. Pfister, M. Brand, and L. McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

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

[54] J. Meyer-Spradow. *Interaktive Entwicklung Raycasting-basierter Visualisierungs-Techniken für medizinische Volumen-Daten mit Hilfe von Datenflussnetzwerken*. 2009.

[55] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *Computer Graphics and Applications, IEEE*, 29(6):6–13, 2009.

[56] M. Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[57] A. Ngan, F. Durand, and W. Matusik. Experimental validation of analytical BRDF models. In *ACM SIGGRAPH 2004 Sketches*, page 90. ACM, 2004.

[58] K. Nicolaides. *The Natural Way to Draw*. Houghton Mifflin, 1941.

[59] M. Nulkar and K. Mueller. Splatting with shadows. In *Volume Graphics*, pages 35–50. Springer-Verlag, 2001.

[60] NVIDIA. GPU computing SDK 4.1, 2012.

[61] D. Patel, S. Bruckner, I. Viola, and E. Groeller. Seismic volume visualization for horizon extraction. In *IEEE Pacific Visualization (PacificVis 2010)*, 2010.

[62] E. Penner and R. Mitchell. Isosurface ambient occlusion and soft shadows with filterable occlusion maps. In *IEEE/EG International Symposium on Volume and Point-Based Graphics (VG)*, pages 57–64, 2008.

[63] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[64] B. Pirotte, S. Goldman, O. Dewitte, N. Massager, D. Wikler, F. Lefranc, N. O. Ben Taib, S. Rorive, P. David, J. Brotchi, and M. Levivier. Integrated positron emission tomography and magnetic resonance imaging-guided resection of brain tumors: a report of 103 consecutive procedures. *Journal of Neurosurgery*, 104(2):238–53, 2006.

[65] C. Rezk-Salama. GPU-based monte-carlo volume raycasting. In *Pacific Graphics*, 2007.

[66] C. Rezk-Salama and A. Kolb. A vertex program for efficient box-plane intersection. In *In Proc. Vision, Modeling and Visualization*, pages 115–122, 2005.

[67] C. Rezk-Salama, M. Hadwiger, T. Ropinski, and P. Ljung. Advanced illumination techniques for GPU volume raycasting. In *ACM SIGGRAPH Courses Program*, 2009.

[68] C. Rieder, F. Ritter, M. Raspe, and H.-O. Peitgen. Interactive Visualization of Multimodal Volume Data for Neurosurgical Tumor Treatment. *Computer Graphics Forum (Special Issue on Eurographics Symposium on Visualization)*, 27(3):1055–1062, 2008.

[69] C. Rieder, M. Schwier, H. K. Hahn, and H.-O. Peitgen. High-quality multimodal volume visualization of intracerebralral pathological tissue. In *Proc. VCBM*, pages 167–176, 2008.

[70] C. Rieder, A. Weihusen, C. Schumann, S. Zidowitz, and H.-O. Peitgen. Visual Support for Interactive Post-Interventional Assessment of Radiofrequency Ablation Therapy. *Computer Graphics Forum (Special Issue on Eurographics Symposium on Visualization)*, 29(3):1093–1102, 2010.

[71] T. Ritschel. Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets. In *Short Paper Proceedings of Eurographics 2007*, pages 17–20, 2007.

[72] T. Ritschel, T. Grosch, and H.-P. Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 75–82. ACM, 2009.

[73] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser. Smart hardware-accelerated volume rendering. In *Proceedings of the symposium on Data visualisation 2003*, VISSYM '03, pages 231–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[74] T. Ropinski, C. Döring, and C. Rezk-Salama. Interactive volumetric lighting simulating scattering and shadowing. In *PacificVis 2010*, pages 169–176, 2010.

[75] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. H. Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics)*, 27(2):567–576, 2008.

[76] M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert. Obscurance-based volume rendering framework. In *IEEE/EG International Symposium on Volume and Point-Based Graphics (VG)*, pages 113–120, 2008.

[77] P. Schlegel, M. Makhinya, and R. Pajarola. Extinction-based shading and illumination in GPU volume ray-casting. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):1795–1802, 2011.

[78] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. In *Computer Graphics Forum (Eurographics/IEEE VGTC Symposium on Visualization 2009 issue)*, pages 855–862, 2009.

[79] L. Serra, R. A. Kockro, C. G. Guan, N. H. E. C. Lee, E. C. K. Lee, Y. H. Lee, C. Chan, and W. L. Nowinski. Multimodal volume-based tumor neurosurgery planning in the virtual workbench. In *in the Virtual Workbench, Proc. IEEE VRAIS'98*, pages 167–173, 1998.

[80] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 73–80, New York, NY, USA, 2007. ACM.

[81] J. Sherman, D. Prevedello, L. Shah, P. Raghavan, N. Pouratian, R. Starke, M. Lopes, M. Shaffrey, and D. Schiff. Mr imaging characteristics of oligodendroglial tumors with assessment of 1p/19q deletion status. *Acta Neurochirurgica*, 152:1827–1834, 2010.

[82] P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM SIGGRAPH*, pages 527–536, 2002.

[83] J. Snyder. Code generation and factoring for fast evaluation of low-order spherical harmonic products and squares. Technical report, Microsoft Corporation, 2006.

[84] A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *IEEE Visualization*, page 47, 2003.

[85] W. Sun and A. Mukherjee. Generalized wavelet product integral for rendering dynamic glossy objects. In *ACM SIGGRAPH*, pages 955–966, 2006.

[86] E. Sunden, A. Ynnerman, and T. Ropinski. Image plane sweep volume illumination. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2125–2134, 2011.

[87] C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauß. Enhancing Slice-based Visualizations of Medical Volume Data. In D. Fellner, T. Möller, and S. Spencer, editors, *Proceedings of the Eurographics / IEEE VGTC Symposium on Visualization*, pages 123–130. EGPub, 2006.

[88] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *Proceedings of IEEE Visualization'04*, pages 139–145, 2004.

[89] V. Šoltészová, D. Patel, S. Bruckner, and I. Viola. A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum*, 29(3):883–891, 2010.

[90] L. C. Wanger. The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *Symp. on Interactive 3D graphics*, pages 39–42, 1992.

[91] L. C. Wanger, J. A. Ferwerda, and D. P. Greenberg. Perceiving spatial relationships in computer-generated images. *IEEE Comput. Graph. Appl.*, 12(3):44–51, 54–58, 1992.

[92] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting reflectance functions from complex surfaces. *ACM SIGGRAPH*, 26(2):255–264, 1992.

[93] Y. Wu, H. Qu, K.-K. Chung, M.-Y. Chan, and H. Zhou. Quantitative effectiveness measures for direct volume rendered images. In *PacificVis*, pages 1–8, 2010.

[94] C. Wyman, S. Parker, P. Shirley, and C. Hansen. Interactive display of isosurfaces with global illumination. *IEEE Trans. on Visualization and Computer Graphics*, 12:186–196, 2006.

[95] A. Yonas. *Attached and Cast Shadows*, pages 100–109. Praeger Publishers, 1979.

[96] N. B. Zafar, J. Akesson, D. Roble, and K. Museth. Scattered spherical harmonic approximation for accelerated volume rendering. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 148, 2006.

[97] C. Zhang and R. Crawfis. Volumetric shadows using splatting. In *IEEE Visualization*, pages 85–92, 2002.

[98] S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. In *EG Workshop on Rendering*, pages 45–55, 1998.

Acronyms

ANOVA	analysis of variance
BRDF	bidirectional reflectance distribution function
CPU	central processing unit
CT	computed tomography
DVR	direct volume rendering
FBO	frame buffer object
FPS	frames per second
GLSL	OpenGL shading language
GPU	graphics processing unit
MRI	magnetic resonance imaging
OpenCL	open computing language
OpenGL	open graphics library
PET	positron emission tomography
SH	spherical harmonics
STR	segmented tumor region
TCV	tumor context view
TIC	tumor intersection closeup