TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

DISSERTATION

Interactive Illustrative Volume Visualization

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Institut für Computergraphik und Algorithmen
Abteilung für Computergraphik

eingereicht an der Technischen Universität Wien
bei der Fakultät für Informatik

von

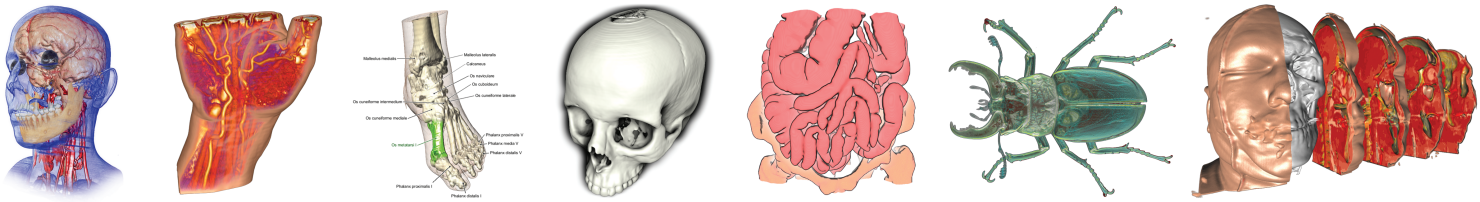
Dipl.-Ing. Stefan Bruckner
Matrikelnummer 9926214
Castellezgasse 20/1/1
1020 Wien

Wien, im März 2008

STEFAN BRUCKNER

Interactive Illustrative Volume Visualization

Dissertation



TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Kurzfassung

Illustrationen spielen eine wichtige Rolle in der Kommunikation komplexer Sachverhalte. Ihre Herstellung ist jedoch schwierig und teuer. Dreidimensionale bildgebende Verfahren haben sich in den letzten Jahren als unverzichtbares Werkzeug in Disziplinen wie der medizinischen Diagnose und Behandlungsplanung, im technischen Bereich (z.B. Materialprüfung), der Biologie, und der Archäologie etabliert. Modalitäten wie Röntgencomputertomographie (CT) oder Magnetresonanztomographie (MRT) generieren täglich hochauflösende volumetrische Scans. Es leuchtet nicht ein, dass trotz dieses Datenreichtums die Produktion einer Illustration noch immer ein aufwendiger und zum Großteil manueller Prozess ist.

Diese Dissertation beschäftigt sich mit der computerunterstützten Erstellung von Illustrationen direkt auf Basis solcher Volumendaten. Zu diesem Zweck wird das Konzept eines direkten Volumenillustrationssystems eingeführt. Dieses System erlaubt die Gestaltung einer Illustration direkt anhand von gemessenen Daten, ohne dass ein zusätzlicher Modellierungsschritt notwendig wäre. Abstraktion, ein wichtiger Bestandteil traditioneller Illustrationen, wird verwendet um visuelle Überladung zu vermeiden, wichtige Strukturen hervorzuheben und versteckte Details sichtbar zu machen. Abstraktionstechniken beschäftigen sich einerseits mit der Erscheinung von Objekten und erlauben die flexible künstlerische Schattierung von Strukturen in volumetrischen Datensätzen. Andererseits kontrollieren diese Techniken welche Objekte sichtbar sein sollen. Neue Methoden zur Generierung von Transparenz- und Explosionsdarstellungen werden hierfür vorgestellt. Die präsentierten Visualisierungstechniken verwenden die Fähigkeiten moderner Graphikhardware um eine interaktive Darstellung zu ermöglichen.

Das resultierende System erlaubt die Erstellung von expressiven Illustrationen direkt anhand von volumetrischen Daten und hat eine Vielzahl von Anwendungen wie etwa die medizinische Ausbildung, Patientenaufklärung und wissenschaftliche Kommunikation.

Abstract

Illustrations are essential for the effective communication of complex subjects. Their production, however, is a difficult and expensive task. In recent years, three-dimensional imaging has become a vital tool not only in medical diagnosis and treatment planning, but also in many technical disciplines (e.g., material inspection), biology, and archeology. Modalities such as X-Ray Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) produce high-resolution volumetric scans on a daily basis. It seems counter-intuitive that even though such a wealth of data is available, the production of an illustration should still require a mainly manual and time-consuming process.

This thesis is devoted to the computer-assisted generation of illustrations directly from volumetric data using advanced visualization techniques. The concept of a direct volume illustration system is introduced for this purpose. Instead of requiring an additional modeling step, this system allows the designer of an illustration to work directly on the measured data. Abstraction, a key component of traditional illustrations, is used in order to reduce visual clutter, emphasize important structures, and reveal hidden detail. Low-level abstraction techniques are concerned with the appearance of objects and allow flexible artistic shading of structures in volumetric data sets. High-level abstraction techniques control which objects are visible. For this purpose, novel methods for the generation of ghosted and exploded views are introduced.

The visualization techniques presented in this thesis employ the features of current graphics hardware to achieve interactive performance. The resulting system allows the generation of expressive illustrations directly from volumetric data with applications in medical training, patient education, and scientific communication.

Books have the same enemies as people: fire, humidity, animals, weather, and their own content.

— *Paul Valéry*

Contents

A classic is something that everybody wants to have read and nobody wants to read.

— *Mark Twain*

Preface

THIS thesis represents a summary of work carried out from 2004 to 2007 at the Institute of Computer Graphics and Algorithms, Vienna University of Technology under the kind guidance of Meister Eduard Gröller. I want to thank him and my past and present colleagues including Sören Grimm, Armin Kanitsar, Ivan Viola, Matej Mlejnek, Alexandra La Cruz, Ernesto Coto, Peter Rautek, Peter Kohlmann, Muhammad Muddassir Malik, Maurice Termeer, Erald Vuçini, Daniel Patel, Raphael Fuchs, Martin Haidacher, and all the others for creating a friendly and stimulating working environment. Every single one of them contributed in making this time a particularly educational, creative, and enjoyable period of my life. I especially want to thank my girlfriend Petra, to whom I dedicate this thesis, for her patience and support during stressful times – without her this would not have been possible.

The work presented in this thesis was carried out as part of the ex**vis**ation project[1] supported by the Austrian Science Fund (FWF) grant no. P18322. The data sets used are courtesy of AFGA HealthCare, the OsiriX Foundation, the United States National Library of Medicine, Lawrence Berkeley National Laboratory, and General Electric.

Vienna, Austria, March 2008 *Stefan Bruckner*

[1] `http://www.cg.tuwien.ac.at/research/vis/exvisation`

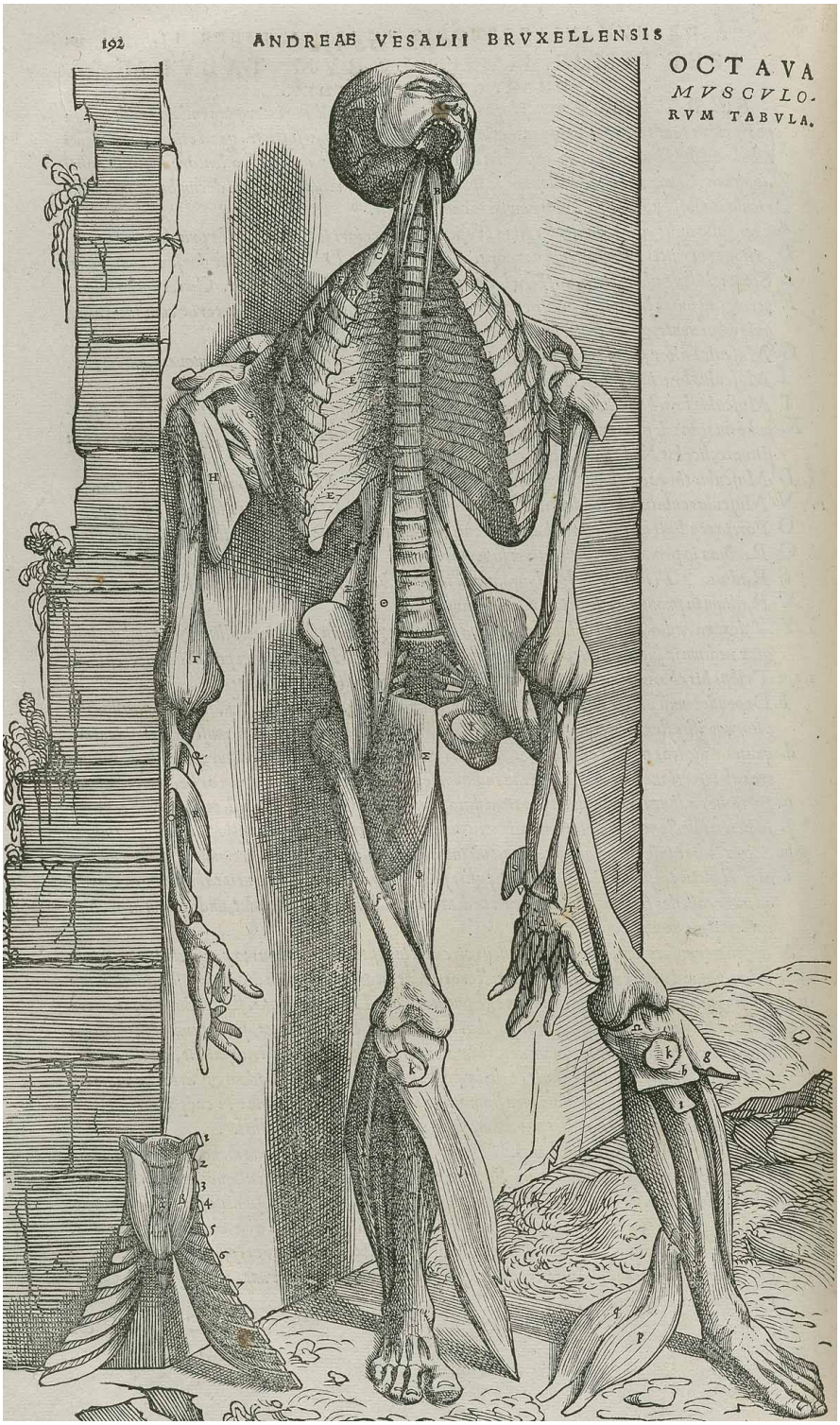


Figure 1.1 – Illustration from Andreas Vesalius' *De Humani Corporis Fabrica* (1543).

In the beginning the Universe was created. This has made a lot of people very angry and has been widely regarded as a bad move.

— *Douglas Adams*

CHAPTER

1 Introduction

Illustrations play a major role in the education process. Whether used to teach a surgical or radiological procedure, to illustrate normal or aberrant anatomy, or to explain the functioning of a technical device, illustration significantly impacts learning. This chapter reviews the history of illustration and introduces the notion of direct volume illustration, i.e., the computer-assisted generation of illustrations based on volumetric data.

1.1 Traditional Illustration

EVER since the dawn of man, humans have sought out ways of communicating acquired knowledge to contemporaries and future generations alike. The human visual channel is – due to its high bandwidth – a very effective means for this purpose. Images, unlike written descriptions, require little previous knowledge and can therefore be intuitively understood by a broad audience. The artist's ability to create a tangible image of an idea, concept, or discovery has been indispensable in communicating that idea to other individuals [21].

1.1.1 Prehistory

The beginning of illustration long predates written records. Initial pictorial iconography, which took the form of rock paintings and petroglyphs (engravings carved into a stone surface), first appeared in the Upper Palaeolithic period, as early as 30,000 B.C. These ancient artists were hunters whose very survival depended upon their knowledge of the living machinery. Thus, some of these depictions are suspected to have had the same basic purpose as modern illustrations – to preserve and pass on information. Figure 1.2 shows an early "X-ray style" rock painting discovered in Australia.

1.1.2 Antiquity

The scribes, painters, and stone cutters of ancient Egypt were probably among the first commercial artists. Their visual and written language, hieroglyphics,

Figure 1.2 – Aboriginal "X-ray style" rock painting at Kakadu National Park, Australia (ca. 6,000 B.C).

depicted religious practices, political propaganda, scientific data, and daily life. Though possessing some sound fundamentals in science and art, their drawings and pictographs failed to combine height, width, and depth into one view of an object.

The ancient Greeks, led by their intense interest in the beauty and structure of the human body, were the first to conduct serious investigations into physiology and anatomy. Hippocrates (460-375 B.C), Father of Medicine, and his contemporaries were responsible for the foundation of science and medicine as empirical disciplines. Aristotle (384-322 B.C) was among the first to dissect and study animal anatomy. He established an anatomical nomenclature and is also credited with one of the first anatomical illustrations. The Ptolemaic Medical School of Alexandria became a center of medical study. It was here

where the first true medical illustrations were produced. The great age of discovery ended when Alexandria was systematically destroyed in the early part of the Christian era. A period of similar intellectual and artistic freedom was not seen again until the advent of the Renaissance.

1.1.3 Renaissance

During the Renaissance period, major advancements in painting and illustration took place through the work of individuals such as Leon Battista Alberti (1404-1472), Leonardo da Vinci (1452-1519), and Andreas Vesalius (1514-1564). Artist and architect Leon Battista Alberti's treatise *Della Pictura* (On Painting) of 1436 was the first modern manual for painters containing one of the early treatments of perspective drawing. Leonardo da Vinci's artistic ability combined with his scientific curiosity provided the means and impetus for a merging of visual art with science and invention. He took interest in anatomy, dissecting more than thirty cadavers and making hundreds of drawings (see Figure 1.3). Andreas Vesalius produced the first true atlas of human anatomy. His work, *De Humani Corporis Fabrica* (On the Fabric of the Human Body), appeared in 1543. It was realized that if certain artistic holdings should be skillfully abandoned, greater scientific value could be achieved in the illustrations through the cooperation of practitioners of medicine and art. A specialty had been established combining the ability to draw beautifully, and the ingenuity to set forth scientific facts in an understandable fashion [67]. Figure 1.1 shows an example of Vesalius' work.

1.1.4 Modern Illustration

The industrial revolution further refined the field of illustration. The invention of lithography helped in spreading the use of high-quality illustrations. Mass production and outsourcing created the need to adopt conventions and standards in illustration that were universally understood. The use of now established principles of perspective allowed illustrators an objective recording of one's visual experience. Additionally, during this period illustrators began to increasingly use variant line weights to emphasize mass, proximity, and scale which helped to make a complex line drawing more understandable to the layperson. Cross hatching, stippling, and other basic techniques gave greater depth and dimension to the subject matter.

The Johns Hopkins School of Medicine began training illustrators in 1911 through the School of Medicine's Department of Art as Applied to Medicine. It was the first medical illustration department in the world, and for decades, the majority of credited medical illustrators were taught at Hopkins. The department was headed by Max Brödel (1870-1941) whom many consider to be the father of modern medical illustration. Brödel perfected his half-tone

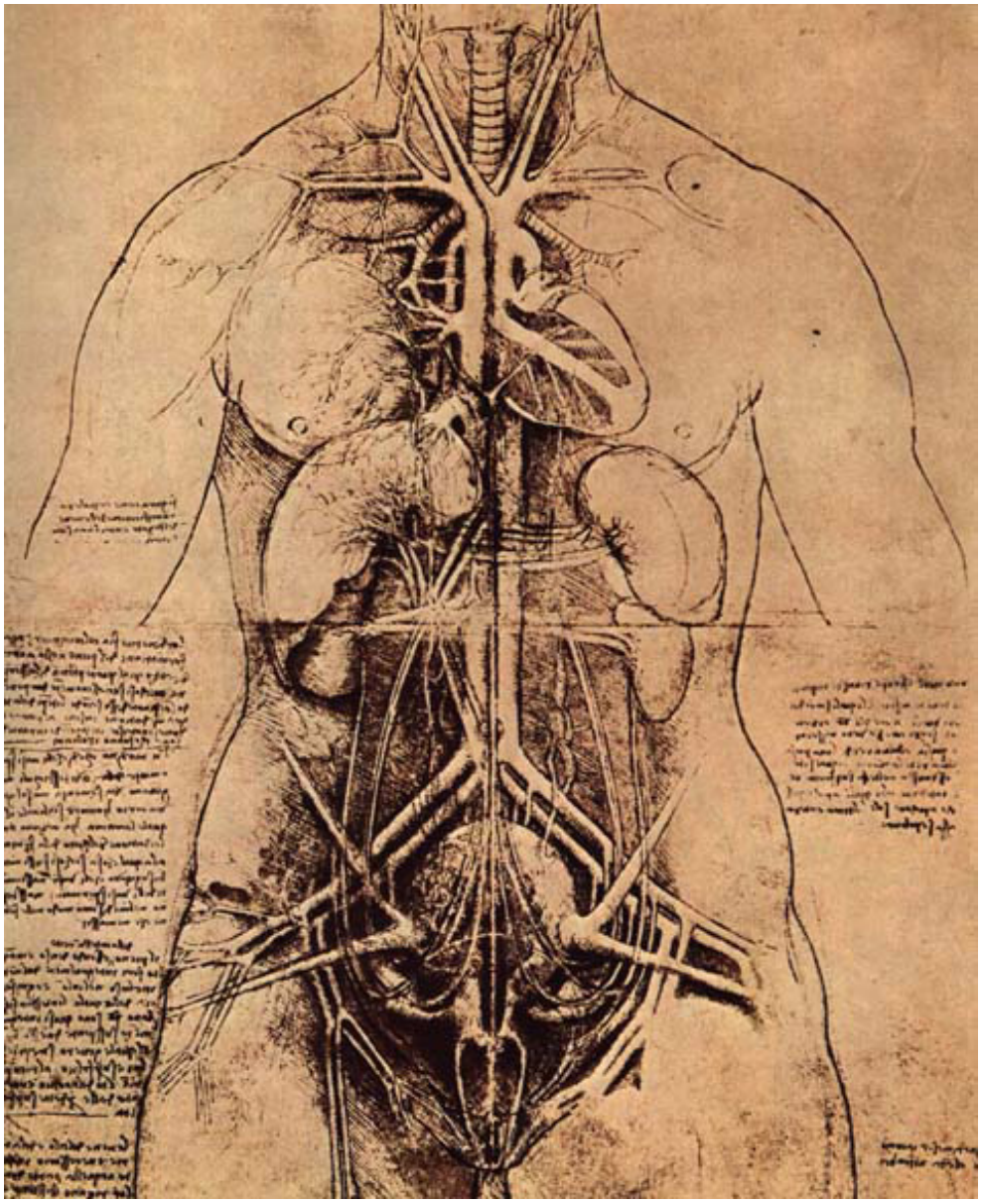


Figure 1.3 – Drawing of a woman's torso by Leonardo da Vinci, from his anatomical notebooks (ca. 1510).

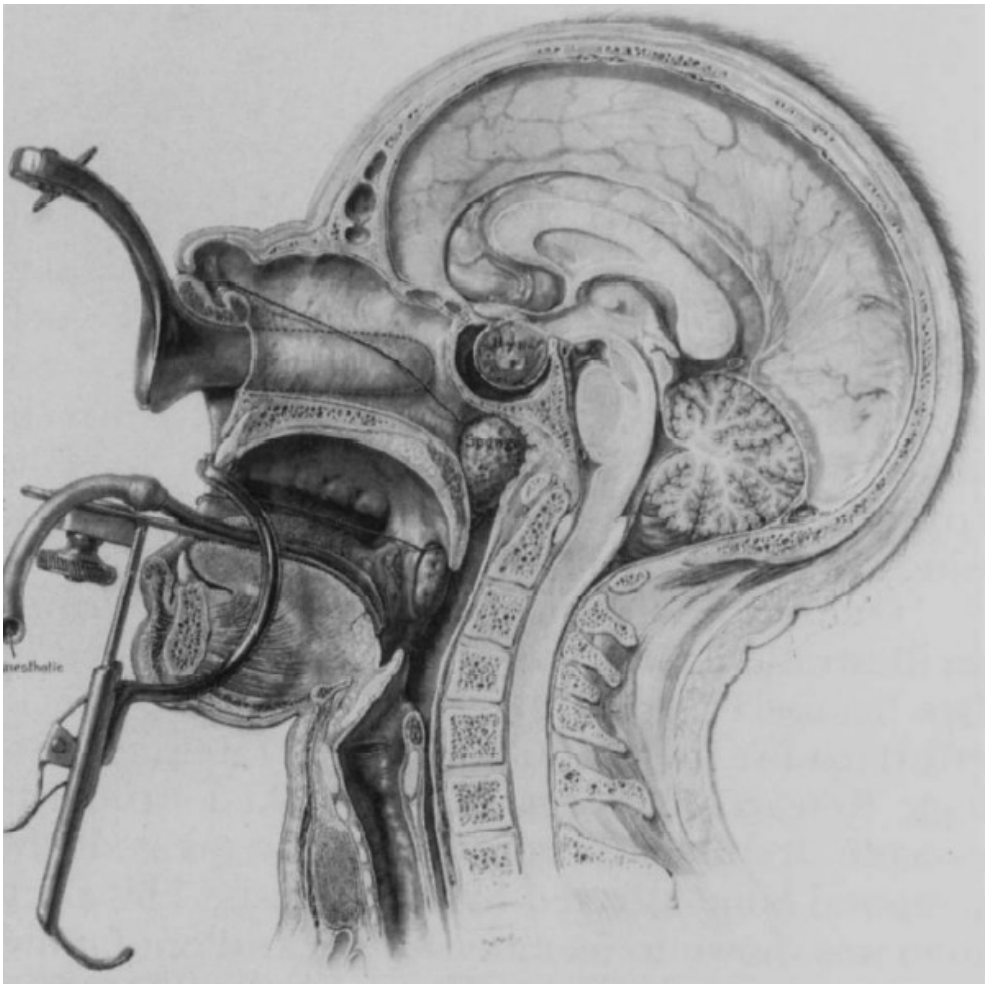
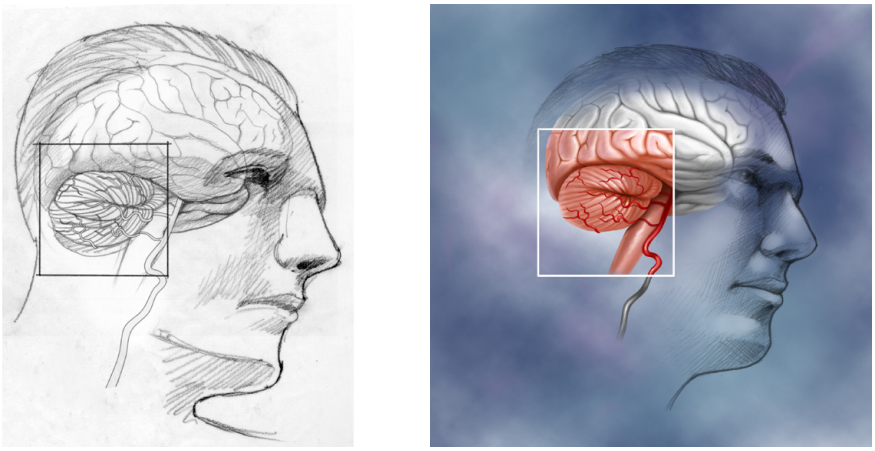


Figure 1.4 – Illustration of a hypophysectomy procedure by Max Brödel.

renderings to have the authenticity of a photograph while still abstracting away unnecessary detail (see Figure 1.4).

Over the course of the 20th century, illustration established itself as a distinct field at the intersection point between art and science through the likes of Frank Netter (1906-1991) whose work is still featured in many text books today. Illustrators not only needed artistic skills but also required in-depth knowledge of the subject matter. With the advent of personal computers in the 1980s, illustrators increasingly began to employ modern desktop publishing software in their workflow. While still essentially a manual process, image processing software enabled the flexible combination of multiple drawing layers as well as digital storage as shown in Figure 1.5. As three-dimensional graphics finally became feasible on desktop computers during the 1990s, illustrators started to combine traditional techniques and computer-assisted



(a) (b)

Figure 1.5 – Modern medical illustration by Mike de la Flor [21]. (a) Initial hand-drawn sketch. (b) Final illustration generated using Adobe Photoshop.

modeling and rendering. While this use of modern technology undoubtedly helped illustration to become more efficient, it still remains a time consuming process which requires careful and detailed modeling of the specimen to be depicted as well as profound knowledge of the subject matter.

1.2 Illustrative Visualization

In 1895, Wilhelm Conrad Röntgen (1845-1923) was studying the phenomena accompanying the passage of electric current through gas of extremely low pressure. He found that invisible rays were emitted from the discharge tube. While working in a darkened room he observed that a paper plate covered on one side with barium platinocyanide became fluorescent when placed in the path of the rays. The discharge tube was enclosed in a sealed black carton to exclude all light. The illuminance only depended on the thickness of interposed objects. Soon after, Roentgen finished his first scientific work on this research including the first radiograph taken from the hand of his wife (see Figure 1.6).

In 1917, the Austrian mathematician Johann Radon (1887-1956) proved that any function is well defined by all line integrals of the function. This purely theoretical approach provided the mathematical foundations for further research towards computed tomography – the process of reconstructing spatial information from a series of projections. Nearly half a century later Allan Cormack (1924-1998) did first experiments with X-ray absorption on phantoms made of material like wood or aluminum. He published his work on calculating the radiation absorption distribution in the human body based on

Figure 1.6 – The first X-ray image showing the hand Wilhelm Conrad Röntgen's wife (1895).

transmission measurements in 1963. In 1968, Godfrey Hounsfield (1919-2004) successfully implemented a prototype of a computed tomography device. For their contributions, Cormack and Hounsfield shared the 1979 Nobel Prize for Physiology or Medicine.

Computed Tomography (CT) is the general process of creating cross-sectional images from projections of the object at multiple angles. If unmodified, the term CT conventionally implies images made by measuring the transmission of X-rays. In X-ray CT, the function imaged is the distribution of linear attenuation coefficients since, for monochromatic X-rays, the logarithm of the transmitted intensity is proportional to the integral of the attenuation coefficient along the path. Other tomographic imaging modalities such as Magnetic Resonance Imaging (MRI), which is based on the spin relaxation properties of excited hydrogen nuclei in water and lipids, have since been developed. The key advantage of tomographic imaging modalities is that they capture volumetric information, i.e., the property measured by the scanner can be reconstructed – at least theoretically – at every point in space. In practice, samples are reconstructed on a regular grid at a pre-determined resolution. A tomographic scan therefore allows the inspection of a specimen's interior in a non-destructive manner.

Initially, volume data sets were analyzed by viewing consecutive slices of the data. However, soon first attempts were made to extract three-dimensional structures from such data sets [49]. While early research on the visualization of volume data focused on reconstructing high-quality surfaces [68], the introduction of direct volume rendering by Drebin et al. [31] and Levoy [66] first demonstrated the advantages of directly rendering from the volumetric representation. Since then, a vast amount of research has been devoted to the area of direct volume visualization. At first, many volume rendering approaches were based on an approximation of a realistic physical model. However, the need to improve the visualization of relevant features in complex volumetric data sets, lead to an increased interest in methods used by illustrators, giving rise to the area of illustrative visualization [33, 86].

1.3 Scope of this Thesis

In recent years, three-dimensional imaging has become a vital tool not only in medical diagnosis and treatment planning, but also in many technical disciplines (e.g., material inspection), biology, and archeology. Modalities such as X-Ray Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) produce high-resolution volumetric scans on a daily basis. Furthermore, initiatives such as the Visible Human Project by the Unites States' National Library of Medicine have succeeded in generating highly detailed reference data sets [100]. It seems counter-intuitive that even though such a wealth of data is available, the production of an illustration should still require

tedious and time-consuming geometric modeling of the specimen from scratch. Instead, the design of the illustration could be performed directly based on the volumetric representation. Volume data, as opposed to surface representations, captures physical properties of a specimen at every point in space and can thus be used as a basis for a wide variety of different illustrations.

In this thesis, we present the concept of an interactive direct volume illustration system, which allows the generation of scientific visualizations with the aesthetic quality and information content of traditional illustrations but is based on volumetric data. Direct volume illustration means that instead of going through an elaborate modeling process, the volumetric representation is used directly as a basis for the design and presentation of an illustration. In addition to the technical challenges involved in rendering volumetric data at interactive frame rates, this work presents the extension of common concepts used in illustration to an interactive illustration system. In particular, visual abstraction, a fundamental cornerstone for the effective presentation of information, is the main focus of this work.

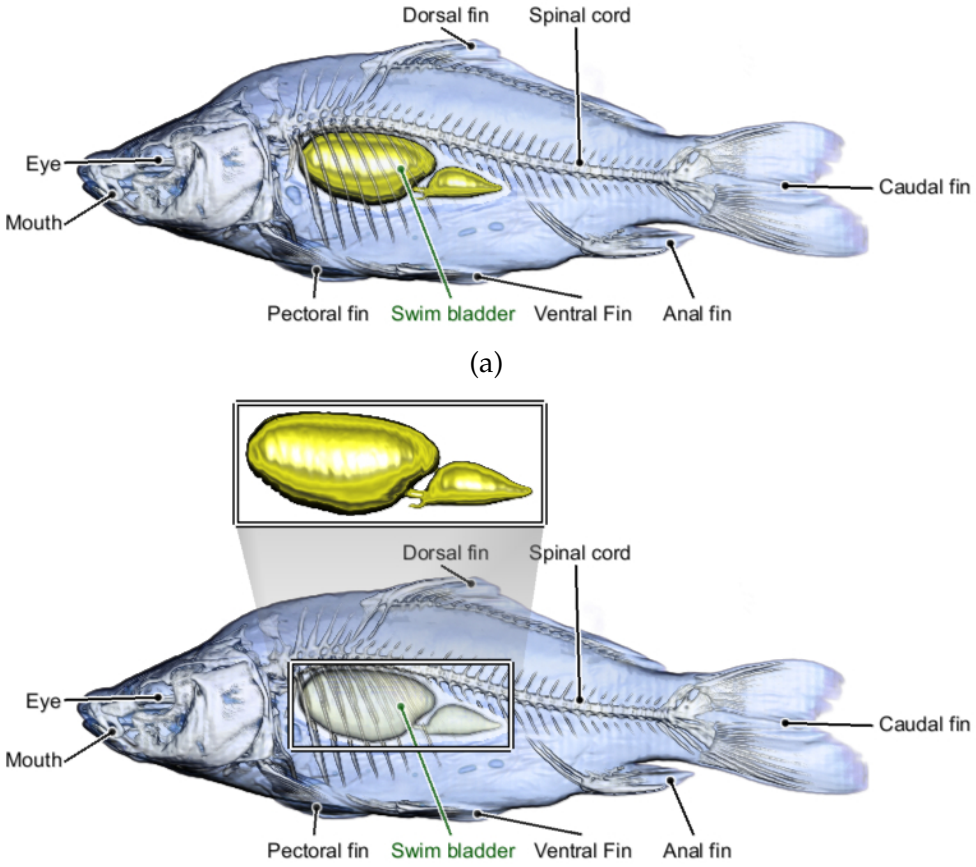


Figure 2.1 – Annotated direct volume illustrations of a carp. (a) The swim bladder is highlighted using a cutaway. (b) The swim bladder is displayed enlarged.

Art is making something out of nothing and selling it.

— *Frank Zappa*

CHAPTER

2 A Direct Volume Illustration System

Although many specimens are readily available as volumetric data sets, illustrations are commonly produced in a time-consuming manual process. Our goal is to create a fully dynamic three-dimensional illustration environment which directly operates on volume data acquired by tomographic imaging modalities. Single images have the aesthetic appeal of traditional illustrations, but can be interactively altered and explored. This chapter describes the basic concepts used in the development of an interactive system for the generation of illustrations based on volumetric data [4].

2.1 Introduction

CONSIDERABLE effort has been devoted to developing, improving and examining direct volume rendering algorithms for visualization of scientific data. It has been shown that volume rendering can be successfully used to explore and analyze volumetric data sets in medicine, biology, engineering, and many other fields. In recent years, non-photorealistic or illustrative methods employed to enhance and emphasize specific features have gained popularity. Although the presented work is based on a large body of research in this area, its focus is somewhat different. Instead of using these techniques to improve the visualization of volume data for common applications such as diagnosis, we want to combine existing and new methods to directly generate illustrations, such as those found in medical textbooks, from volumetric data.

Illustrations are an essential tool in communicating complex relationships and procedures in science and technology. However, the time and effort needed to complete an illustration is considerable and varies widely depending on the experience and speed of the illustrator and the complexity of the content. The more complicated the subject matter, the longer it will take the illustrator to research and solve a complex visual problem. Different illustration methods and styles can also have a significant impact on the time involved in the creation of an illustration. Therefore, illustrators are increasingly using computer technology to solve some of these problems. This,

however, is mostly restricted to combining several manually created parts of an illustration using image processing software.

Volume rendering has gained some attention in the illustration community. For example, Corl et al. [16] describe the use of volume rendering to produce images as reference material for the manual generation of medical illustrations. We aim to take this development one step further. Our goal is to create a fully dynamic three-dimensional volume-based illustration environment where static images have the aesthetic appeal of traditional illustrations. The advantages of such a system are manifold: Firstly, the whole process of creating an illustration is accelerated. Different illustration methods and techniques can be explored interactively, as demonstrated in Figure 2.1. It is easy to change the rendering style of a whole illustration – a process that would otherwise require a complete redrawing. Moreover, the research process is greatly simplified. Provided that the object to be depicted is available as a volumetric data set, it can be displayed with high accuracy. Based on this data, the illustrator can select which features he wants to emphasize or present in a less detailed way. Illustration templates can be stored and reapplied to other data sets. This allows for the fast generation of customized illustrations which depict, for instance, specific pathologies. Finally, the illustration becomes more than a mere image. Interactive illustrations can be designed where a user can select different objects of interest and change the viewpoint. This chapter gives an overview of our approach to the design of such a system.

2.2 Related Work

Several systems for the generation of illustrations using computer graphics have been developed. Dooley and Cohen [29, 30] present a system for the automatic generation of semi-transparent line and surface illustrations from 3D models. Pioneering work by Seligman and Feiner [35, 92, 93] first treated the topic of visibility constraints. Their work on geometrically modeled objects employs cutaways and ghosting to resolve visibility conflicts. Preim et al. [85] present Zoom Illustrator, a semi-interactive tool for illustrating anatomic models. Their approach focuses on the integration of three-dimensional graphics and textual representations. Höhne et al. [52] use segmented volume data to generate an anatomic atlas which allows text-based browsing and interactive cutting operations. Owada et al. [81, 82] present a system for modeling and illustrating volumetric objects. They semi-automatically generate artificial cutting textures based on surface models. Svakhine et al. [104] discuss a volume visualization system which employs illustration motifs to control the appearance of objects at varying degrees of complexity.

2.3 Overview

Illustrations are produced to enable a viewer to extract information. They are images with a communicative intent. For this purpose, they do not just contain straightforward depictions of the data – the presentation is dedicated to a thematic focus. Thus, portions of the data may be represented in more detail or more prominently while others may be simplified, shrunken, or even left out. This distortion of the visualization with respect to the underlying model is referred to as abstraction. The data from which an image is generated can be viewed as a complex information space. The term abstraction denotes the process through which an extract of this information space is refined in order to reflect the importance of the features of the underlying model for the visualization goal at hand [103].

Abstraction is a key component of many traditional illustrations. As there is often not enough space available to display all information in sufficient detail, the general idea is to emphasize regions of particular interest while reducing other information. Portions of the data which are not critical but still important for orientation are retained and depicted in a more stylized manner. These kind of focus+context visualizations are not only motivated by space limitations but also by human visual perception. Humans are capable of simultaneously perceiving both local detail and global structure [94]. Abstraction makes it possible to show more detailed or targeted information in those regions where it is most needed for the intent of the illustration.

VolumeShop[1], our direct volume illustration framework, is different from many previous approaches in that it not only provides an environment for the presentation of a finished illustration, but also attempts to supply interactive tools for the generation of the illustration itself. It is therefore based on incomplete information about the nature of the illustration. This information is supplied by the illustrator through interaction with the system. While the human is kept in the loop to decide which specific abstraction techniques are most suited for the communicative intent of the illustration, the abstraction process itself is performed by the system. Figure 2.2 depicts a conceptual overview of this workflow.

For example, the decision which parts of the data are more important in the context of the illustration ultimately has to be taken by the user through interaction with the system. In order to enable selective abstraction, *VolumeShop* discriminates between two basic types of volumes: data volumes and selection volumes. A data volume stores the measured data, for example acquired by a CT scanner, which forms the basis of the illustration. A selection volume specifies a particular structure of interest in a corresponding data volume. It stores real values in the range [0,1] where zero means "not selected" and one means "fully selected". Such a degree-of-interest function allows for a

[1] `http://www.volumeshop.org`

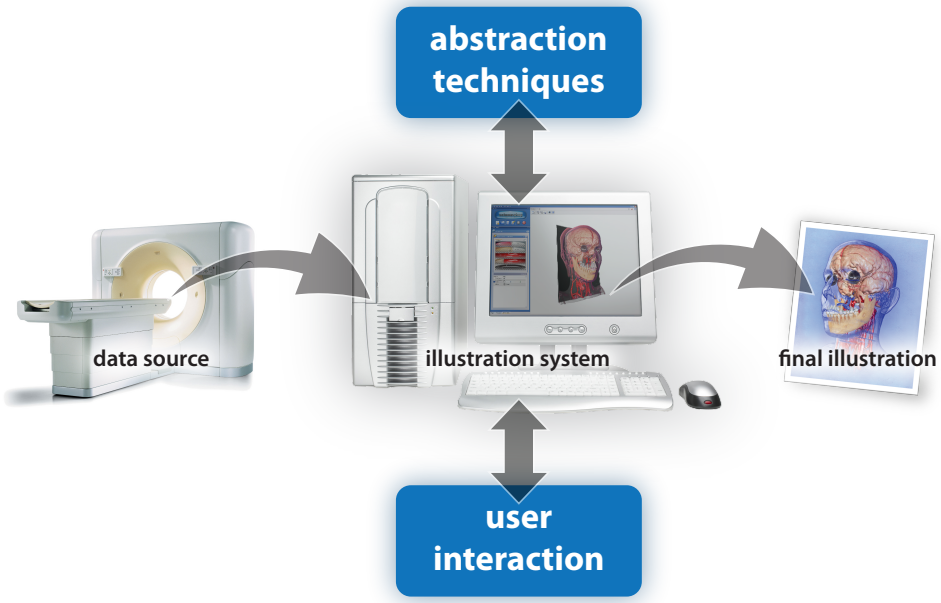


Figure 2.2 – Conceptual overview of *VolumeShop*, our direct volume illustration environment.

more fine-grained control over the level of abstraction compared to a binary classification. Selection volumes can be generated using a wide variety of algorithms collectively known as approaches for volume segmentation. Segmentation, i.e., the identification of individual objects in volumetric data sets, is an area of extensive research, especially in the context of medical applications. Approaches range from very general methods to algorithms specifically developed for particular types of data. In contrast to diagnostic requirements, however, voxel-exact classification of features is not necessarily of primary concern for illustration. Rather, it is important that the illustrator can quickly and easily add and remove structures of interest to and from the selection. Furthermore, as illustrations commonly use smooth transitions between different degrees of abstraction, this fuzzyness should be also supported by the selection definition method. For this reason, one option is to use a three-dimensional volumetric painting approach to define the selection. When the user clicks on the image, a ray is cast from the corresponding position on the image plane into the data volume. At the first non-transparent voxel that is intersected by the ray, a volumetric brush (e.g., a three-dimensional Gaussian) is ”drawn” into the selection volume for each non-transparent voxel within the bounding box of the brush. Different composition methods can be chosen, for example addition (i.e., actual painting) or subtraction (i.e., erasing). Such

an approach has the advantage of providing an interaction metaphor which is familiar to illustrators.

2.3.1 System Architecture

VolumeShop is based on an easily extensible architecture. Various plug-ins for tasks such as data import and export, interaction, rendering, and compositing can be flexibly combined for different illustration tasks and new plug-ins can be created in order to extend the existing functionality. All plug-ins expose their parameters as properties which can be linked together in order to combine their capabilities. Additionally, editor plug-ins can be created on top of this property system to provide high-level user interfaces targeted for specific applications. The system was designed to enable rapid-prototyping on the developer side while still providing the ability to hide complexities from the end-user. Components such as editor plug-ins allow complete customization of the user-interface. Parameterized project setups including plug-in configurations can be serialized to XML files which enables the generation of illustration templates.

Data such as images and volumes are encapsulated as resources. Resources provide a high-level programming interface in the form of iterators and manipulators which hides the underlying representation from the plug-in developer. Volumes are stored in a bricked memory layout using reference counting, i.e., they are subdivided into small cubes which are accessed using an index data structure. Redundant information is not duplicated, thus, if two bricks contain the same data, they are stored in memory only once. The copy-on-write idiom is used for handling modifications. This is most useful for the selection volumes due to their sparse nature. Furthermore, several pieces of meta data (e.g., octrees, coordinate systems, bounding boxes, etc.) are stored for each volume and transparently updated on modification.

This foundation serves as a technological basis for the techniques discussed in this thesis. While the presented concepts and algorithms are general and could be implemented in a different environment, *VolumeShop* was specifically designed to support the requirements of illustrative visualization exploiting the capabilities of current graphics hardware.

2.3.2 Object Model

While both data and selection volumes are treated identical for low-level tasks such as data import and export, the visualization components of the system assign semantics to them. When illustrating a volumetric data set, we want to enable interactive selection and emphasis of specific features. The user should be able to specify a region of interest which can be highlighted and transformed, similar to common image editing applications [65]. We also want to permit arbitrary intersections between objects and control how the in-

tersection regions are visualized. In the following, a conceptual framework for the visualization of multiple volumetric objects in a direct volume illustration framework is described.

The approach identifies three different objects for the interaction with a volumetric data set: a *selection* is a user-defined focus region, the *ghost* corresponds to the selection at its original location, and the *background* is the remaining volumetric object. A transformation T can be applied to the selection, e.g., the user can move, rotate, or scale this object. While the concept of background and selection is used in nearly every graphical user interface, ghosts normally exist, if at all, only implicitly. In the context of illustration, however, such an explicit definition of a ghost object can be advantageous.

We assume a scalar-valued volumetric function f_V and a selection function f_S, which are defined for every point p in space. The selection function f_S has values in $[0, 1]$ which indicate the degree of selection. Based on this degree of selection we define three fuzzy *selection sets* S_S, S_G, and S_B (see Figure 2.3, first row) with their respective membership functions μ_S, μ_G, and μ_B:

$$\begin{aligned} \mu_{S_S}(p) &= f_S(T(p)) \\ \mu_{S_G}(p) &= f_S(p) \\ \mu_{S_B}(p) &= 1 - f_S(p) \end{aligned} \tag{2.1}$$

where T is the transformation that has been applied to the selection.

To control the appearance of our three objects, i.e., selection, ghost, and background, we define color and opacity transfer functions which we denote c_S, α_S, c_G, α_G, and, c_B, α_B. We use the opacity transfer functions to define the membership functions of three *volume sets*, V_S, V_G, and V_B (see Figure 2.3, second row):

$$\begin{aligned} \mu_{V_S}(p) &= \alpha_S(T(p)) \\ \mu_{V_G}(p) &= \alpha_G(p) \\ \mu_{V_B}(p) &= \alpha_B(p) \end{aligned} \tag{2.2}$$

For each of our three objects we can now define an *object set* as the intersection between the corresponding selection and volume set (see Figure 2.3, third row):

$$\begin{aligned} S &= S_S \cap V_S \\ G &= S_G \cap V_G \\ B &= S_B \cap V_B \end{aligned} \tag{2.3}$$

These sets correspond to our basic objects selection, ghost, and background. Thus, in the following we will use these terms to refer to the respective object sets and vice versa. For volume rendering, we now need a way to determine the color and opacity at a point p in space depending on its grade of membership in these sets. We assume n sets $X_1, X_2, \ldots, X_n$ and their corresponding color transfer functions $c_1, c_2, \ldots, c_n$. We can then define the color

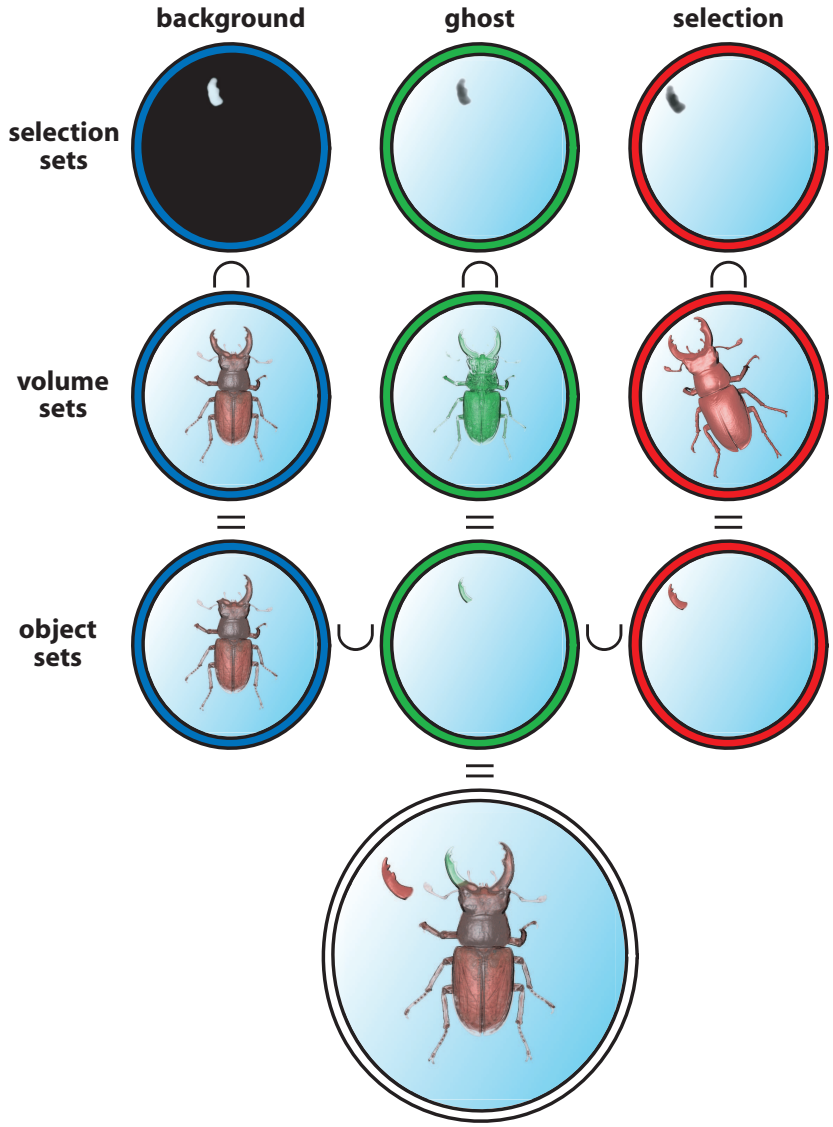


Figure 2.3 – Overview of the basic multi-object combination process for background, ghost, and selection: the intersection between selection sets and volume sets results in object sets which are then combined.

at a point p as a weighted sum using the respective membership functions $\mu_{X_1}, \mu_{X_2}, \ldots, \mu_{X_n}$ as weights:

$$c(p) = \frac{\sum_{i=1}^{n} c_i(p) \cdot \mu_{X_i}(p)}{\sum_{i=1}^{n} \mu_{X_i}(p)} \tag{2.4}$$

As the membership functions of our sets are based on the opacity and the degree of selection, we define the opacity at p as the grade of membership in the union of all sets:

$$\alpha(p) = \mu_{X_1 \cup X_1 \cup \ldots \cup X_n}(p) \tag{2.5}$$

Using Equations 2.4 and 2.5 for our three sets S, G, and B and the color transfer functions c_S, c_G, and c_B leads to a meaningful combination of colors and opacities when used in direct volume rendering. However, we want to provide the user with additional control over the appearance of regions of intersection [10, 42, 113]. Frequently, for example, illustrators emphasize interpenetrating objects when they are important for the intent of the illustration.

To achieve this we first need to identify potential regions of intersection. According to our definitions $B \cap G = \emptyset$, i.e., background and ghost never intersect. The selection, however, can intersect either the background, the ghost, or both. Thus, we direct our attention to the sets $GS = G \cap S$ and $BS = B \cap S$. For every point which is a member of one of these sets, we want to be able to specify its appearance using special intersection transfer functions c_{GS}, c_{BS} for color and α_{GS}, α_{BS} for opacity. Thus, we define two new sets V_{GS} and V_{BS} with the following membership functions:

$$\begin{aligned} \mu_{V_{GS}}(p) &= \alpha_{GS}(f_V(p), f_V(T(p))) \\ \mu_{V_{BS}}(p) &= \alpha_{BS}(f_V(p), f_V(T(p))) \end{aligned} \tag{2.6}$$

The intersection transfer functions are two-dimensional. Their arguments correspond to the value of volumetric function f_V at point p and at $T(p)$, the value of the function at p transformed by the selection transformation T. Based on these two sets, we now define two alternative sets $\widehat{GS}$ and $\widehat{BS}$ for the regions of intersection:

$$\begin{aligned} \mu_{\widehat{GS}}(p) &= \begin{cases} 0 & \mu_{GS}(p) = 0 \\ \mu_{S_G \cap S_S \cap V_{GS}}(p) & \text{otherwise} \end{cases} \\ \mu_{\widehat{BS}}(p) &= \begin{cases} 0 & \mu_{BS}(p) = 0 \\ \mu_{S_B \cap S_S \cap V_{BS}}(p) & \text{otherwise} \end{cases} \end{aligned} \tag{2.7}$$

To evaluate the combined color and opacity at a point p in space, we use Equation 2.4 and 2.5 with the sets $S - (\widehat{GS} \cup \widehat{BS})$, $G - \widehat{GS}$, $B - \widehat{BS}$, $\widehat{GS}$, and $\widehat{BS}$ and the respective color transfer functions c_S, c_G, c_B, c_{GS}, and c_{BS}. We use the standard definitions for fuzzy set operators where the minimum operator

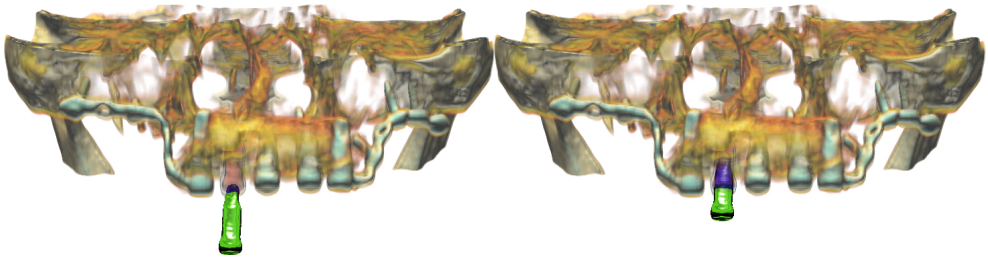


Figure 2.4 – Using intersection transfer functions to illustrate implant placement in the maxilla. As the selection (green) is moved into the ghost (faint red), the intersection transfer function causes it to be displayed in blue.

is used for the intersection and the maximum operator is used for the union of two fuzzy sets [116].

The intersection transfer functions are used to control the appearance in the region of intersection between two objects. The concrete implementation of these intersection transfer functions is dependent on the employed abstraction techniques. It can be derived automatically from the appearance of the respective objects, but it is also possible to allow full user control, for instance, by allowing the user to paint on the two-dimensional function to highlight specific scalar ranges. Figure 2.4 shows an example where the ghost/selection intersection transfer function is used to illustrate the placement of an implant in the maxilla. This kind of emphasis is not only useful for the final illustration, but can act as a kind of implicit visual collision detection during its design.

2.3.3 Abstraction Techniques

Abstraction techniques are the means by which the effect of the abstraction process is achieved. Since there are usually several abstraction techniques which produce similar effects, the designer of an illustration has to select one or a combination of several abstraction techniques. To provide visual access to a structure of interest, for example, the objects occluding it may be removed or relocated, a cutaway view can be used, or a rotation of the object can be employed. This choice is constrained by parameters of the output medium chosen and of human perception. To meet the restrictions of human perception it is essential to establish an equilibrium between the level of detail of the objects of interest and the objects depicting their context so that the user can easily understand the image. On the one hand, it is crucial to reduce the cognitive load for the interpretation of an illustration. On the other hand, enough contextual information must be provided to understand an image. Abstraction techniques can be classified as low- or high-level techniques, based on the kind of modifications they perform [46].

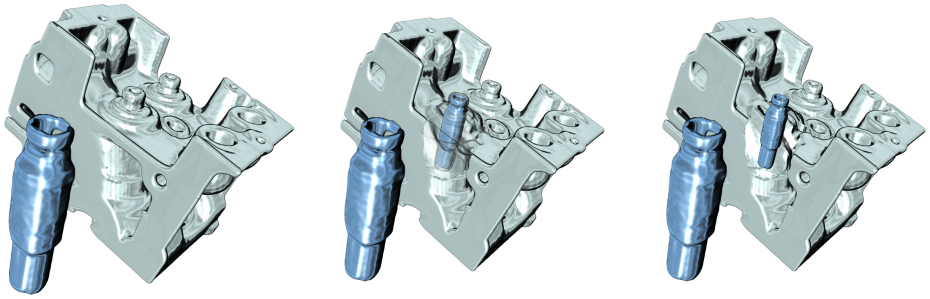


Figure 2.5 – Importance-driven cutaway with varying sparseness of the occluding region.

Low-Level Abstraction Techniques

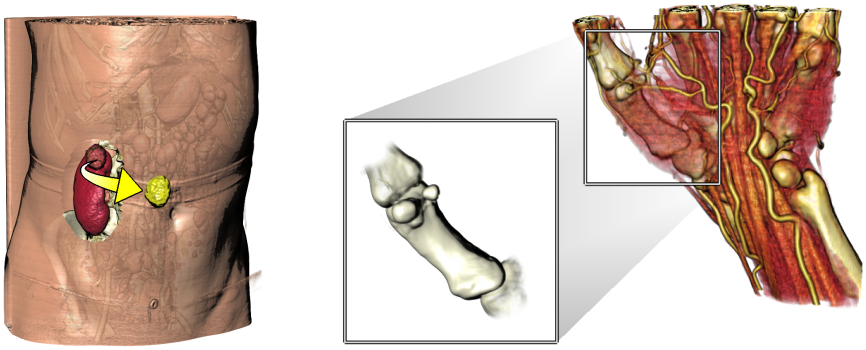
Low-level abstraction techniques deal with *how* objects should be presented. Stylized depictions, such as line drawings, are frequently used to depict context information, while more visually prominent rendering styles are employed in the presentation of important structures. While the area of non-photorealistic rendering has produced many algorithms capable of simulating artistic techniques, an important task in the context of an interactive illustration system is the integration of different rendering styles using a unified representation. In Chapter 3, which is devoted to low-level abstraction techniques, we discuss methods for accomplishing this.

High-Level Abstraction Techniques

High-level abstraction techniques are concerned with *what* should be visible and recognizable. Many of these techniques specifically deal with the problem of occlusion, as spatial relations frequently conflict with the importance of objects. Viola et. al. [107, 108] introduced the concept of importance-driven volume rendering, which generates cutaway views based on an importance specification. If a less important object occludes a more important object, the visual representation of the occluding region becomes more sparse (see Figure 2.5). Further high-level abstraction techniques are presented in Chapter 4.

Annotations

In addition to the abstraction of the data itself, it is often useful to clearly point out which parts of the object have been abstracted. Annotations are used for this purpose. Illustrators commonly employ certain visual conventions to indicate the type of abstraction technique that has been applied. Arrows, for instance, normally suggest that an object actually has been moved during the illustrated process (e.g., in the context of a surgical procedure) or that an object needs to be inserted at a certain location (e.g., in assembly instructions).



(a) (b)

Figure 2.6 – Using different artistic visual conventions. (a) Illustration of a tumor resection procedure. (b) Detailed depiction of a hand bone using a blow-up.

Similarly, enlarged or alternative depictions of certain structures are frequently indicated by a a connected pair of shapes, such as rectangles or circles. These annotations can be generated by collecting data about the abstraction process itself and using an appropriate visualization technique to display them. Figure 2.6 shows two examples for the use of annotations in our system.

Furthermore, hand-made illustrations in scientific and technical textbooks commonly use labels or legends to establish a co-referential relation between pictorial elements and textual expressions. As we allow multiple selections to be defined, labels can be employed to display additional information such as a natural language description of the corresponding objects. They not only recreate the appearance of static illustrations but are also useful for simplifying orientation in our interactive environment. Ali et al. [1] give a comprehensive description of label layout styles. A simple algorithm for visually pleasing label layout can be derived from the following guidelines:

- Labels must not overlap.
- Connecting lines between labels and anchor points must not cross.
- Labels should not occlude any other structures.
- A label should be placed as close as possible to its anchor point.

In many textbook illustrations, labels are placed along the silhouette of an object to prevent occlusions. We can approximate this by extracting the convex hull of the projections of the bounding volumes of all visible objects. The resulting polygon is radially parameterized. Based on its location in parametric space, a label is always placed in such a way that it remains outside the convex hull. All labels are initially placed at the position along the

Figure 2.7 – Annotated illustration of a human foot - the current selection is highlighted.

silhouette polygon which is closest to their respective anchor point. We then use a simple iterative algorithm which consists of the following steps:

1. If the connection lines of any two labels intersect, exchange their positions.

2. If exchanging the positions of two labels brings both closer to their anchor points, exchange their positions.

3. If a label overlaps its predecessor, move it by a small delta.

These three steps are executed until either all intersections and overlaps are resolved or the maximum number of iterations has been reached. Remaining intersections and overlaps can be handled by disabling labels based on priority, which can be defined by the screen-space depth of the anchor point, i.e., labels whose reference structures are farther away will be disabled first. While this basic algorithm does not result in an optimal placement, it is very fast for a practical number of labels and generally leads to a visually pleasing layout. Due to the initialization of label locations at positions on the silhouette closest to the anchor points, the labels generally move smoothly in animated

views. Discontinuities only occur when intersections and overlaps need to be resolved. Figure 2.7 shows a labeled illustration of a human foot highlighting the current selection.

2.4 Summary

In this chapter, we introduced the general concept of a direct volume illustration environment. *VolumeShop*, an interactive system for the generation of high-quality illustrations, employs an intuitive object model for interaction with volumetric data. This model is used by low-level and high-level abstraction techniques to create interactive illustrations based on the user's communicative intent. Information about the abstraction process itself can be conveyed using annotations. The following chapters will give a detailed overview of different abstraction techniques based on a volumetric representation.

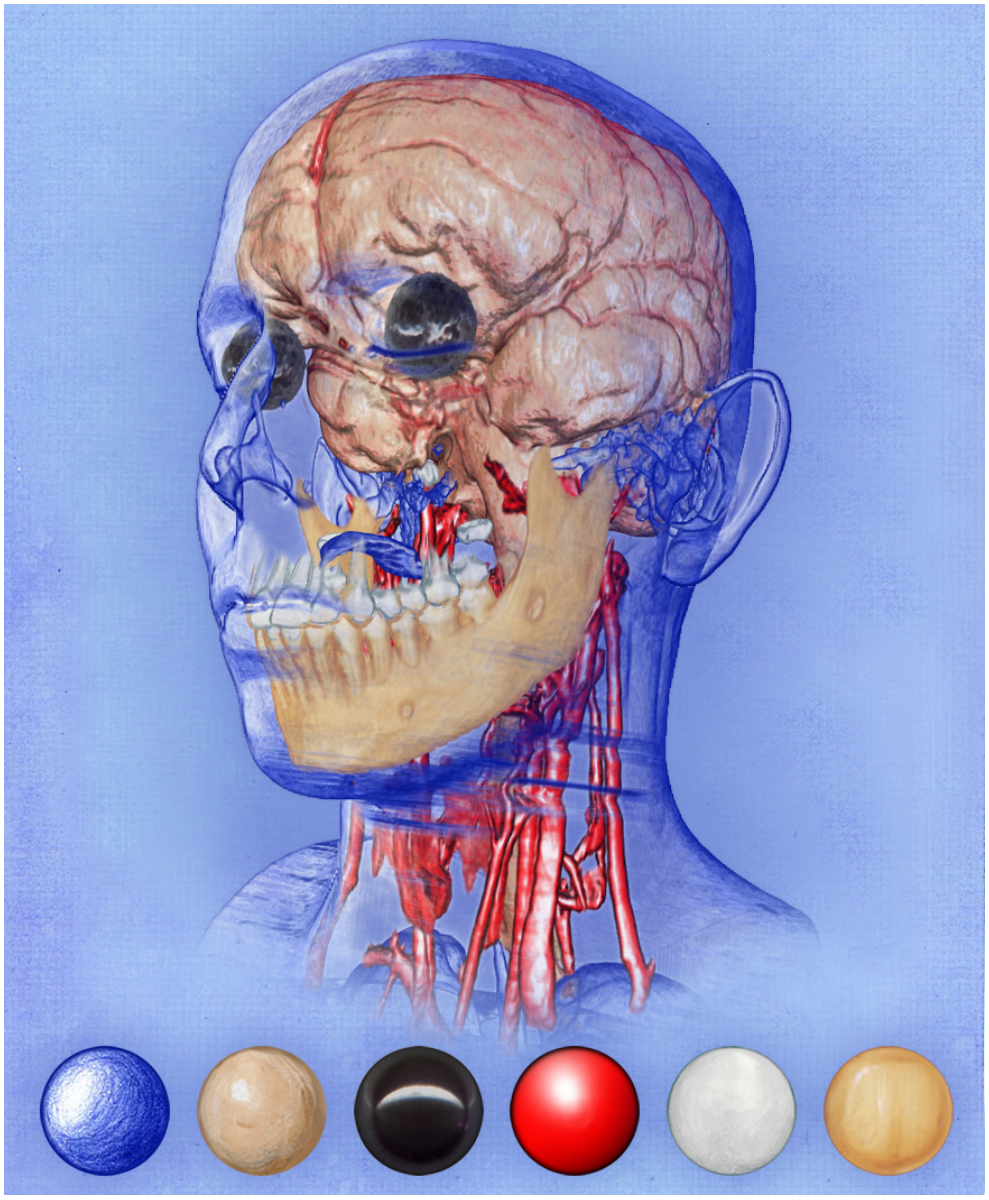


Figure 3.1 – Rendering of segmented volume data using a multi-dimensional style transfer function based on data value and object membership.

There is no abstract art. You must always start with something. Afterwards you can remove all traces of reality.

— *Pablo Picasso*

CHAPTER

3 Low-Level Abstraction Techniques

Stylization can be an effective tool in guiding a viewer's attention to certain features. Illustrations commonly modulate properties such as shading, contours, or shadows to put subtle emphasis on important structures or to reduce the prominence of contextual objects. In order to allow the same kind of flexibility in a direct volume illustration system, methods have to be developed which are able to generate these effects using a sample-based representation. This chapter presents common low-level abstraction techniques adapted to volumetric data [4, 6, 7].

3.1 Introduction

LOW-LEVEL abstraction techniques attempt to adjust the appearance of objects in order to highlight important structures or to de-emphasize less relevant information. Historically, most volume rendering techniques are based on an approximation of a realistic physical model. It was noticed, however, that traditional depictions of the same types of data – as found in medical textbooks, for example – deliberately use non-realistic techniques in order to focus the viewer's attention to important aspects [33, 86]. The illustration community has century-long experience in adapting their techniques to human perceptual needs in order to generate an effective depiction which conveys the desired message. Thus, their methods can provide us with important insights into visualization problems.

In this chapter we discuss two techniques frequently found in traditional illustrations and how they can be integrated in an illustration system based on volume data. First, we focus on the aspect of shading. Illustrations commonly employ different shading styles ranging from realistic depictions of material properties to highly stylized representations. While many algorithms that simulate particular artistic techniques have been developed, many of these methods require tedious tuning of various parameters to achieve the desired result. We aim to circumvent this issue by presenting the user with a gallery of styles extracted from actual illustrations.

The second low-level abstraction technique presented in this chapter is based on the fact that volumetric data commonly has high depth complexity

which makes it difficult to judge spatial relationships accurately. Artists and illustrators frequently employ halos to visually detach an object from its background. In this technique, regions surrounding the edges of certain structures are darkened or brightened which makes it easier to judge occlusion. Based on this idea, a flexible method for enhancing and highlighting structures is presented which can achieve effects such as soft shadowing and glow.

3.2 Related Work

In computer graphics, many techniques have been developed to capture lighting effects in order to plausibly embed objects in photographs or video or to create new scenes under the same environmental conditions [22, 23, 91]. For non-photorealistic rendering, approaches have been presented to reproduce numerous artistic techniques, such as tone shading [39], pencil drawing [98], hatching [84], or ink drawing [99]. While these are specialized algorithms which aim to accurately simulate a particular technique, Sloan et al. [97] employ a simple method to approximately capture non-photorealistic shading from existing artwork. Their approach forms one building block of our approach for stylized shading.

In the context of volume visualization, the combination of different rendering styles is of particular interest, as it allows to put emphasis on features of interest. Lu et al. [71, 72] developed a volume rendering system that simulates traditional stipple drawing. Nagy et al. [77] combine line drawings and direct volume rendering techniques. Yuan and Chen [115] enhance surfaces in volume rendered images with silhouettes, ridge and valleys lines, and hatching strokes. Tietjen et al. [106] use a combination of illustrative surface and volume rendering for visualization in surgery education and planning. Salah et al. [90] employ point-based rendering for non-photorealistic depiction of segmented volume data. Techniques by Lu and Ebert [70] as well as Dong and Clapworthy [28] employ texture synthesis to apply different styles to volume data. Their approaches, however, do not deal with shading.

Multi-dimensional transfer functions have been proposed to extend the classification space and to allow better selection of features. Kniss et al. [58, 59] use a two-dimensional transfer function based on scalar value and gradient magnitude to effectively extract specific material boundaries and convey subtle surface properties. Hladůvka et al. [50] propose the concept of curvature-based transfer functions. Kindlmann et al. [57] employ curvature information to achieve illustrative effects, such as ridge and valley enhancement. Lum and Ma [74] assign colors and opacities as well as parameters of the illumination model through a transfer function lookup. They apply a two-dimensional transfer function to emphasize material boundaries using illumination.

One way to add depth cues to volume rendered images is to use a more realistic illumination model. Yagel et al. [114] employ recursive ray-tracing

which allows for effects such as specular reflection and shadows. Behrens and Ratering [3] add shadows to texture-based volume rendering. The model presented by Kniss et al. [60] captures volumetric light attenuation effects including volumetric shadows, phase functions, forward scattering, and chromatic attenuation. Max [75] gives a comprehensive overview of different optical models for volume rendering. The problem of increasing the physical realism is, however, that these models often lack control over the specific appearance of certain structures of interest. As they are based on actual physical laws, it is difficult to control individual visualization properties separately. Some approaches therefore use inconsistent illumination. Stewart [101] introduces vicinity shading, a view-independent model to enhance perception of volume data based on occlusions in the local vicinity of a sample point resulting in shadows in depressions and crevices. Lee at al. [64] present a system for automatically generating inconsistent lighting based on the object geometry. Kersten et al. [55] study the effect of different depth cues on the perception of translucent volumes.

Halos and similar techniques have been used by numerous researchers to enhance depth perception. As an early example, Appel et al. [2] proposed an algorithm for generating haloed lines in 1979. Interrante and Grosch [53] employ halos to improve the visualization of 3D flow. Their approach uses line integral convolution of a texture of slightly enlarged noise spots to compute a halo volume which is then used during ray casting. Wenger et al. [112] use similar techniques for volume rendering of thin thread structures. Rheingans and Ebert [86] present feature halos for scalar volume visualization. Their approach computes an additional halo volume based on properties of the original data values. Svakhine and Ebert [105] extend this method for GPU-based volume rendering by computing the halo volume on the graphics hardware. Loviscach [69] presents a GPU-based implementation of halos for polygonal models. Ritter et al. [87] encode spatial distance in halo-like non-photorealistic shadows for the visualization of vascular structures. The approach of Luft et al. [73] is capable of enhancing surface-based images using halos by performing an unsharp masking operation on the depth buffer.

3.3 Stylized Shading

The goal of stylized shading is to visually enhance important features or to de-emphasize unwanted details by using non-photorealistic shading techniques. However, it is difficult to integrate multiple non-photorealistic rendering approaches into a single framework due to great differences in the individual methods and their parameters. In this section, we discuss techniques to integrate illustrative rendering styles into a direct volume illustration system using the concept of style transfer functions. This approach enables flexible data-driven illumination which goes beyond using the transfer function to just

assign colors and opacities. An image-based lighting model uses sphere maps to represent non-photorealistic rendering styles which can be extracted from existing artwork. Style transfer functions allow us to combine a multitude of different shading styles in a single rendering. The basic concept is extended with a technique for curvature-controlled style contours and an illustrative transparency model. The presented method allows interactive generation of high-quality volumetric illustrations.

3.3.1 Style Representations

Most illumination models use information about the angle between normal n, light vector l and view vector v to determine the lighting intensity. In volume rendering, the directional derivative of the volumetric function, the gradient, is commonly used to approximate the surface normal. Additionally, the gradient magnitude is used to characterize the "surfaceness" of a point; high gradient magnitudes correspond to surface-like structures while low gradient magnitudes identify rather homogeneous regions. Many distinct approaches have been presented that use these quantities in different combinations to achieve a wide variety of effects.

As a sufficiently flexible illumination model requires numerous parameters, a common problem in the integration of multiple rendering styles into a single framework is the selection of a style representation. A good style representation should be compact, i.e., it should capture the essence of an object's shading in a self-contained and intuitive manner. Additionally, it should be easy to transfer, for instance, through extraction from an existing piece of artwork. Finally, such a representation should also allow efficient rendering on current graphics hardware in order to permit interactivity.

Lighting Maps

A straight-forward candidate for a visual style representation is a simple two-dimensional function we will refer to as lighting map. The arguments of this function are the dot product between the normal n and the light vector l and the dot product between the normal n and the half-way vector h, where h is the normalized sum of l and the view vector v. A two-dimensional lookup table stores the ambient, diffuse, and specular lighting contributions for every $n \cdot l$ and $n \cdot h$ pair.

It is straight-forward to use this kind of lighting map for common Blinn-Phong lighting. However, many other models can also be specified in this way and evaluated at constant costs. We use the terms "ambient", "diffuse", and "specular" to illustrate the simple correspondence in case of Blinn-Phong lighting. However, the semantics of these components are defined by the model used for generation of the lighting map. Essentially, "ambient" means a contribution in environment color, "specular" specifies the contribution in

light color, and "diffuse" corresponds to the contribution in object color. Thus, a lighting map might use these terms to achieve effects completely unrelated to ambient, diffuse, and specular lighting.

For example, contour lines are commonly generated by using a dark color where the dot product between normal and view vector $n \cdot v$ approaches zero, i.e., these two vectors are nearly orthogonal. If we have $n \cdot l$ and $n \cdot h$ with $h = \frac{1}{2}(l+v)$, then $n \cdot v = 2(n \cdot h) - n \cdot l$. We can thus create a lighting map where we set ambient, diffuse and specular components to zero where $n \cdot l \approx 2(n \cdot h)$. One advantage of this approach is that artifacts normally introduced by using a threshold to identify contour lines can be remedied by smoothing them in the lighting map with no additional costs during rendering. Other methods, such as cartoon shading [15] or metal shading [39] can be realized straightforwardly and combined with effects like contour enhancement. Figure 3.2 shows an image rendered using four different lighting maps.

While this approach captures some important characteristics of common illumination models, its flexibility is limited due to the fact that it is still based on the traditional notions of light vector and view vector. Furthermore, color variations can only be introduced through predefined parameters such as object color and light color. This not only means that more complex color transitions are not possible, but it also requires the specification of these extra parameters in addition to the lighting map, i.e., it is not a self-contained representation.

Lit Sphere Shading

The deficiencies of lighting maps suggest that artistic rendering styles frequently do not clearly differentiate between luminance and chromaticity, an observation employed in the tone-shading approach of Gooch et al. [39]. Sloan et al. [97] presented a simple yet effective method for representing artistic shading which incorporates color information. They describe an approach for capturing artistic lighting by using an image of a sphere shaded in the desired style. The basic idea is to capture color variations of an object as a function of normal direction. As a sphere provides coverage of the complete set of unit normals, an image of a sphere under orthographic projection will capture all such variations on one hemisphere (see Figure 3.3). This image is then used as a sphere map indexed by the eye space normals to shade another object. Essentially, the sphere acts as a proxy object for the illumination. In their work, Sloan et al. also describe a method for extracting lit sphere maps from non-spherical regions in a piece of artwork. They present an interactive tool which allows rapid extraction of shading styles from existing images.

The lit sphere map itself is a square texture where texels outside an inscribed disk are never accessed. Normal vectors parallel to the viewing direction map to the center of the disk and normal vectors orthogonal to the viewing direction map to the circumference of the disk. The lit sphere map is

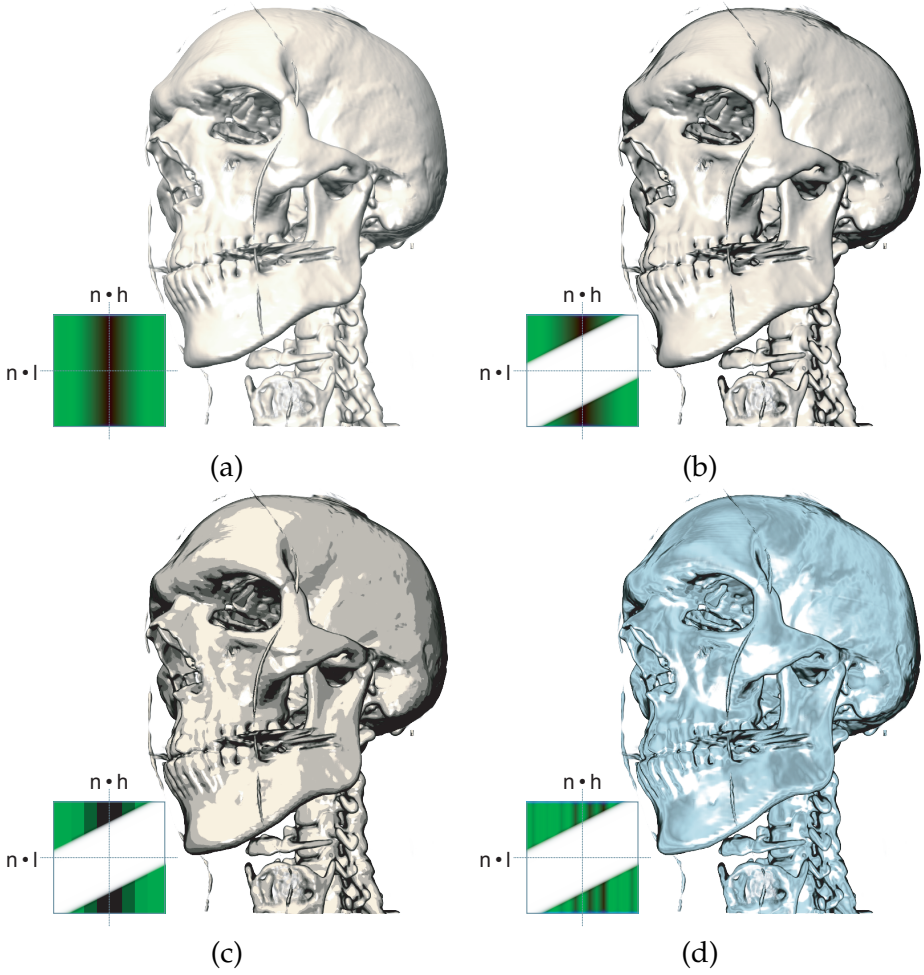


Figure 3.2 – The same data set rendered with four different lighting maps. The rgb-encoded lighting map for each image is displayed in the lower left corner. (a) Standard Blinn-Phong lighting. (b) Blinn-Phong lighting with contour enhancement. (c) Cartoon shading with contour enhancement. (d) Metal shading with contour enhancement.

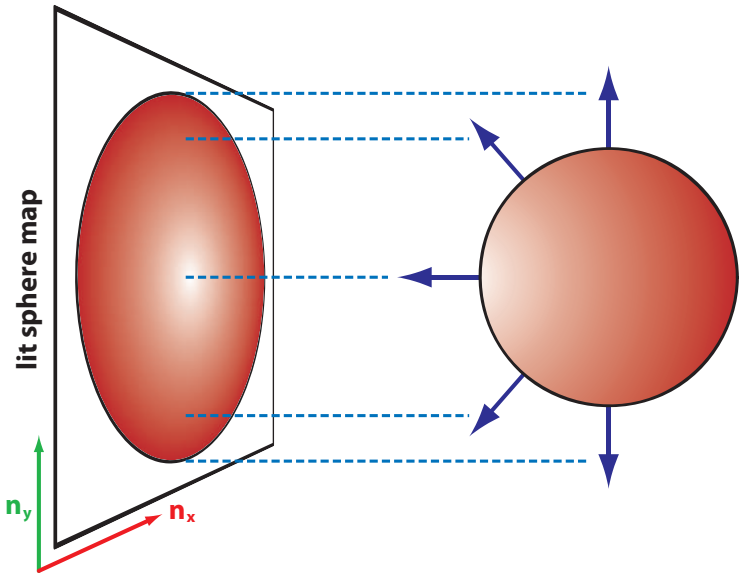


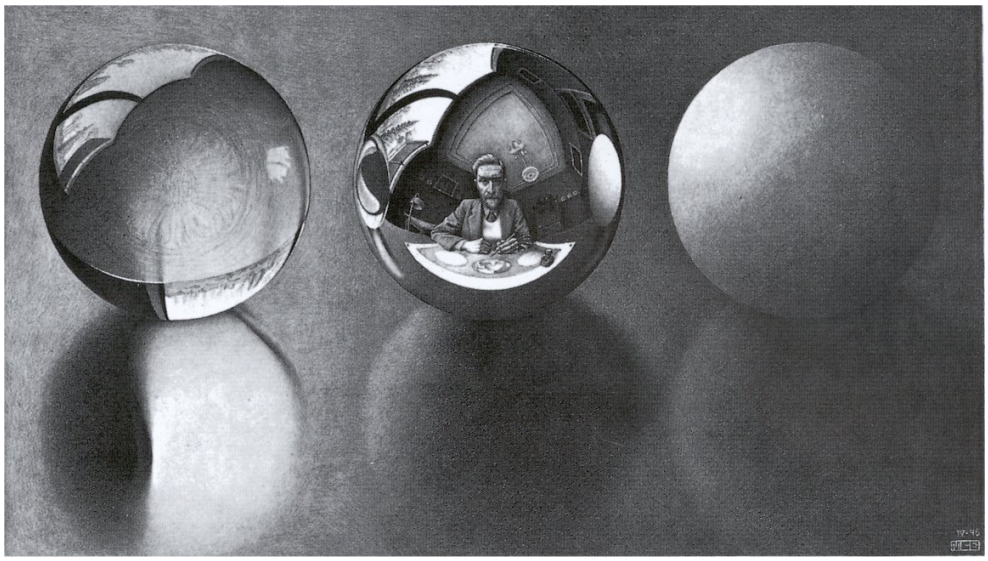
Figure 3.3 – Lit sphere shading. The shading of an object is represented as a function of eye space normal orientation.

indexed by simply converting the n_x and n_y components of the eye space normal $n = (n_x, n_y, n_z)$ which are in the range $[-1..1]$ to texture coordinate range (usually [0..1]). As the n_z component is ignored, lighting does not distinguish between front and back faces. This is desired as the gradient direction in the volume which serves as the normal might be flipped depending on the data values at a material boundary.

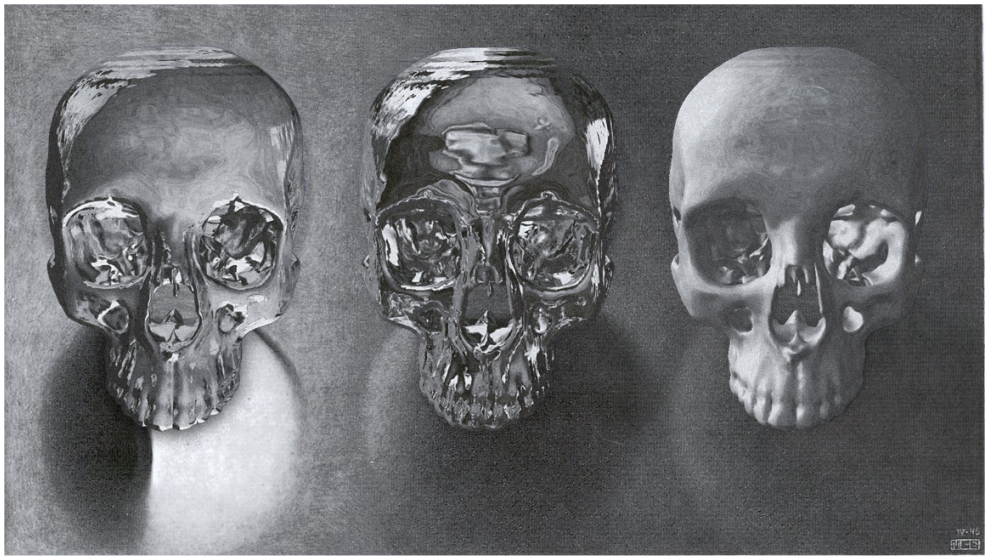
While lit sphere shading fails to capture complex aspects of realistic illumination, it is well-suited to represent the general shading style of an object. Images of an illuminated sphere are relatively easy to obtain as illustrators, for example, frequently perform lighting studies on spheres. Additionally, the extraction process described by Sloan et al. allows to build up a large database of styles with little effort. Another advantage is the view-dependency of this technique. All lighting effects will appear as if the light source was a headlight, i.e., as if it were rotating with the camera. Generally, this is the desired setup in volume visualization. For these reasons, lit sphere maps are a good choice as a basic style representation. Figure 3.4 depicts an example which uses spheres in an existing painting to shade other objects.

3.3.2 Style Transfer Functions

A basic style representation, such as lit sphere maps, allows us to shade objects in a specified manner. In volume visualization, however, discrete object information is rarely available. Instead, reconstructed properties of the volumetric function are used to define the visual appearance of a sample



(a)



(b)

Figure 3.4 – Using lit sphere maps from existing artwork. (a) *Three Spheres II* (1946) by Dutch artist M. C. Escher. (b) Direct volume renderings of a human skull using the respective spheres as style, Escher's painting is used as background.

point. Therefore, in order to be useful for the visualization of volumetric data, we need a continuous parametrization of our style space.

We assume a continuous volumetric scalar field $f(p)$ and its vector of first partial derivatives, the gradient, $g(p) = \nabla f(p)$. For simplicity of notation, if the position in question is unambiguous, f and g will be used to denote samples of the respective function at the current sample position. For the purpose of shading, the normalized gradient vector $\frac{g}{|g|}$ serves as the normal n. Unless specified otherwise, n will refer to the eye space normal, i.e., it has been transformed from object space to eye space. Conventionally, a transfer function assigns color and opacity to each scalar value. There are approaches that use multi-dimensional transfer functions which employ derivatives of the volumetric function, such as the gradient magnitude or the curvature magnitudes. For simplicity we will restrict our discussion to one-dimensional transfer functions at this point. Our technique equally applies to multi-dimensional transfer functions (see Section 3.3.5 for a discussion of this matter).

During rendering, at each sample point the data value and the gradient are reconstructed. The transfer function defines the color and opacity contribution of this sample, while the gradient is used to compute the illumination. The illumination model and its parameters are usually fixed, i.e., they are not dependent on the transfer function. Lum et al. [74] presented an approach were the parameters of the Phong illumination model are specified by an additional lighting transfer function. We extend this idea of data-dependent lighting to enable a wide variety of non-photorealistic shading styles. In our approach, we integrate color and shading information in a combined style transfer function. Mathematically, this is equivalent to extending the transfer function domain by including normal direction. A one-dimensional transfer function based on the scalar value becomes three-dimensional, a two-dimensional transfer function becomes four-dimensional, etc.

Transfer functions are usually implemented as lookup tables. The memory requirements for storing a complete style transfer function lookup table would be prohibitively high due to the increase in dimensionality. However, as there is only a discrete number of styles it is not necessary to store the whole function. We can store the set of styles separately. The transfer function lookup table now contains references to these styles instead of colors. The only restriction necessary is that when interpolating between two styles, the interpolation is performed uniformly for all normal directions, i.e., transitions only occur between whole styles. If a non-uniform transition is desired, this can easily be accomplished by adding one or multiple intermediate styles. Conceptually, this can be illustrated by replacing the single color of a transfer function entry by a lit sphere map (see Figure 3.5). When performing a style transfer function lookup, styles are first interpolated according to the specified transfer function. Using this interpolated style, the eye space normal direction then determines the final sample color.

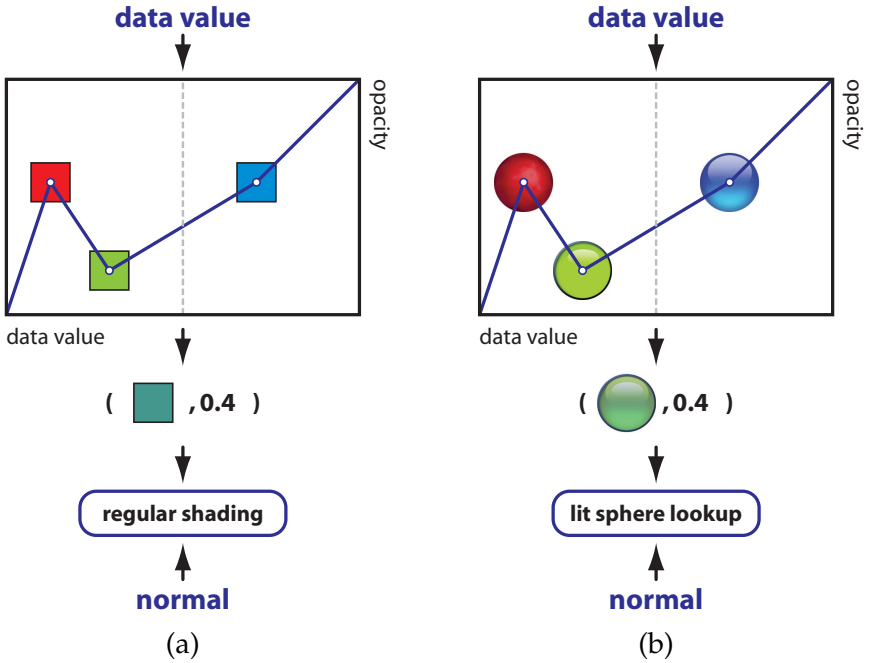


Figure 3.5 – Basic concept of style transfer functions. (a) Regular transfer function. (b) Style transfer function.

From a user's point of view, the transfer function now not only specifies the color over the range of data values, but also the shading as a function of eye space normal direction. The complexity of specifying a transfer function, however, is not increased. Instead of assigning a single color to a certain value range, a pre-defined shading style represented by a lit sphere map is chosen. In this context, one advantage of sphere maps as opposed to other mappings is that they can be directly presented to the user as an intuitive preview image for the style.

Style transfer functions allow for a flexible combination of different shading styles in a single transfer function. Unshaded volume rendering (a constant color sphere), tone shading, cartoon shading, metallic shading, painterly rendering, and many other styles can be used in the same rendering. Style transfer functions also enable inconsistent lighting of different structures in a single data set as a means of accentuating features [63]. Figure 3.6 shows examples of different styles applied to a data set.

3.3.3 Style Contours

Illustrators frequently employ contours to enhance the depiction of objects. Contours help to clearly delineate object shape and resolve ambiguities due to occlusion by emphasizing the transition between front-facing and back-

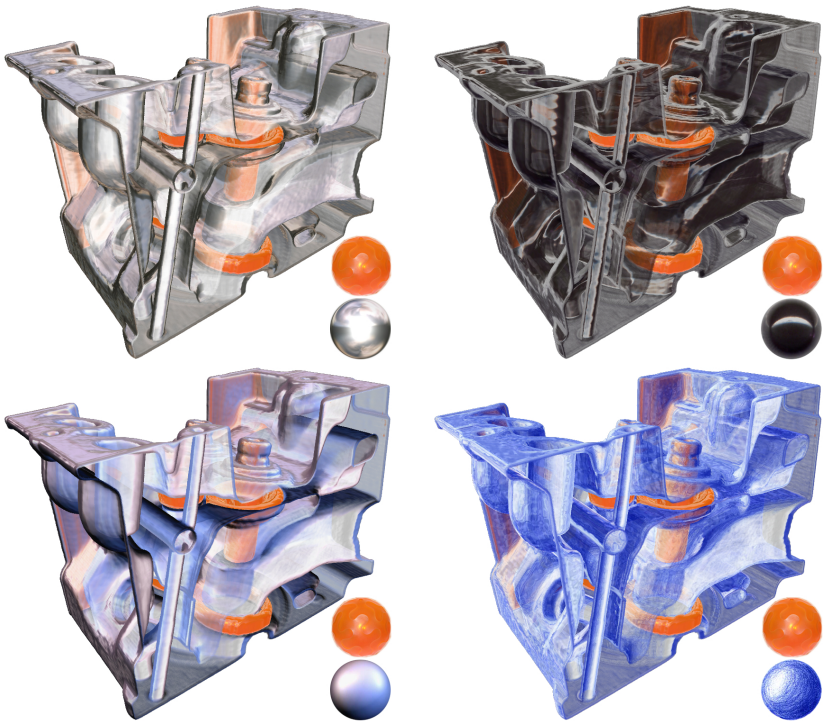


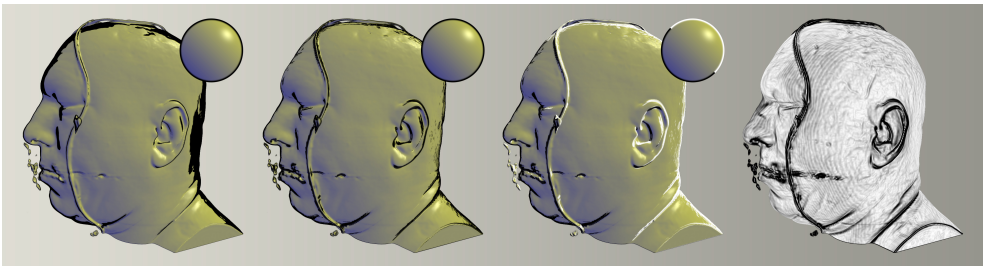
Figure 3.6 – Engine block rendered using different style transfer functions. The lit sphere maps used in the transfer function are depicted in the bottom right corner of each image.

facing surface locations [89]. In volume rendering, contours are generally produced using the dot product between the view vector v and the normal n. The sample color is darkened if v is approximately orthogonal to n, i.e., $v \cdot n$ is close to zero. The drawback of this method is an uncontrolled variation in the apparent contour thickness. Where the surface is nearly flat, a large region of surface normals is nearly perpendicular to the view vector, making the contours too thick. Conversely, in fine structures, where the emphasis provided by contours could be especially helpful, they appear to be too thin. These deficiencies are illustrated in Figure 3.7 (a).

To remedy this problem, Kindlmann et al. [57] proposed to regulate contours based on κ_v, the normal curvature along the view direction. A sample is defined to be on a contour if the following condition is true:

$$|n \cdot v| \leq \sqrt{T\kappa_v(2 - T\kappa_v)} \tag{3.1}$$

where T is a user-defined thickness value. While this method is effective in depicting contours of constant thickness, it requires the expensive reconstruction of second-order derivatives of the volumetric function. Specifically, the curvature measure κ_v is based on the geometry tensor. The geometry tensor



(a) (b) (c) (d)

Figure 3.7 – Style contours. (a) Contours without curvature-controlled thickness. (b) Curvature-controlled contours with constant color. (c) Curvature-controlled contours with varying colors. (d) Our curvature measure (darker regions correspond to higher curvature).

is constructed from the Hessian matrix. Computing the geometry tensor in a fragment program is very expensive and would not allow for interactive performance. On the other hand, pre-computation would require two additional 3D textures (the geometry tensor is symmetric and can be stored in six values per voxel). Hadwiger et al. [45] circumvent this problem by restricting themselves to iso-surfaces, but for direct volume rendering no viable solution has been presented so far.

We propose a simple approximation of κ_v which can be computed efficiently and therefore allows for interactive performance. We are interested in the rate of change in normal direction of the iso-surface corresponding to the current sample value along the viewing direction. When performing volume ray casting, we step along the ray direction and evaluate the normal at every sample point. The angle between two subsequent normals along the ray taken at a sufficiently small distance gives us information about the curvature along the viewing direction (see Figure 3.8). When performing ray casting, we can therefore use the angle between the normal at the current sample point and the previous normal divided by the step size as an estimate for κ_v. This is of course not accurate, as we are not stepping along the iso-surface. However, due to the finite resolution of the volume this coarse approximation has proven to be sufficient for our purposes. The advantage of this approach is that it introduces almost no additional costs as the normal is evaluated at every sample point anyway.

Since we now have a measure for the curvature along the viewing direction, we can employ the criterion proposed by Kindlmann et al. [57] to determine whether a sample is located on a contour (Equation 3.1). Using a fixed contour color, however, would be potentially inconsistent with the selected styles. Instead, the contour color should be determined by the style transfer function. For this reason we adjust the coordinates for the lit sphere map lookup based on our curvature measure: if a sample falls below the contour threshold,

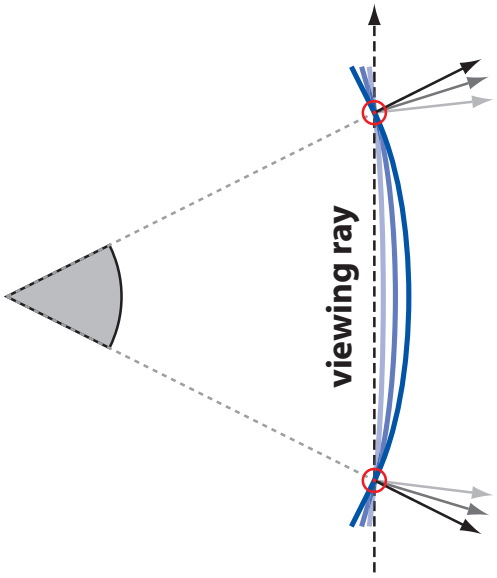


Figure 3.8 – Using the angle between the normals of two subsequent points along a viewing ray as an approximate measure for the curvature along the view direction κ_v.

we simply push the coordinates outwards along the radius of the sphere in the following way: If $r = |n_{x,y}|$, i.e., the length of the eye space normal n projected onto the lit sphere map, we adjust the length of $n_{x,y}$ to $r' = \min(1, \frac{r}{\delta})$ with $\delta = 1 - \min(1, \frac{\sqrt{T\kappa_v(2-T\kappa_v)}-|n\cdot v|}{\sqrt{T\kappa_v(2-T\kappa_v)}})$. This not only allows for varying contour appearance between different styles, but also for a variation based on the normal direction. Contours in a highlight region, for example, may be brighter than in a dark region. Figure 3.7 (b) uses a style with constant contour color while Figure 3.7 (c) employs varying contour colors. Our curvature measure is depicted in Figure 3.7 (d).

3.3.4 Illustrative Transparency

In volume visualization transparency is frequently used in order to depict complex three-dimensional structures. Our approach provides two basic ways to control opacity:

Uniform opacity α_u**.** The opacity value in the transfer function controls the overall opacity of a sample independent of normal direction.

Directional opacity α_d**.** Each entry in a lit sphere map is an (r, g, b, α) tuple. This allows for varying opacity based on the normal direction.

While α_u allows to control opacity independent of style, α_d is a function of the style. For the overall opacity we want to apply the following two

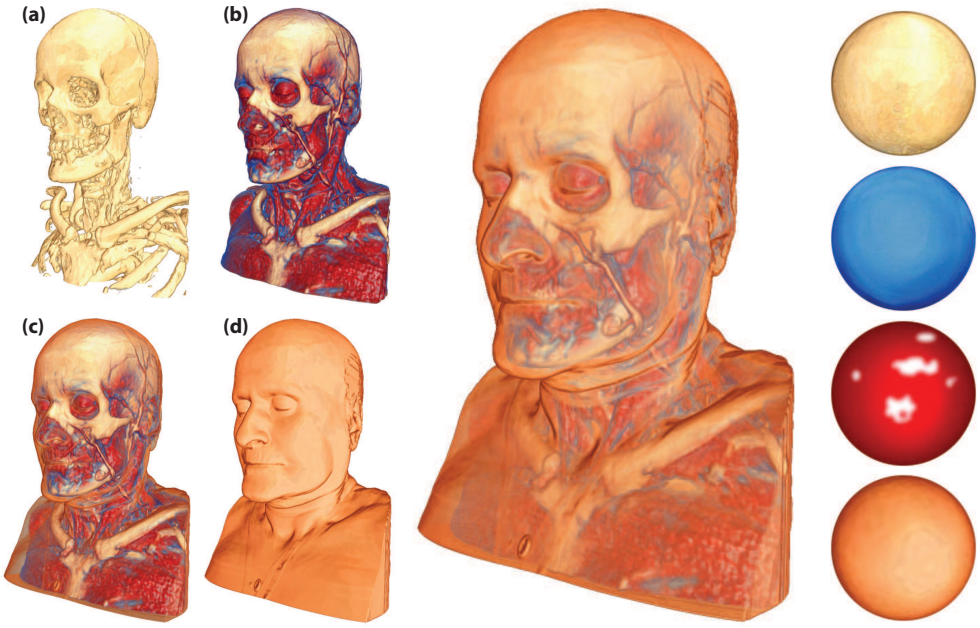


Figure 3.9 – Illustrative volume rendering using a style transfer function. Images (a)-(d) depict different opacity settings.

constraints in order to maintain the semantics of opacity control in the transfer function:

- If the value of α_u is one, the opacity of a sample should be solely determined by α_d.
- If the value of α_u is zero, the sample should be completely transparent.

Transparency in illustrations frequently employs the 100-percent-rule where transparency falls off close to the edges of transparent objects and increases with the distance to edges [25]. Since this technique non-uniformly decreases the opacity of an object, it results in a clearer depiction of transparent structures while still enabling the viewer to see through them. In order to achieve an effect similar to the 100-percent-rule, we employ a modulation of α_u with the curvature measure proposed in the previous section and the gradient magnitude to compute the overall opacity α of a sample:

$$\alpha = \alpha_d \cdot \alpha_u^{0.5+\max(0,|n\cdot v|-\sqrt{T\kappa_v(2-T\kappa_v)})\cdot(1-|g|)} \tag{3.2}$$

If the exponent is lower than one, the opacity of a sample is enhanced, if it is greater than one the opacity is reduced. The term $\max(0, |n \cdot v| - \sqrt{T\kappa_v(2 - T\kappa_v)})$ is zero when the sample point is on the contour, and increases as points are farther away from the contour. The term $1 - |g|$ ranges

from zero to one and is included to prevent enhancement of nearly homogeneous regions, where noise causes the gradient direction to vary rapidly. When decreasing α_u from one to zero, flat and homogeneous regions become more transparent first. As α_u drops further, the remaining contour regions also begin to become more transparent. The constant of 0.5 restricts the maximum opacity enhancement. This value was empirically determined and has proven to be effective for all our test data sets. The overall effect is weighted by α_d. An example for our transparency model is shown in Figure 3.9.

3.3.5 Implementation

In this section we describe our implementation of style transfer functions for a GPU-based ray casting approach. Our renderer makes use of conditional loops and dynamic branching available in Shader Model 3.0 GPUs. It was implemented in C++ and OpenGL/GLSL. The presented techniques can be integrated into an existing renderer using regular transfer functions with little effort.

Style Transfer Function Lookup

A transfer function is usually implemented as a lookup table which corresponds to a 1D texture on the GPU. For conventional transfer functions, this texture stores an (r, g, b, α) tuple for every data value. At each sample point, the interpolated data value is used to perform a texture lookup into this 1D texture to retrieve the color and opacity of the sample. Within the transfer function texture, linear interpolation is performed. A naive implementation of a style transfer function would simply replace the 1D texture by a 3D texture which stores an (r, g, b, α) tuple for every data value and normal direction. This approach requires only one texture lookup and exploits native trilinear interpolation. As discussed in Section 3.3.2 this is not practical due to high storage requirements. Thus, we use an alternative approach which does not suffer from this problem. Our implementation uses three different textures:

Transfer function texture tft. This 1D texture stores the uniform opacities α_u and index values i for each data value. The index values in the transfer function texture range from zero to $N - 1$, where N is the number of styles specified in the style transfer function. For example, an index value of one corresponds to the second style, two corresponds to the third style, etc. Fractional values indicate that an interpolation between two styles has to be performed.

Index function texture ift. As one style might be used multiple times for different value ranges, we define M as the number of distinct styles in the style transfer function. The one-dimensional index function texture maps the index values i (ranging from zero to $N - 1$) retrieved from

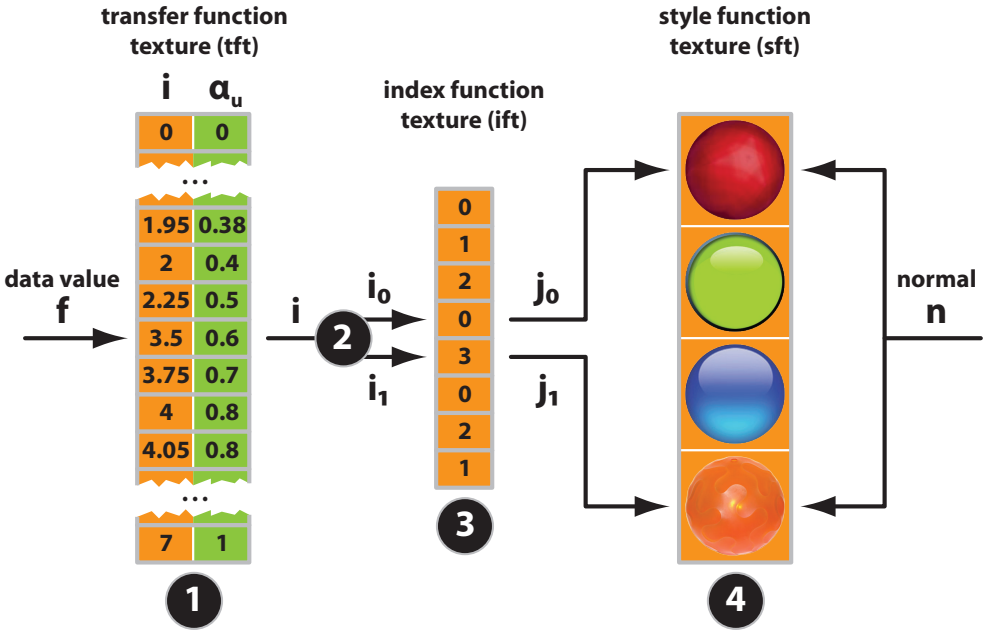


Figure 3.10 – Style transfer function lookup for data value f and normal n.

the transfer function texture to locations j in the style function texture (ranging from zero to $M - 1$). As this mapping is discrete, no interpolation is performed for index function texture lookups. If no style is used multiple times, the index function texture lookup can be skipped.

Style function texture sft**.** This texture contains the discrete set of M distinct styles specified in the current style transfer function. As each style is a two-dimensional image, an intuitive representation for this function would be a 3D texture. Since this can lead to problems with mip-mapping, an alternative way of storage may be more appropriate.

Using these three textures, the complete lookup proceeds as follows (see Figure 3.10):

1. Using the data value f, retrieve the index value i and the uniform opacity α_u from the transfer function texture tft: $(i, \alpha_u) = tft(f)$.

2. Compute the indices to be used in the index function texture lookup $i_0 = \lfloor i \rfloor$, $i_1 = i_0 + 1$ and the interpolation weight $w = i - i_0$.

3. Retrieve the style indices j_0 and j_1 using two lookups into the index function texture ift: $j_0 = ift(i_0)$, $j_1 = ift(i_1)$. If no style occurs multiple times in the style transfer function, these lookups can be skipped.

4. Using the n_x and n_y components of the eye space normal and the style indices j_0 and j_1, perform two lookups into the style function texture sft and linearly interpolate between them: $(r, g, b, \alpha_d) = sft(n_x, n_y, j_0) \cdot (1 - w) + sft(n_x, n_y, j_1) \cdot w$.

Multi-dimensional Style Transfer Functions

So far, we have restricted our discussion to extending one-dimensional transfer functions based on the data value to style transfer functions. However, the presented techniques also apply to multi-dimensional domains. To illustrate the generality of our approach, we briefly describe the changes necessary to employ two-dimensional transfer functions:

The transfer function texture becomes a 2D texture and stores two indices ix and iy instead of i. The first index ix increases along the horizontal axis and the second index iy increases along the vertical axis. The index function texture also becomes two-dimensional. Its width is now the maximum number of horizontal style nodes in the two-dimensional transfer function, its height is the maximum number of vertical nodes. Analogous to the one-dimensional case, the indices for the index function texture lookup are: $ix_0 = \lfloor ix \rfloor$, $iy_0 = \lfloor iy \rfloor$, $ix_1 = ix_0 + 1$, $iy_1 = iy_0 + 1$. The two interpolation weights are also computed accordingly: $wx = ix - ix_0$, $wy = iy - iy_0$. Four lookups into the index function texture ift are performed to retrieve the four style indices: $j_{00} = ift(ix_0, iy_0)$, $j_{10} = ift(ix_1, iy_0)$, $j_{01} = ift(ix_0, iy_1)$, and $j_{11} = ift(ix_1, iy_1)$. Finally, these four style indices and the n_x and n_y components of the eye space normal are used to perform four lookups into the style function texture sft. The final color is computed by bilinear interpolation: $(r, g, b, \alpha_d) = (sft(n_x, n_y, j_{00}) \cdot (1 - wx) + sft(n_x, n_y, j_{10}) \cdot wx) \cdot (1 - wy) + (sft(n_x, n_y, j_{01}) \cdot (1 - wx) + sft(n_x, n_y, j_{11}) \cdot wx) \cdot wy$.

This procedure is independent of the actual quantities mapped to each axis. While data value and gradient magnitude are common choices [58, 59], many other attributes are useful in the context of specific applications. Rendering of segmented data, for example, is frequently realized through a two-dimensional lookup using the data value and an object identifier [44]. Figure 3.1 shows an example of such a multi-dimensional style transfer function. In this way other measures such as distance [117], saliency [56], or importance [108], either predefined or computed on-the-fly, can be mapped to visual styles easily.

Mip-Mapping for Style Transfer Functions

Current graphics hardware uses mip-mapping to avoid aliasing in texture mapping. To take advantage of the GPU's mip-mapping capabilities for style lookups, certain considerations have to be made. First, for 3D textures, each dimension is halved for every subsequent mip-map level. If styles are stored

Figure	Regular TF	Style TF
Figure 3.9 (a)	11.7 fps	11.9 fps
Figure 3.9 (b)	10.5 fps	9.6 fps
Figure 3.9 (c)	10.1 fps	8.1 fps
Figure 3.9 (d)	12.5 fps	12.8 fps

Table 3.1 – Performance comparison of style transfer functions and regular transfer functions measured on a system equipped with an AMD Athlon 64 X2 Dual 4600+ CPU and an NVidia GeForce 7900 GTX GPU. Performance numbers are given in frames per second. Data dimensions: $256 \times 256 \times 230$. Viewport size: 512×512. Object sample distance: 0.5.

as slices of a 3D texture, undesired mixing between styles occurs at higher mip-map levels. Thus, if the style function texture is implemented as a 3D texture, mip-mapping has to be disabled. One solution to this problem is the EXT_texture_array OpenGL extension. A texture array is a collection of two-dimensional images arranged in layers. Mip-mapping is performed separately for each layer. This extension is currently only available on GeForce 8 series graphics hardware. Another possibility is to arrange the styles in a single 2D texture. Although it slightly complicates indexing, this variant is supported on a wider range of hardware. In this case, perform custom mip-map generation has to be performed to avoid mixing between styles at their borders. Conventionally, when performing a texture lookup the appropriate mip-map level is determined by the hardware using the screen-space derivatives of the texture coordinates. These derivatives are undefined when the texture fetch takes place within conditionals or loops. Thus, as our algorithm uses raycasting, we cannot exploit the standard mechanism. We therefore manually compute the size of a projected voxel at each sample point to determine the mip-map level. In areas of high curvature, the gradient direction varies more quickly, which can lead to artifacts. Additionally, when its magnitude approaches zero, the gradient vector becomes a less reliable predictor for the normal direction. We therefore also bias the determined mip-map level using a function of curvature and gradient magnitude.

3.3.6 Results

In comparison to a regular one-dimensional transfer function, a style transfer function lookup requires a maximum of four additional texture fetches (two for the index function texture and two for the style function texture). The index function texture does not require filtering and is rather small. It therefore heavily benefits from texture caching. On GeForce 8 series hardware it could also be implemented as a buffer texture using the EXT_texture_buffer_object OpenGL extension for additional performance gains. Although the additional

texture fetches incur an overhead, the cost for evaluating the illumination model is saved when using style transfer functions.

To evaluate the performance of our approach, we compared the use of a style transfer function for classification and shading to a regular one-dimensional transfer function with simple Phong shading. The same opacities were used for both transfer functions. Both implementations use empty-space skipping and early-ray termination. The viewport size was 512×512 and the object sample distance was set to 0.5. We used the data set depicted in Figure 3.9 (dimensions: $256 \times 256 \times 230$). Our test system was equipped with an AMD Athlon 64 X2 Dual 4600+ CPU and an NVidia GeForce 7900 GTX GPU. The results of this comparison are shown in Table 3.1. If only one style is visible, the performance is approximately equal (style transfer functions are even slightly faster) as all lighting computations are replaced by texture fetches which benefit from coherent access. If multiple styles are visible, the style transfer function performs slightly worse due to texture caching effects. In total, however, the overhead of employing a style transfer function is only minor but greatly increases the flexibility.

3.3.7 Discussion

In our experiments, style transfer functions have proven to be a simple method for generating images and animations in a wide variety of different appearances. Lit sphere maps are particularly effective in representing the styles typically used in medical illustrations. Our approach is well-suited for this application, as illustrations frequently rely on certain shading conventions. This means that a database of styles can potentially be reused for a large number of data sets. Figures 3.1 and 3.9, for example, use styles obtained from medical illustrations. Another advantage is that the theme of an image can be changed quickly by simply replacing one set of styles with another one. This is illustrated in Figure 3.11, where two very different results are achieved by a simple exchange of styles.

While the representation of styles as lit sphere maps has proven to be effective and efficient, it has drawbacks. One problem already discussed by Sloan et al. [97] occurs when the sphere contains prominent texture features. When the camera is rotated, they will appear to follow the eye leading to an undesired metallic impression. To solve this problem, texture and lighting information have to be separated. The texture information could then be aligned to the object, for example based on curvature directions. This might be an interesting direction for future research.

3.4 Volumetric Halos

Resolving the spatial arrangement of complex three-dimensional structures in an image can be a difficult task. Particularly renderings of volume data

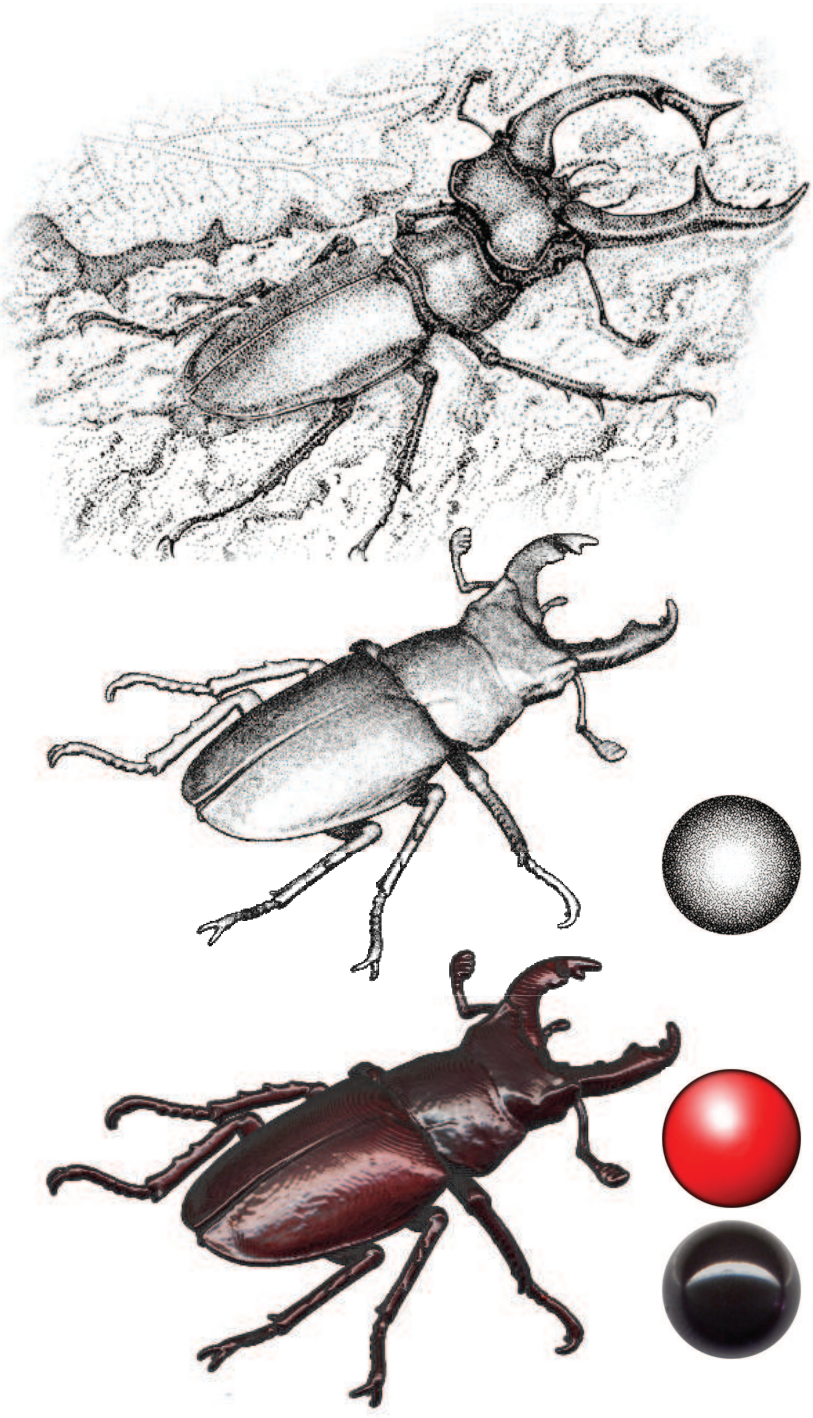


Figure 3.11 – Changing the theme of an image by replacing styles. Top: Drawing of a staghorn beetle by A. E. Brinev. Middle: Volume rendering of a staghorn beetle using a similar style. Bottom: Staghorn beetle rendered using a more realistic style.

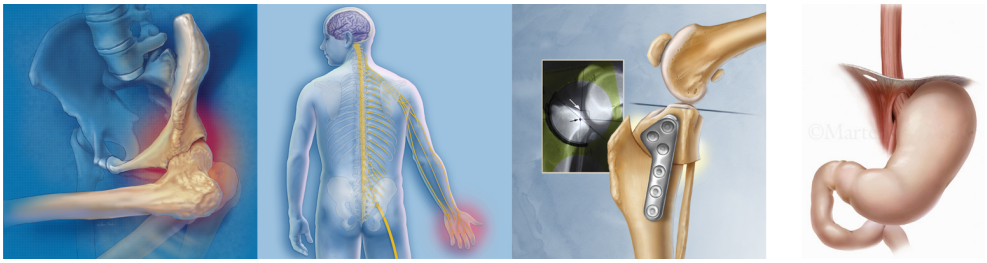


Figure 3.12 – Examples of the different uses of halos in medical illustration for emphasis and accentuation from the *Medical Illustration Source Book*[1].

acquired by imaging modalities such as CT or MRI often suffer from this problem. As such data frequently contain many fine, overlapping structures, the resulting images often look confusing and are difficult to interpret without additional cues. For this reason artists and illustrators have long exploited the fact that the human visual system is especially sensitive to local variations in contrast by drawing halos around objects. Near the boundary of objects that are located in front of other structures the tone is locally altered: the background or partly occluded objects are slightly darkened to create the impression of depth. Similarly, bright halos are frequently placed around objects to visually detach them from the background. In the simplest case, a halo is just a gap around the edges of an object. In photography and film halos, achieved by careful placement of lights and camera, are commonly used to accentuate objects. While this technique is motivated by natural lighting phenomena such as shadows and atmospheric effects, halos in illustrations are typically overemphasized and localized to enhance occlusion cues. Halos are used in a wide variety of different styles as illustrated in Figure 3.12. Manifestations of the effect range from thick opaque outlines to soft darkening of the background almost undistinguishable from realistic shadows under diffuse illumination. Frequently, halos are used as a subtle way to put emphasis on certain objects. Our goal is to enable the same kind of flexibility for computer-generated halos in direct volume rendering. As it is often dependent on the context of the visualization which kind of halo provides the most effective cues, we present an approach which allows interactive adjustment of their appearance.

3.4.1 Halo Pipeline

Previous halo generation approaches for volume rendering have frequently relied on a pre-processing step which generates a volume of halo contributions [53, 86]. This halo volume is then used during the rendering process to identify halo regions. The problem of this approach is that it does not

[1] `http://www.medillsb.com`

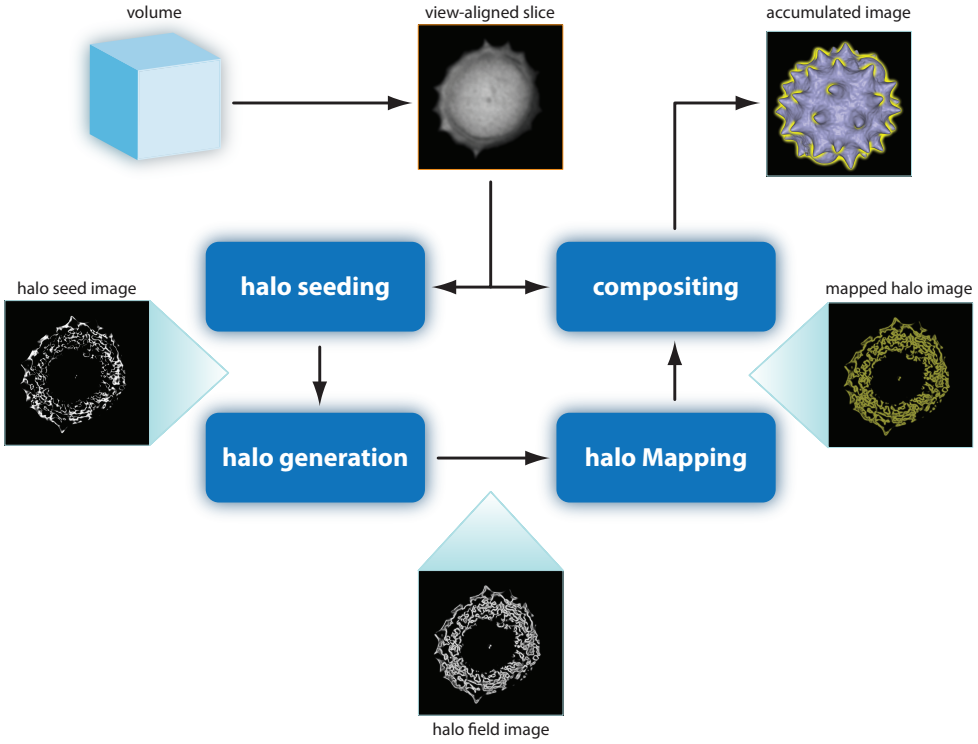


Figure 3.13 – Overview of the halo pipeline. The volume is processed in view-aligned slices. Halo seeding classifies halo-emitting structures, halo generation distributes the seed intensities, and halo mapping assigns colors and opacities to these distributed seed intensities. Compositing combines the mapped halo intensities with the actual volume rendering.

allow for easy modifications of many parameters. In order to remedy this, our approach determines halo contributions during volume rendering. The algorithm operates on view-aligned slices through the volume in front-to-back order. In addition to regular sampling, classification, shading, and compositing, a halo generation pipeline is executed for every slice to process its halo contributions. The pipeline consists of three basic stages and an additional compositing step for blending the halo with the regular volume rendering. Figure 3.13 illustrates this process. First, regions to emit a halo are identified. We will refer to this step as halo seeding (see Section 3.4.2). Next, a field of halo intensity values is generated from the seeds by applying a filtering process (see Section 3.4.3). Finally, the halo intensities are mapped to the actual color and opacity contributions of the halo and combined with the regular volume rendering (see Section 3.4.4). For simplicity, the following description is only concerned with one halo. Our approach allows multiple halos to be defined, each with its own set of parameters.

3.4.2 Halo Seeding

For generating volumetric halos we need to classify which structures should emit halos – we call this process halo seeding. During halo seeding, a seed intensity value is generated for all samples on a view-aligned slice through our volumetric function f. Every point with nonzero halo seed intensity is a seed point. These seed intensity values are used in the subsequent step to derive the halo intensity values for other locations.

As halos are only drawn around the contours of objects, we need to limit our seeds to these regions. In volume rendering, contours can be characterized by the angle between the view vector v and the normal n. If these vectors are nearly orthogonal, the sample point is on a contour. Furthermore, the magnitude of the gradient vector $|g|$ can be used for preventing noise in nearly homogeneous regions to produce erroneous halo seeds. Using these two attributes, we can generate effective halo seeds for a given volumetric data set [86].

However, since we also want to generate localized halos which are only emitted by certain structures, we introduce a halo transfer function $h(p)$ to specify the halo contributions at the sample position p. The halo transfer function consists of several separable scalar-valued functions in the range $[0..1]$. Our approach currently supports three different components, but this could be easily extended to include, for instance, segmentation information, if available:

Value influence function $h_v(p)$. This function is based on the data value at the sample point. It is useful, for example, for generating localized halos by limiting their influence to a certain value range.

Directional influence function $h_d(p)$. This function is based on the direction of the eye space normal, i.e., the angle between the projected gradient vector and the positive vertical axis of the image plane. It allows for directionally varying halos.

Positional influence function $h_p(p)$. This function is based on the distance of the sample point to a user-defined focus point to allow easy generation of localized halos for regions which cannot be identified solely using the data value.

The halo transfer function is then simply defined as the product of these components [59]:

$$h(p) = h_v(p)\ h_d(p)\ h_p(p) \tag{3.3}$$

The halo transfer function defines a basic seed intensity at a sample position p. This value is then combined with the gradient magnitude $|g|$ and the

dot product between view vector v and the normal n to form the final seed intensity $s(p)$:

$$s(p) = h(p)\,|g|^{\alpha}\,(1 - n \cdot v)^{\beta} \tag{3.4}$$

where α and β are used to control the influence of the gradient magnitude and the dot product, respectively. For halos these values are usually fixed and do not require adjustment. We use values of $\alpha = 32$ and $\beta = 0.125$ for all result images. The values of s are clamped to the range $[0..1]$. The result of applying s to all pixels of the view-aligned slice is the halo seed image S.

This definition can lead to unevenly distributed halo seeds as contours identified by the dot product between view vector and normal can vary in thickness. To avoid this, we could additionally employ a modulation based on the normal curvature along the viewing direction as proposed by Kindlmann et al. [57] analogously to the method described in Section 3.3.3. However, as our halo generation approach (see Section 3.4.3) takes special care to equalize halo contributions, we found that this is generally not necessary.

3.4.3 Halo Generation

Per definition halos are located outside of objects, while the halo seeds lie within other structures. Therefore, during halo generation the seed intensities are spread out to form a halo field image H. Each point within the halo field is defined by having nonzero halo intensity.

One important aspect of halo generation is the fact that halo contributions from smaller structures should not be lost during the spreading process. A naive approach would be a simple convolution of the halo seeds with a low-pass filter. This method, however, results in a reduction of halo contributions from smaller regions. The halo seeds of small structures are essentially blurred out of existence if the filter kernel is too large, although exactly those features could particularly benefit from the emphasis provided by a halo. If the kernel size is too small, on the other hand, the seed intensities are not distributed enough to generate a visible halo. This effect is illustrated in Figure 3.14. In Figure 3.14 (a) an artificial halo seed image featuring regions of multiple scale is shown. When a low-pass filter is applied to it, as shown in Figure 3.14 (b), the seed values are spread out, but contributions from smaller areas are lost. Other approaches such as the unsharp masking technique used by Luft et al. [73] also suffer from this problem. Thus, while acceptable from an aesthetic point of view, these methods are not suitable for our purpose.

A different approach to generating the halo field would be to perform a distance transform on the seed image. As this is computationally expensive and not well-defined on non-binary images, we employ a spreading approach which preserves the halos of small features while still generating a smooth halo field. The algorithm executes in N passes. During each pass $i \in [1..N]$ two input images are used: H_0, the initial image, and H_{i-1}, the result of the

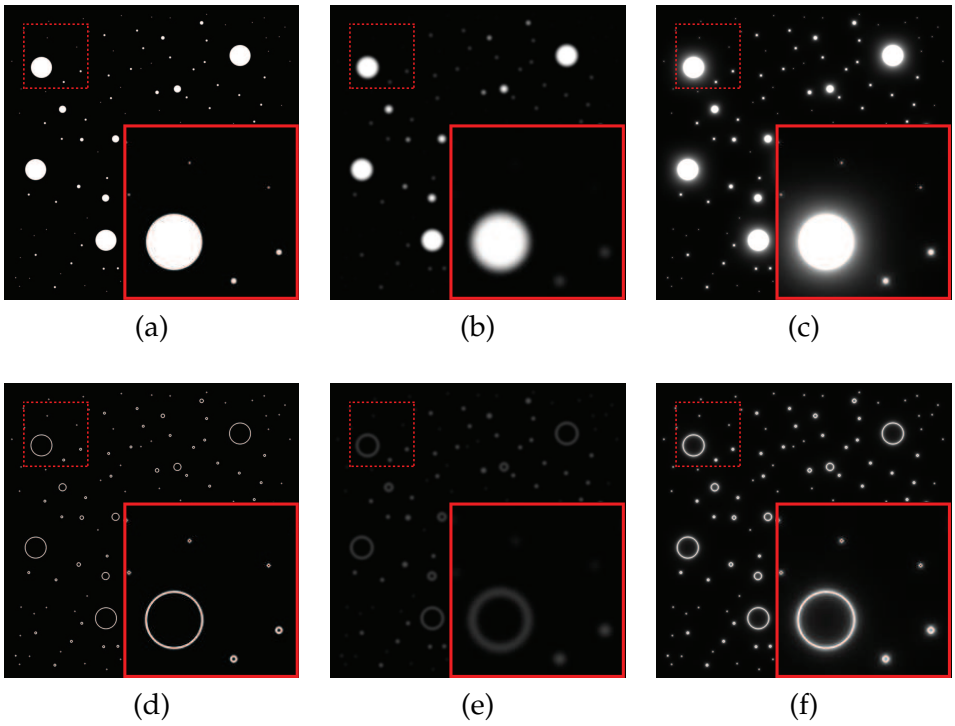


Figure 3.14 – Comparison of halo generation strategies on a test image. (a) Original halo seed image. (b) Low-pass filtered halo seed image. (c) Spreading process applied to halo seed image. (d) Gradient of halo seed image. (e) Low-pass filtered gradient image. (f) Spreading process applied to gradient image.

previous pass. For the first pass, the two input images are identical. For each output pixel (x, y), the algorithm first performs a convolution with a low-pass filter over the pixels of H_{i-1}. Then it combines the result of this operation with the corresponding pixel from H_0:

$$H_i(x, y) = \delta F_i(x, y) \; + (1 - \delta F_i(x, y)) \, H_0(x, y) \tag{3.5}$$

with

$$F_i(x, y) = \sum_{u,v} w(u, v) H_{i-1}(x + ku, y + kv) \tag{3.6}$$

where w is the weight function of the filter, $u, v \in \{-1, 0, 1\}$, $k = 2^{N-i}$, and δ is a user-specified parameter in the range $[0..1]$ which controls the amount of spreading. If δ is zero the unfiltered seed image will be passed through. Increasing δ causes an increased contribution from previous passes and therefore results in a smoother, more spread-out halo while still preserving higher frequency components. Small features are preserved due to the fact that spread out values are filled up in every pass.

This algorithm is effective in distributing the halo seed values to neighboring pixels without removing high frequencies. This is visible by comparing Figure 3.14 (b), which uses normal low-pass filtering, to Figure 3.14 (c), which depicts the result obtained with our approach.

However, there is still a problem as larger regions now generate a significantly larger halo. In order to remedy this, we apply the spreading algorithm to the gradient of the halo seed image instead of the original which equalizes the contributions, as illustrated in the bottom row of Figure 3.14. Figure 3.14 (d) depicts the gradient of the seed image. Figure 3.14 (e) shows that only applying a low-pass filter to the gradient image is not effective. The presented process applied to the gradient of the halo seed image, however, results in a smooth halo field with equalized contributions from structures of multiple scale, as depicted in Figure 3.14 (f).

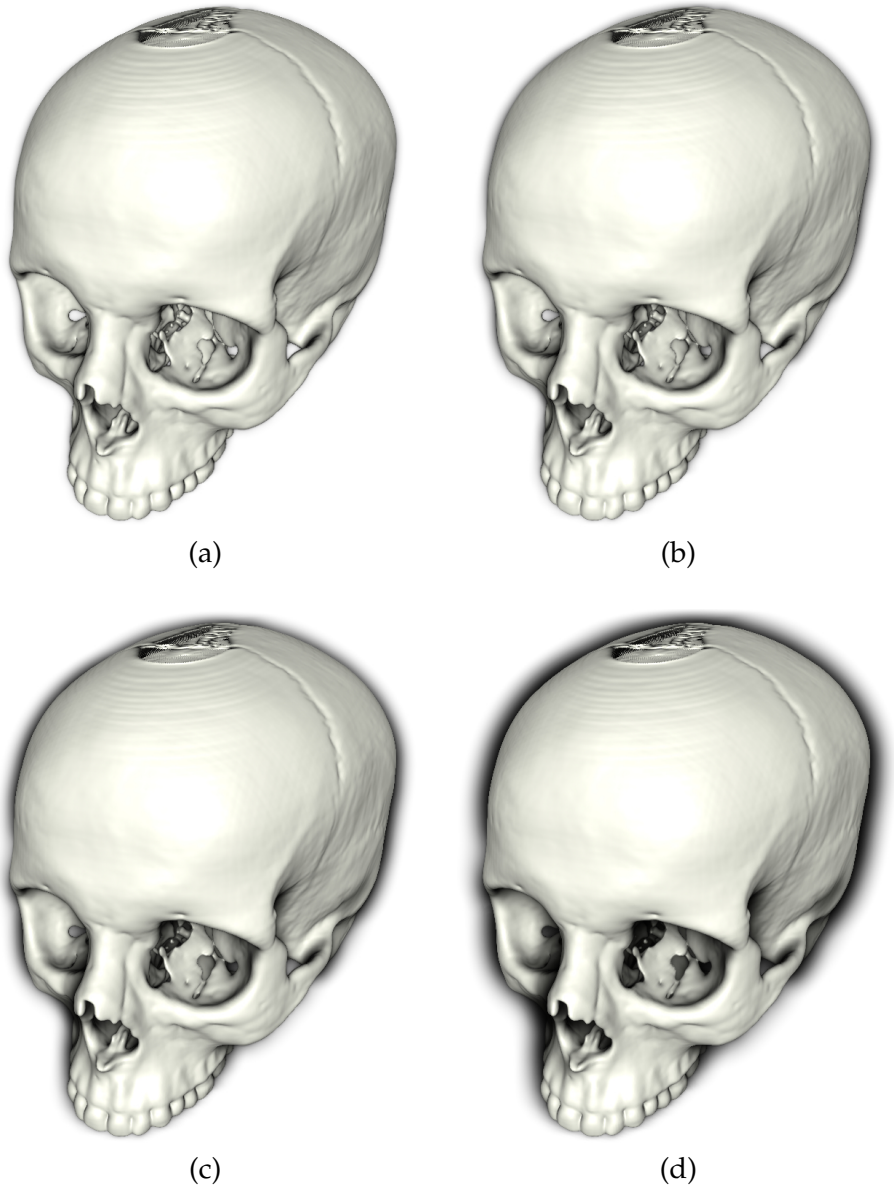
If some reduction of high frequencies is desired, a median filtering could be applied to the seed image before the spreading process. However, in our experiments we found that this is not necessary in general. We use a filter kernel size of 3×3 with Gaussian weights. The number of iterations N determines the maximum amount of spreading that can occur – for all our purposes a value of $N = 4$ has shown to be sufficient. This algorithm is conceptually similar to the jump flooding paradigm for parallel computing [88]. Figure 3.15 shows results of the spreading processes for different values of δ.

3.4.4 Halo Mapping and Compositing

After generating the halo field image H it has to be mapped to visual contributions in the image. For this purpose we employ a halo profile function: this function maps all nonzero halo intensity values to colors and opacities. Halo intensities of zero are always transparent. While the spreading parameter δ only controls the distribution of intensities in the halo field, the profile function allows further adjustment of the halo appearance.

In the simplest case, the halo profile function just maps halo intensities directly to opacities using a constant color. Other possibilities include, for instance, a halo profile with constant opacity which results in a sharp border. Figure 3.16 shows a few examples. In the last row of this figure, the use of directionally varying halos is also demonstrated. Finally, the mapped halo has to be combined with the volume's contribution. Based on how this combination is performed, we can distinguish between two different kinds of halos:

Emissive halos. Similar to scattering of light by small particles such as fog, this type of halo causes a visible contribution by itself. From the point of view of compositing, the halo behaves as if it were part of the volume. Thus, for emissive halos the halo intensity value is first mapped using the halo profile function and then blended after the actual volume con-



(a) (b)

(c) (d)

Figure 3.15 – Results of the halo spreading algorithm using $N = 4$ iterations with (a) $\delta = 0.70$, (b) $\delta = 0.80$, (c) $\delta = 0.90$, (d) $\delta = 0.95$.

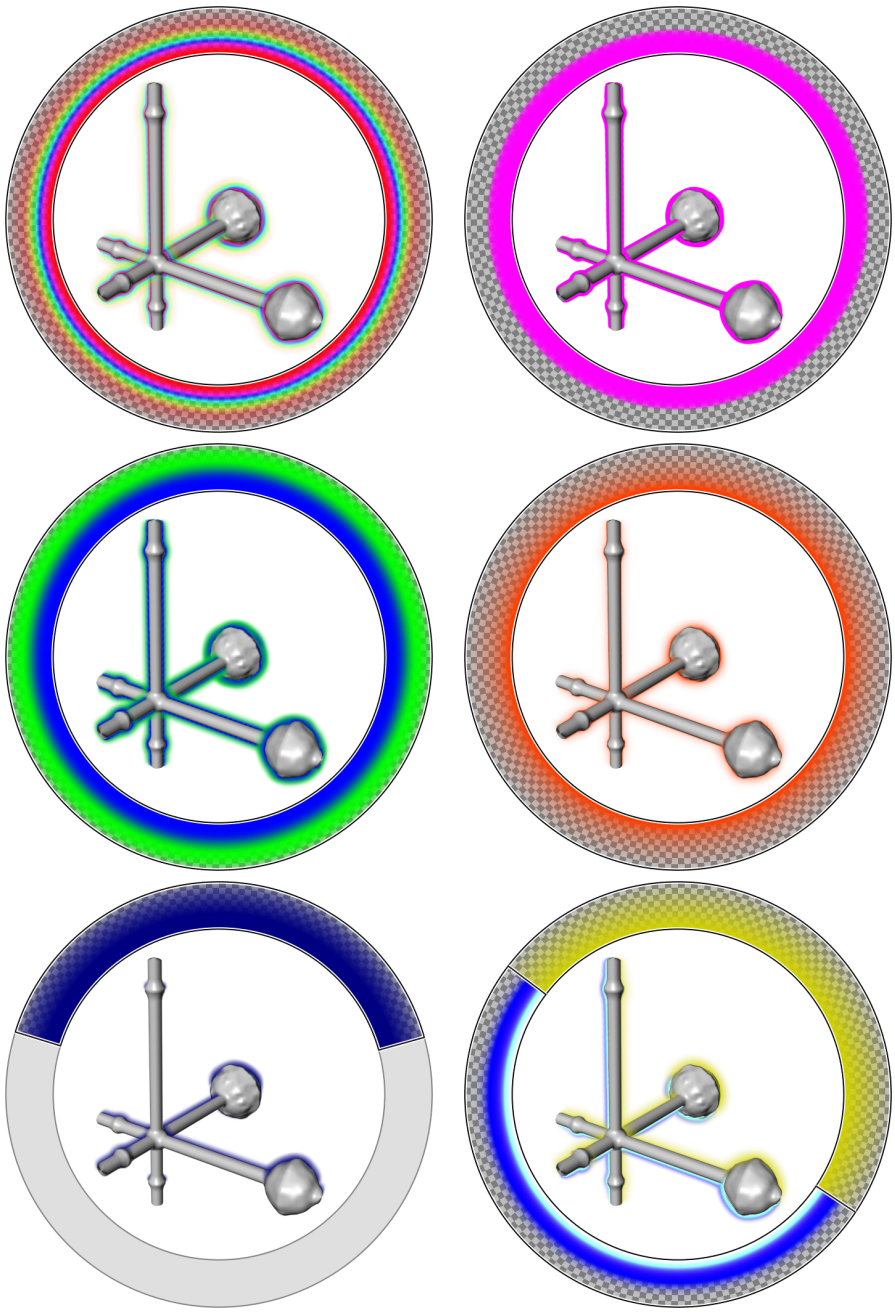


Figure 3.16 – Different halo profile functions applied to a simple data set. The corresponding halo profile function is shown for each image.

tributions using the front-to-back formulation of the over-operator. The halo therefore (partially) occludes everything located behind it including the background.

Occlusive halos. In addition to emissive halos, illustrators sometimes employ another kind of halo: the halo only contributes to the image if it occludes other structures – the halo by itself has no emissive contribution. Although similar to a shadow, this type of halo is usually drawn in the same style irrespective of the distance between the two objects and not necessarily consistent with the overall lighting conditions to strengthen the occlusion cues. This type of halo can be useful as it might be less intrusive and only highlights occlusions. For generating occlusive halos, contributions of the halo field image H need to be accumulated to be able to influence samples located behind them – this is similar to a shadow buffer [60]. For this purpose, we introduce an additional halo occlusion image O. The current halo field image H is combined with O in every pass using maximum blending. Halo mapping is then performed based on the halo occlusion image. The resulting mapped halo color is mixed with the volume sample color using the mapped halo contribution's opacity as an interpolation weight. The opacity of the volume sample remains unchanged. Thus, if no sample is occluded by the halo, it has no contribution to the image.

Both halo types are useful for different purposes. While emissive halos can be used to emphasize particular features by giving them an outline or a glow, occlusive halos provide a means for accentuating occlusions by enhancing the contrast in areas where one object crosses another. For instance, occlusive halos are frequently used in depictions of vascular structures. Figure 3.17 shows a comparison of these two styles. Figure 3.17 (a) depicts a volume rendering without halos. In Figure 3.17 (b) an emissive halo is used for bones and vessels while Figure 3.17 (c) employs an occlusive halo. The emissive halo also generates a dark border around objects which do not occlude other structures. Figure 3.17 (c) and (d) also demonstrate the use of the directional component of the halo transfer function. In Figure 3.17 (c) a one-sided halo was generated using the directional component of the halo transfer function. Figure 3.17 (d) shows an omnidirectional halo for comparison. The advantage of directional halos approach over realistic shadows lies in the fact that they can be easily adjusted without influencing the global appearance while providing emphasis in the targeted areas. Illustrators generally do not draw physically correct shadows, but they exploit the fact that the human visual system interprets this cue as an indicator for occlusion. Emissive halos are also useful for putting emphasis on particular features by generating a glowing effect, as shown in Figure 3.18.

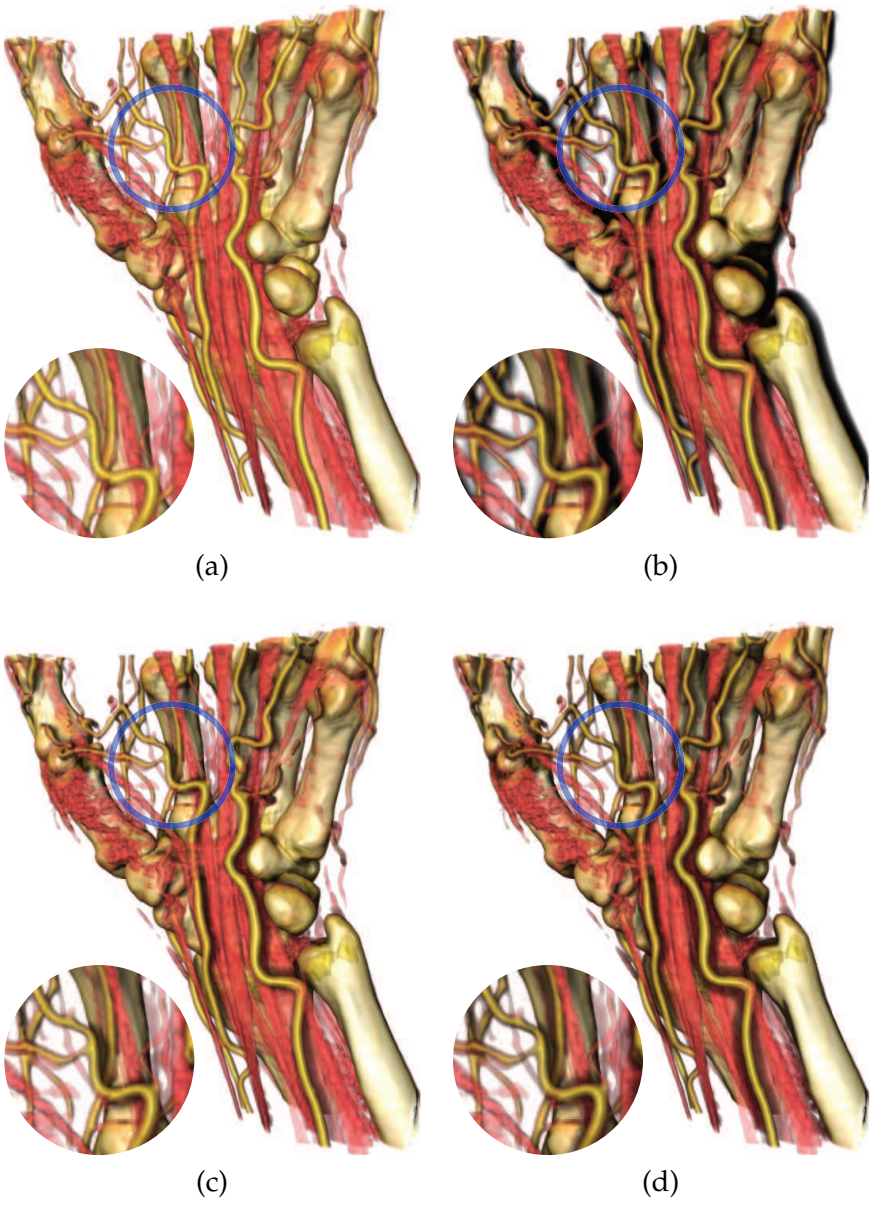
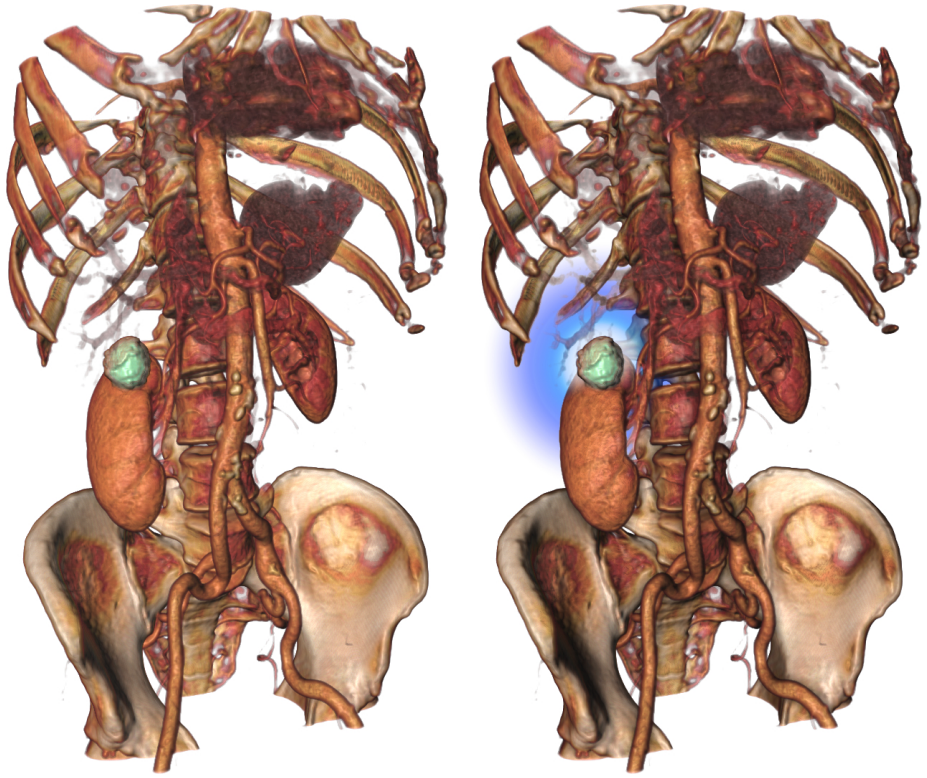


Figure 3.17 – Emissive and occlusive halos. (a) Volume rendering without halos. (b) A directional emissive halo is emitted by bones and vessels. (c) A directional occlusive halo is emitted by bones and vessels – it is only visible where it occludes other structures. (d) The occlusive halo emitted by bones and vessels is omnidirectional.



(a) (b)

Figure 3.18 – Using emissive halos to highlight features. (a) Volume rendering without halos. (b) A tumor is emphasized using an emissive halo and a soft occlusive halo is used to increase contrast.

3.4.5 Implementation

Our GPU-based volume renderer with halo support was implemented in C++ using OpenGL/GLSL. As already outlined in Section 3.4.1 our approach for integrating halos with direct volume rendering is based on an interleaving of the halo generation pipeline with conventional view-aligned slicing of the volume. In the following, we discuss further details of our implementation.

Initially, six off-screen buffers are generated: I_0 and I_1 are used for compositing in a ping-pong approach. In our description, we use I_p to denote the accumulated image from the previous iteration, and I_c for the image written in the current iteration – they are swapped after each iteration. The buffer S stores the halo seed image, and H_0, H_1, and H_2 are used for halo generation. We use H to denote the buffer which contains the current halo field image, which can be either H_1 or H_2. For occlusive halos, an additional halo occlusion image O is required. We use the OpenGL ARB_framebuffer_object extension

to render to and read from these buffers. The renderer slices the volume in viewing direction and has two basic phases:

Rendering. Although described separately in Section 3.4.1, as they are conceptually different stages, it is advantageous to combine halo seeding and halo mapping with regular sampling, classification, shading, and compositing in a single rendering pass. Since all these steps require the data value and gradient at the same sample location, this avoids redundant texture reads and computations and eliminates the need for an explicit mapped halo image. We take advantage of OpenGL's capabilities to write to multiple render targets in one rendering pass. The two images written to are I_c, the current accumulated image, and S, the halo seed image. First, conventional sampling, classification and shading is performed. Next, halo mapping is performed on the halo intensities read from the current halo field image H for emissive halos. For occlusive halos, the halo occlusion image O is used instead. For emissive halos, the shaded color of the volume sample is first composited with the previously accumulated color read from I_p and then written into I_c. Then the mapped halo contribution is composited. In the case of an occlusive halo the shaded sample color is mixed with the mapped halo contribution and then composited with the previously accumulated color. Finally, halo seeding is performed and the result is written to S.

Generation. This phase performs halo generation as described in Section 3.4.3. First, the gradient magnitude of the halo seed image is computed and written into H_0. Next, several iterations of the spreading algorithm are executed on H_0 by ping-ponging between buffers H_1 and H_2. The final halo field image is then located in buffer H_1 or H_2, depending on the number of iterations. The current halo field image H is set to this buffer. If occlusive halos are used, this image is additionally blended with the halo occlusion image O. Halo generation is the most expensive part of the rendering procedure. However, it is not necessary to perform this step for every iteration. We can use OpenGL's blending functionality to accumulate the halo seeds of several slices and then perform halo generation on this image only every M-th slice. As the seed contributions are accumulated, no features will be missed. Only if M is chosen too high, some depth accuracy is sacrificed. In practice, a value of $M = 4$ has proven to result in no visible artifacts – all result images were generated using this setting.

As the GPU operates on vectors rather than scalars we can support four distinct halos, each with its own set of parameters, without additional costs. Each halo is assigned a color channel and all operations, including halo generation, are simply performed on four halo channels instead of one.

M	frames/second	% of reference
1	4.65	15.85
2	7.72	26.31
4	10.26	34.97
8	12.59	42.91

Table 3.2 – Performance of halo rendering measured on an AMD Athlon 64 X2 Dual 4600+ CPU equipped with an NVidia GeForce 8800 GTX GPU. The volume size was $256 \times 256 \times 256$ with a viewport size of 512×512 and an object sample distance of 0.5. The reference renderer without halos achieved 29.34 frames/second in this benchmark.

3.4.6 Results

To evaluate the performance of our implementation we compared a standard volume renderer to our algorithm. We used the UNC head test dataset (dimensions: $256 \times 256 \times 256$) and an object sample distance of 0.5. The viewport size was 512×512. No high-level optimizations such as empty-space skipping were used. Our test system was an AMD Athlon 64 X2 Dual 4600+ CPU equipped with an NVidia GeForce 8800 GTX GPU. We performed a 360 degree rotation along each axis and averaged the frame rates. The reference renderer without halos achieved 29.34 frames/second in this benchmark. The frame rates of our halo renderer for different values of M are shown in Table 3.2. Compared to the reference renderer, our approach achieves approximately one third of the frame rate for the typical setting of $M = 4$. Halo seeding and mapping alone, due to the increased number of texture fetches and operations per sample, result in a slowdown of about 50 percent. Although halo processing incurs an overhead, interactive frame rates are still achieved. Moreover, as our implementation is not heavily optimized, there is potential room for improvement.

Halos provide a simple additional option for the generation of volumetric illustrations. They do not require any pre-processing and can be easily integrated into existing volume visualization tools. Similar to layer effects in image editing software such as Adobe Photoshop, they can be applied to enhance or highlight specific regions with great stylistic flexibility ranging from opaque contour-like lines to smooth object shadows. These techniques are ubiquitous in traditional illustrations and so it makes sense that volume visualization can benefit from them. In our experiments, we found that volumetric halos can be an effective and versatile tool for enhancing the visualization of volume data with little additional effort. A wide variety of data sets can benefit from halos. We present a small selection of visualization results and compare them with unenhanced depictions.

Halos are effective for the visualization of transparent objects. Illustrators frequently employ this technique in line drawings: The background object

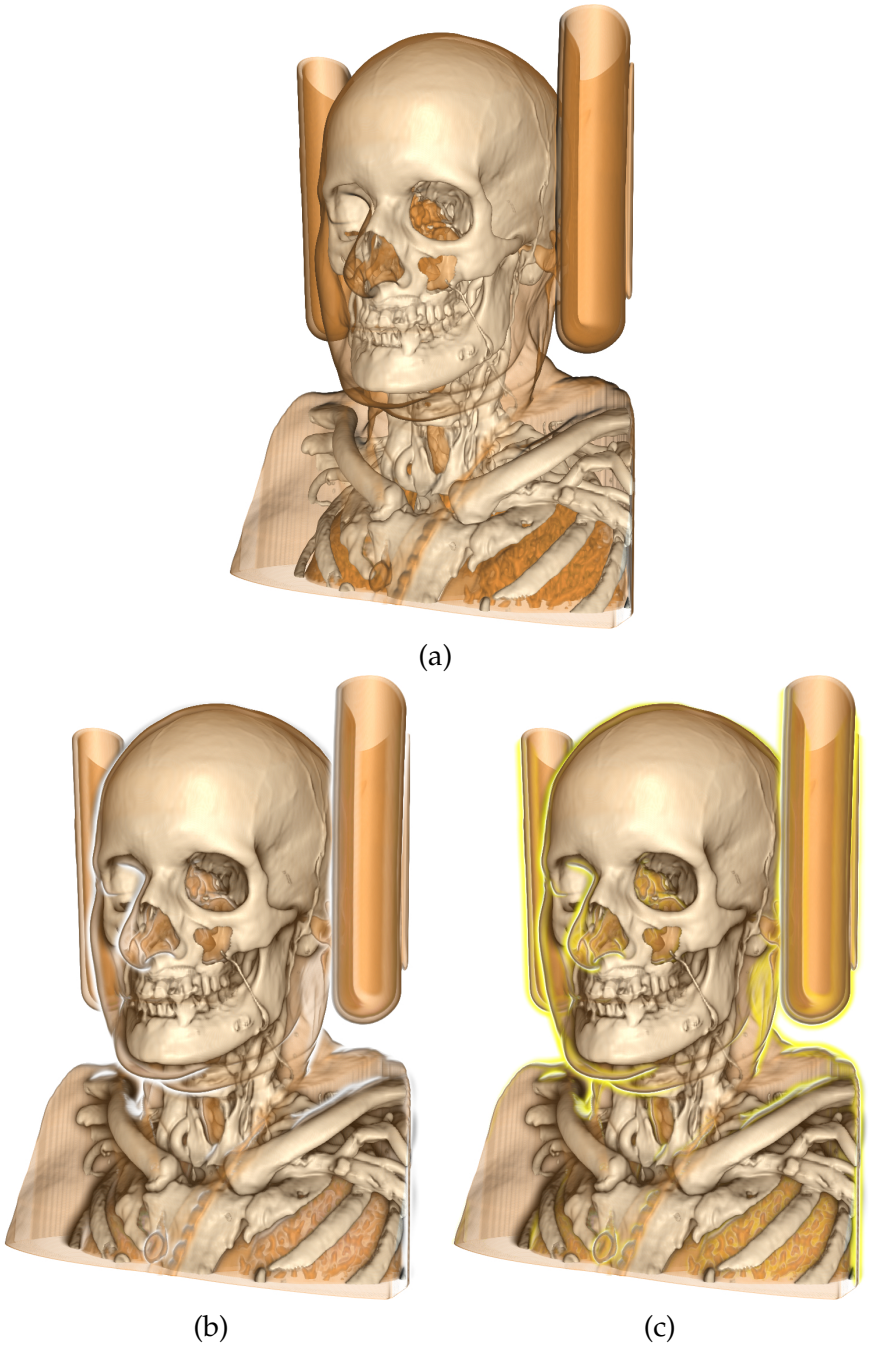


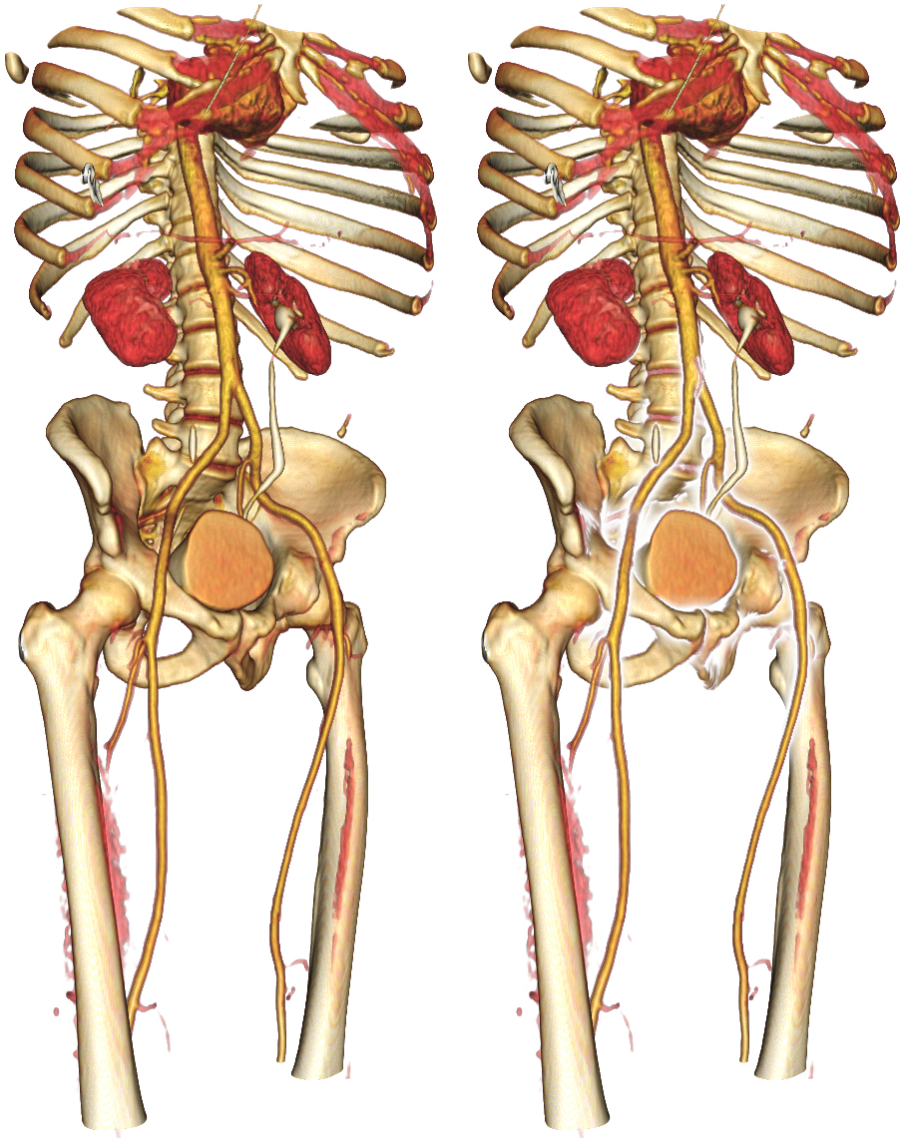
Figure 3.19 – Improving visualization of transparent structures using halos. (a) Volume rendering without halos. (b) Depth is added to the bone by using a dark smooth halo, contours of the the transparent skin are enhanced using a white opaque halo. (c) A different halo profile function is used for the skin (fade from white to yellow).

is rendered in full detail and disappears as it approaches the boundaries of the overlapping translucent structure [51]. Figure 3.19 demonstrates this approach: an opaque halo is used to enhance the contours of the transparent skin. Additionally, a smooth dark halo is assigned to the bone to add depth to the image. Figure 3.20 demonstrates the use of the positional component of the halo transfer function. A focus point is placed in the pelvic region and a bright halo is used to indicate occlusions. Through these simple means the viewer's attention is directed to the aorta. Figure 3.21 shows the combination of different halos for the same data value range. A smooth dark halo and a thin bright halo are assigned to the outer part of the engine block. The darkening of the white contour halo helps in indicating occlusion relationships. The inner structures additionally have a slight cyan glow. Halos may also serve as a complete replacement for normal gradient-based illumination similar to the approach described by Desgranges et al. [24]. Depending on the appearance of the halo, this results in an additional degree of stylization which is useful to depict contextual objects. In Figure 3.22 we demonstrate this effect. Figure 3.22 (b) and Figure 3.22 (c) depict different combinations of halos using shading, while Figure 3.22 (d) uses no additional shading. The stylized cartoon-like effect in Figure 3.22 (d) is achieved through a smooth background-colored halo in combination with a black contour halo.

3.4.7 Discussion

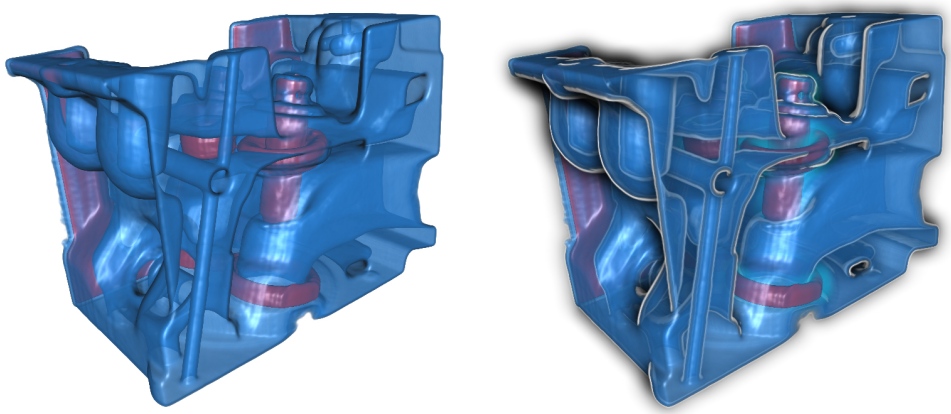
As demonstrated, halos can be used to generate easily controllable inconsistent lighting, which is not physically plausible but aesthetically pleasing. Localized shadow-like halos help to emphasize spatial relationships in an non-intrusive way as they do not change the global lighting situation. As shown by Ostrovsky et al. [80], humans are largely insensitive to such inconsistencies. Artists and illustrators therefore use inconsistent lighting to guide the viewer's attention and to enhance comprehensibility. Cavanagh [13] has suggested that our brain perceives the shape-from-shading cues only locally. This might be the reason why local cues which are inconsistent with global lighting are so effective. With volumetric halos we can generate this kind of illumination in an interactive result-oriented manner. In general, halos perform best for volumetric data which contain clearly defined features such as tomographic scans. For rather amorphous data, their benefit is limited due to the absence of distinct boundaries.

Currently, we provide a simple interactive user-interface for changing all halo parameters. The value influence function of the halo transfer function is specified using a conventional transfer function widget. The user has the option to link this function with the normal transfer function, i.e., halos can be connected to the corresponding visible structures. For the directional influence function, we provide a simple angular brushing widget which allows the specification of a range of directions on a radial layout. The positional



(a) (b)

Figure 3.20 – Using halos to add occlusion cues. (a) Volume rendering without halos. (b) A white opaque halo is specified in the pelvic region using a focus point in order to accentuate the aorta.



(a) (b)

Figure 3.21 – Combining multiple halo effects. (a) Volume rendering without halos. (b) The semi-transparent part of the engine block uses a dark smooth halo and a white opaque halo. The inner structures have an additional cyan glow.

influence function is defined by clicking on a point in the image. A viewing ray is cast from this position and records the first intersection with visible structures. The focus point is set to this position. By dragging the mouse, the inner radius and transition of the spherical focus region can be modified. The halo profile function is defined using a gradient widget. Other parameters, such as the spreading parameter δ and an additional opacity scale of the profile function can also be edited using a click-and-drag interface. While these tools are very basic, they have proven to be surprisingly effective in quickly improving the appearance of volume renderings. However, there are some enhancements that could be easily accomplished. Extending the focus point to a user-drawn polyline, for example, would be a useful extension. The resulting geometry could then be sliced in parallel to the volume and serve as an input to halo seeding. This approach would allow for even better localization of halos without the need for explicit segmentation. Other such sketch-based interaction metaphors open an interesting direction for further research.

While our interactive approach allows for quick adjustment of halo parameters, an unsolved question is which settings perform best for which types of data. A study of perceptual performance could lead to valuable insights and the resulting data could be used to derive halo templates for different scenarios.

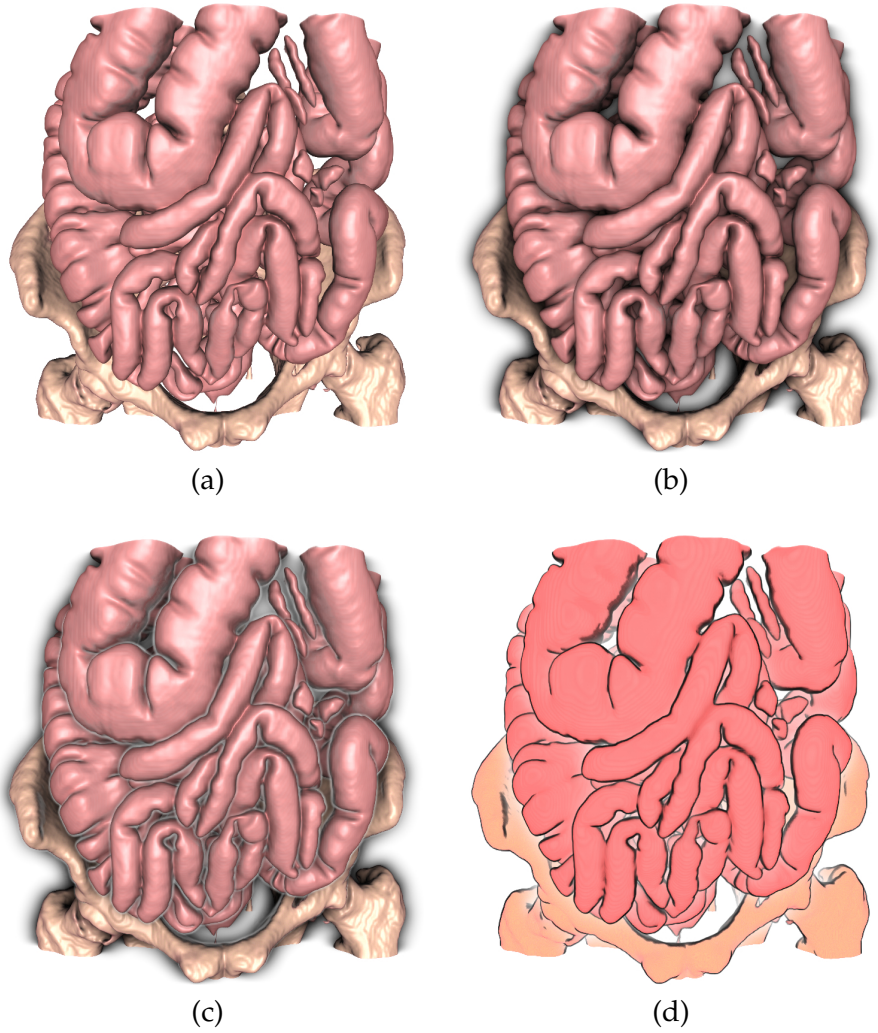


Figure 3.22 – Combining global and regional halos. (a) Volume rendering without halos. (b) A smooth shadow-like halo is used to improve depth perception. (c) An additional bright semi-transparent halo is placed around the colon. (d) Rendering without shading – a smooth white halo and a black contour halo are used to indicate the three-dimensional structure resulting in a stylized cartoon-like image.

3.5 Summary

This chapter discussed techniques for the low-level abstraction of volumetric data based on common methods used in traditional illustration. Style transfer functions allow an efficient parametrization of object appearance. An intuitive and compact representation of rendering styles which enables their extraction from existing artwork allows for a flexible combination of different visual representations, ranging from highly stylized to realistic depictions. Advanced illumination effects, such as localized soft shadowing and glowing structures can be achieved using volumetric halos. This method enables subtle enhancement of small-scale structures through a detail-preserving halo generation procedure. Both techniques can take advantage of current GPUs and provide interactive performance.

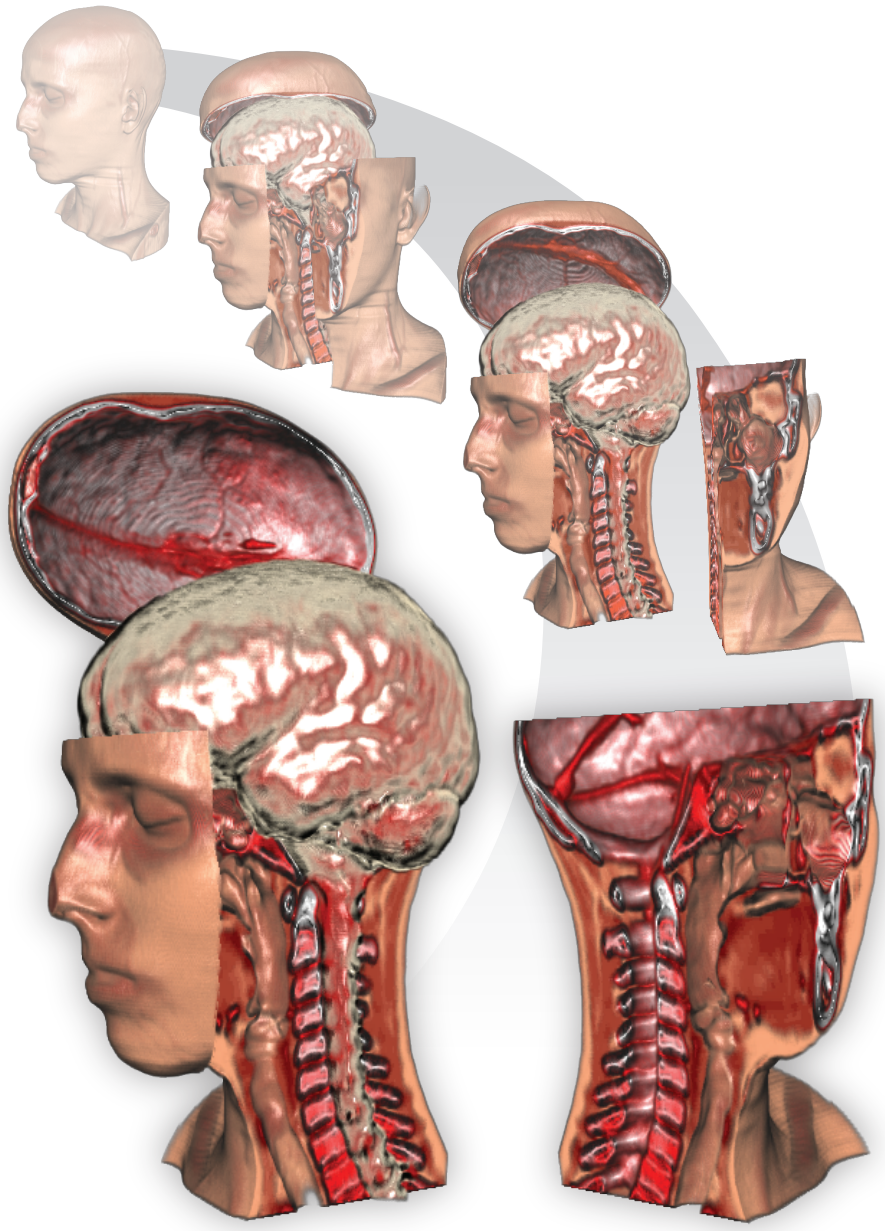


Figure 4.1 – Interactive exploded-view illustration of a human head with increasing degrees-of-explosion. Two hinge joints are used to constrain part movement.

Abstraction is real, probably more real than nature.

— *Josef Albers*

CHAPTER

4 High-Level Abstraction Techniques

Occlusion is a common problem in the visualization of volumetric data. Frequently, important features are obscured by other structures. However, completely removing these structures also eliminates the additional context information they provide. In order to solve this dilemma, illustrators employ high-level abstraction techniques, such as ghosted and exploded views, which provide human perception with sufficient cues to reconstruct a mental model of the data. This chapter presents interactive volume visualization methods based on these concepts [5, 8, 9].

4.1 Introduction

HIGH-LEVEL abstraction techniques are methods which attempt to modify object visibility according to the illustration's communicative intent. In contrast to low-level abstraction techniques, which focus on the appearance of objects, high-level techniques deal with *what* should be visible and recognizable in the resulting image. Visualization of volume data, in particular, suffers from the problem of occlusion as a three-dimensional data set is projected onto a two-dimensional image. Frequently, it is desired to simultaneously depict interior and exterior structures.

In this chapter, two techniques to solve this problem are discussed. The first method is based on ghosted views, an approach commonly employed by illustrators to visualize nested structures. In this technique, opacity is selectively reduced in areas where the information content tends to be low. A new method to identify these regions is introduced which represents an alternative to conventional clipping techniques, shares their easy and intuitive user control, but does not suffer from the drawback of missing context information. This context-preserving volume rendering model uses a function of shading intensity, gradient magnitude, distance to the eye point, and previously accumulated opacity to selectively reduce the opacity in less important data regions.

The second technique presented in this chapter is based on the concept of exploded views. Exploded views are an illustration technique where an object is partitioned into several segments which are displaced to reveal

otherwise hidden structures. While transparency or cutaways can be used to reveal a focus object, these techniques remove parts of the context information. Exploded views, on the other hand, do not suffer from this drawback. The discussed method employs a force-based model: the volume is divided into a part configuration controlled by a number of forces and constraints. The focus object exerts an explosion force causing the parts to arrange according to the given constraints. This approach automatically generates part layouts which can change dynamically based on occlusions induced by a change of the viewpoint.

4.2 Related Work

The inherent complexity of volumetric data has lead to the development of several techniques which attempt to reduce clutter and emphasize particular features. A common method is gradient-magnitude opacity-modulation. Levoy [66] proposes to modulate the opacity at a sample position using the magnitude of the local gradient. This is an effective way to enhance surfaces in volume rendering, as homogeneous regions are suppressed. Based on this idea, Ebert and Rheingans [33, 86] present several illustrative techniques which enhance features and add depth and orientation cues. They also propose to locally apply these methods for regional enhancement. Csébfalvi et al. [20] visualize object contours based on the magnitude of local gradients as well as on the angle between viewing direction and gradient vector using depth-shaded maximum intensity projection. The concept of two-level volume rendering, proposed by Hauser et al. [47, 48], allows focus+context visualization of volume data. Different rendering methods, such as direct volume rendering and maximum intensity projection, are used to emphasize objects of interest while still displaying the remaining data as context.

The concept of cutting away parts of the volume to reveal internal structures is quite common in volume visualization. Nearly every volume renderer features simple clipping operations. The basic problem of clipping planes is, however, that context information is unselectively removed when inspecting the interior of an object. There are approaches which try to remedy this deficiency by using more complex clipping geometry. Wang and Kaufman [109] introduce volume sculpting as a flexible approach for exploring volume data. The work of Weiskopf et al. [110, 111] focuses on interactive clipping operations using arbitrary geometry to overcome the limitations of common clipping planes. Konrad-Verse et al. [61] use a deformable cutting plane for virtual resection. The work of Dietrich et al. [26] consists of clipping tools for the examination of medical volume data. Zhou et al. [117] propose the use of distance to emphasize and de-emphasize different regions. They use the distance from a focal point to directly modulate the opacity at each sample position. Thus, their approach can be seen as a generalization of binary clipping.

An automated way of performing view-dependent clipping operations has been presented by Viola et al. [107, 108]. Inspired by cutaway views, which are commonly used in technical illustrations, they apply different compositing strategies to prevent an object from being occluded by a less important object. Krüger et al. [62] use interactive magic lenses based on traditional illustration techniques for focus+context visualization of iso-surfaces.

Chen et al. [14] introduce the concept of spatial transfer functions as a theoretical foundation for modeling deformations in volumetric data sets. Islam et al. [54] extend this work by using discontinuities (i.e., splitting of the volume). Some approaches employ a curve-skeleton [17] to partition volumetric structures into several segments. The curve-skeleton is a reduced representation of a volumetric object which can be generated using techniques such as volume thinning [37]. Gagvani and Silver [38] animate volume data sets using a skeleton-based approach. The interactive system presented by Singh et al. [96] allows manual editing of volume deformations based on a skeleton. They extend this work by introducing selective rendering of components for improved visualization [95]. Correa and Silver [18] use traversal of the skeleton tree to illustrate properties such as blood flow. They also present an interactive technique for volume deformation [19].

Exploded views have been investigated in the context of architectural visualization by Niedauer et al. [78]. Finally, McGuffin et al. [76] were the first to thoroughly investigate the use of exploded views for volume visualization. Their approach features several widgets for the interactive browsing of volume data partitioned into several layers using simple cuberille rendering [49].

4.3 Ghosted Views

In direct volume rendering, conceptually every single sample contributes to the final image allowing simultaneous visualization of surfaces and internal structures. However, in practice it is rather difficult and time-demanding to specify an appropriate transfer function. Due to the exponential attenuation, objects occluded by other semi-transparent objects are difficult to recognize. Furthermore, reducing opacities also results in reduced shading contributions per sample which causes a loss of shape cues, as shown in Figure 4.2 (a). One approach to overcome this difficulty is to use steep transfer functions, i.e., those with rapidly increasing opacities at particular value ranges of interest. This strengthens depth cues by creating the appearance of surfaces within the volume, but it does so by hiding all information in some regions of the volume, sacrificing a key advantage of volume rendering. Figure 4.2 (b) shows an example.

Volume clipping usually plays a decisive role in understanding volumetric data sets. It allows us to cut away selected parts of the volume, based on the position of voxels in the data set. Very often, clipping is the only way to

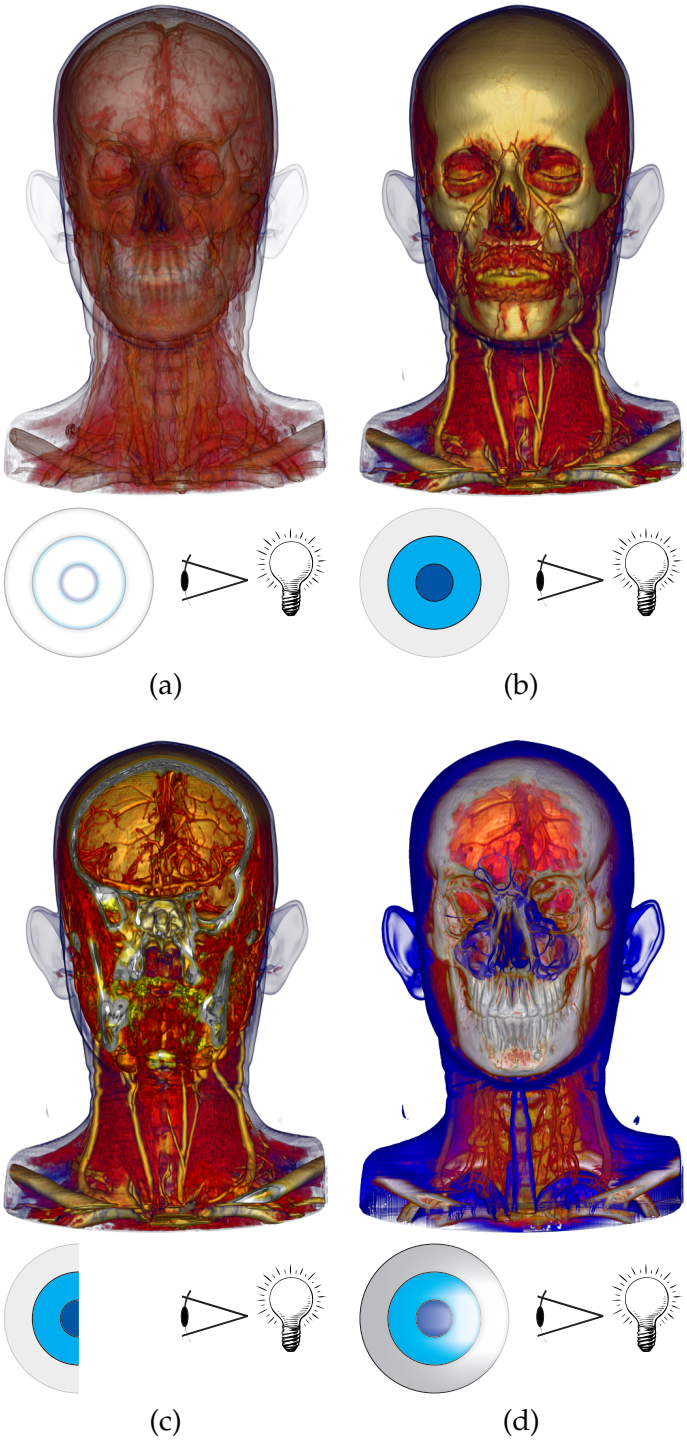


Figure 4.2 – Four different visualizations of a contrast-enhanced CT angiography data set. The illustration below each image depicts the basic opacity specification strategy. (a) Gradient-magnitude opacity-modulation. (b) Direct volume rendering. (c) Direct volume rendering with cutting plane. (d) Context-preserving volume rendering.

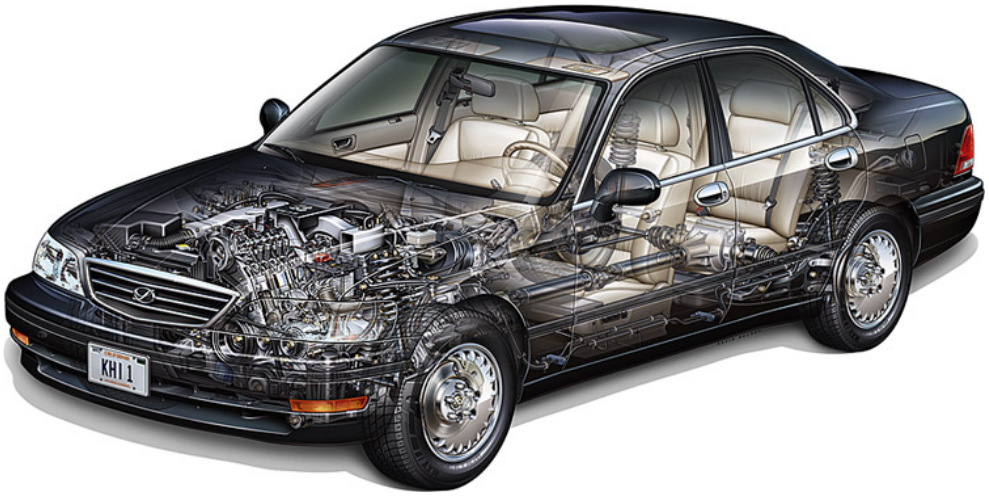


Figure 4.3 – Technical illustration using ghosting to display interior structures.

uncover important, otherwise hidden, details of a data set. These clipping operations are generally not data-aware and do not take into account features of the volume. Consequently, clipping can also remove important context information leading to confusing and partly misleading result images as displayed in Figure 4.2 (c).

In order to resolve these issues, we propose to only suppress regions which do not contain strong features when browsing through the volume. Our idea is based on the observation that large highly lit regions usually correspond to rather homogenous areas which do not contain characteristic features. While the *position* and *shape* of specular highlights, for example, provide good cues for perceiving the curvature of a surface, the area *inside* the highlight could also be used to display other information. Thus, we propose to make this area transparent allowing the user to see the interior of the volume.

In illustrations artists use *ghosting* to visualize important internal as well as external structures. Less significant items are just faintly indicated through outlines or transparency. For example, rather flat surfaces are faded from opaque to transparent to reveal the interior of an object. Detailed structures are still displayed with high opacity. An example is shown in the technical illustration in Figure 4.3: while the door is almost fully transparent, small features such as the door knob are depicted opaque. The goal of this illustration technique is to provide enough hints to enable viewers to mentally complete the partly removed structures. Unlike cutaways, which completely remove occluding objects, ghosting is less invasive in the sense that important features remain visible even when they occlude structures of primary interest.

Using our idea of lighting-driven feature classification, we can easily mimic this artistic concept in an illustrative volume rendering model. Our approach allows context-preserving and smooth clipping by adjusting the model parameters. The approach of clipping planes is extended to allow feature-aware clipping. Figure 4.2 (d) shows a result achieved with out method – the blood vessels inside the head are revealed while preserving context information. Our main contribution is a new illustrative volume rendering model which incorporates the functionality of clipping planes in a feature-aware manner. It includes concepts from artistic illustration to enhance the information content of the image. Cumbersome transfer function specification is simplified and no segmentation is required as features are classified implicitly. Our approach is especially well-suited for interactive exploration.

4.3.1 Context-Preserving Volume Rendering

Diepstraten et al. [25] cite a list of guidelines for using transparency in illustrations. For example, in many cases an illustrator chooses to reduce transparency close to the edges of transparent objects and to increase it with increasing distance to edges. Additionally, the number of overlapping transparent objects should be kept low. These rules cannot be mapped directly to the visualization of unsegmented volume data, as there is no explicit object specification. Therefore our approach relies on available quantities which are combined to achieve a similar effect.

Volume rendering algorithms usually compute a discrete approximation of the volume rendering integral along a viewing ray using the front-to-back formulation of the over operator [83] to compute opacity α_i and color c_i at each step along the ray:

$$\begin{aligned} \alpha_i &= \alpha_{i-1} + \alpha(p) \cdot (1 - \alpha_{i-1}) \\ c_i &= c_{i-1} + c(p) \cdot \alpha(p) \cdot (1 - \alpha_{i-1}) \end{aligned} \tag{4.1}$$

where $\alpha(p)$ and $c(p)$ are the opacity and color contributions at position p, α_{i-1} and c_{i-1} are the previously accumulated values for opacity and color. For conventional direct volume rendering using shading, $\alpha(p)$ and $c(p)$ are defined as follows:

$$\begin{aligned} \alpha(p) &= \alpha_{tf}(f) \\ c(p) &= c_{tf}(f) \cdot s(p) \end{aligned} \tag{4.2}$$

where α_{tf} and c_{tf} are the opacity and color transfer functions. They assign opacity and color to a sample based on the data value f at the sample point. $s(p)$ is the value of the shading intensity at the sample position. For the two-sided Blinn-Phong model using one directional light source, $s(p)$ is:

$$s(p) = (n \cdot l)\, c_d + (n \cdot h)^{c_e}\, c_s + c_a \tag{4.3}$$

where c_d, c_s, and c_a are the diffuse, specular, and ambient lighting coefficients, respectively, c_e is the specular exponent, l is the light vector, and h is the half-way vector.

Lighting plays a decisive role in illustrating surfaces. In particular, lighting variations provide visual cues regarding surface orientation. Large regions of highly illuminated material normally correspond to rather flat surfaces that are oriented towards the light source. Our idea is to reduce the opacity in these regions. Regions, on the other hand, which receive less lighting, for example light silhouettes, will remain visible. Our model uses the result of the shading intensity function for opacity modulation. Furthermore, to mimic – to a certain extent – the look and feel of a clipping plane, we also take the distance to the eye point into account. For viewing rays which have already accumulated a lot of opacity, we reduce subsequent opacity attenuation by our model since too many overlapping transparent objects make perception more difficult. As the gradient magnitude is an indicator for homogeneity of the data, it makes sense to also take this quantity into account.

Combining the available quantities in order to achieve the desired effects leads us to the following equation for the opacity at each sample position p:

$$\alpha(p) = \alpha_{tf}(f) \cdot m(p) \tag{4.4}$$

with

$$m(p) = |g|^{(\kappa_t \cdot s(p) \cdot (1-|p-e|) \cdot (1-\alpha_{i-1}))^{\kappa_s}} \tag{4.5}$$

where $|g|$ is the gradient magnitude normalized to the range $[0..1]$ (zero corresponds to the lowest and one to the hight gradient magnitude in the data set) and $s(p)$ is the shading intensity at the current sample position p. A high value of $s(p)$ indicates a highlight region and decreases opacity. The term $|p - e|$ is the distance of the current sample position p to the eye point e, normalized to the range $[0..1]$. Zero corresponds to the sample position closest to the eye point and one corresponds to the sample position farthest from the eye point. Thus, the effect of our model will decrease as distance increases. Due to the term $1 - \alpha_{i-1}$ structures located behind semi-transparent regions will appear more opaque.

Figure 4.4 illustrates the effect of these different quantities: regions with low gradient magnitude (left, top) and high shading intensity (left, center) are more likely to be suppressed. Due to the inclusion of the eye distance (left, bottom) this suppression will be strongest for regions close to the viewer. Additionally, this effect is dependent on the previously accumulated opacity (right, center). For viewing rays that already have accumulated high opacity, further opacity modulation will be reduced effectively making any further samples more opaque.

The model is controlled by the two user-specified parameters κ_t and κ_s. Through modification of these parameters the user can interactively uncover

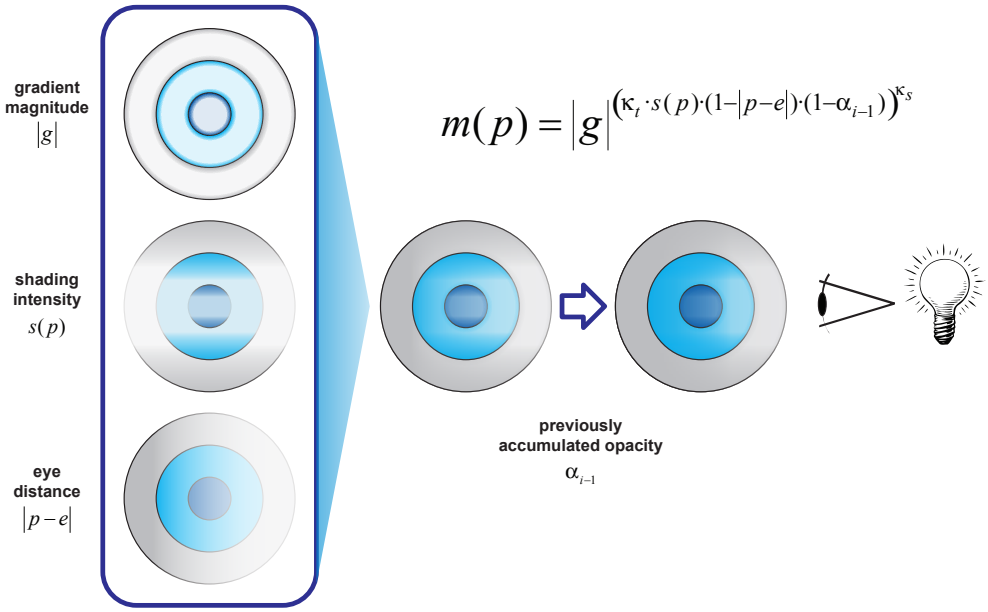


Figure 4.4 – Overview of the context-preserving volume rendering model. Gradient magnitude, shading intensity, and eye distance are combined to modulate sample opacity. Additionally, the effect is weighted by the previously accumulated opacity.

occluded regions. The parameter κ_t roughly corresponds – due to the position-dependent term $1-|p-e|$ – to the depth of a clipping plane, i.e., higher values reveal more of the interior of the volume. The effect of modifying κ_s is less pronounced; it basically controls the sharpness in transition between clipped and visible regions. Higher values will result in very sharp cuts, while lower values produce smoother transitions.

Figure 4.5 shows results for different settings of κ_t and κ_s. As κ_t increases more of the interior of the head is revealed - structures on the outside are more and more reduced to the most prominent features. An increase in κ_s causes a sharper transition between attenuated and visible regions. An interesting property of this model is that it is a unifying extension of both direct volume rendering and gradient-magnitude opacity-modulation. If κ_t is set to zero, the opacity remains unmodified and normal direct volume rendering is performed. Likewise, when κ_s is set to zero, the opacity is directly modulated by the gradient magnitude.

There are several reasons for choosing an exponential mapping in Equation 4.5. By placing the gradient magnitude in the mantissa and the modifying terms in the exponent, the highest gradient magnitudes in the data will always be preserved, as they are close to one, while lower gradient magnitudes will be more affected by the modulation. This ensures that the central role of the gradient magnitude in defining features in a data set of unknown nature is accounted for. In the exponent the term $(\kappa_t(\cdots))^{\kappa_s}$ is chosen as it allows

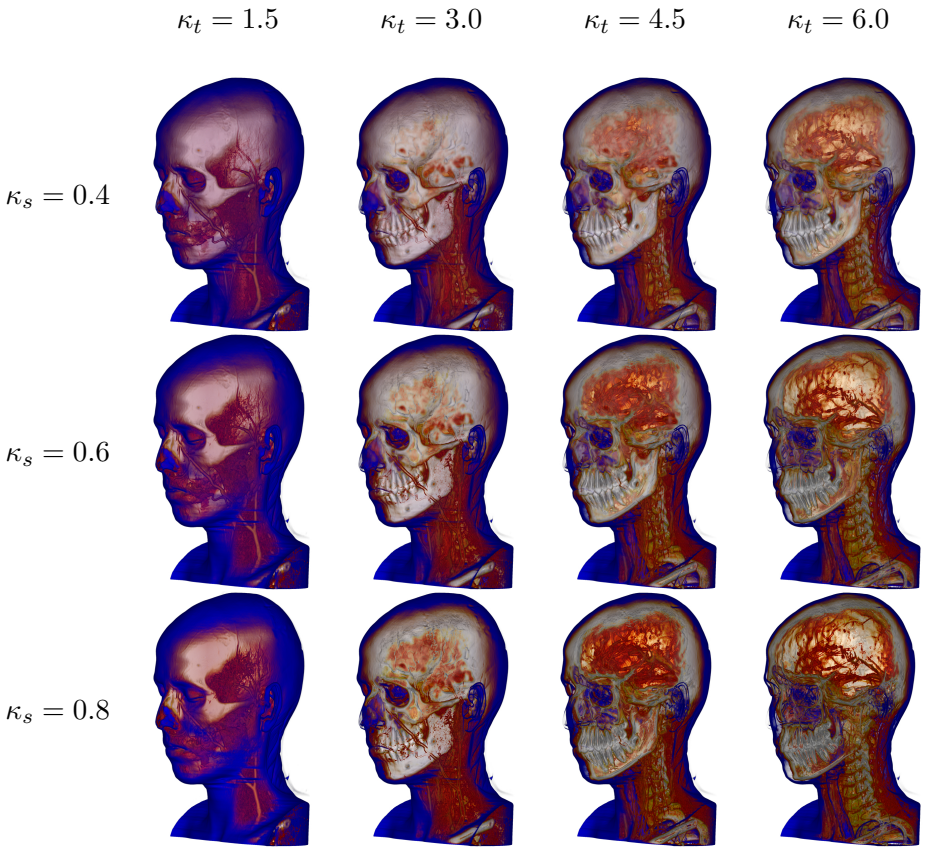


Figure 4.5 – Context-preserving volume rendering of a contrast-enhanced CT angiography data set using different values for κ_t and κ_s. Columns have the same κ_t value and rows have the same κ_s value.

fine-grained control over the shape of the function similar to Phong's specular term. While κ_t controls the basic slope, κ_s allows to adjust the curvature of this function.

4.3.2 Multiple Light Sources

While useful results can be achieved with the basic model featuring only one light source, an obvious extension is the inclusion of multiple light sources. For N light sources we assume a set of shading intensity functions $s_0, s_1, ..., s_{N-1}$ and the parameters $\kappa_{s_0}, \kappa_{s_1}, ..., \kappa_{s_{N-1}}$ and $\kappa_{t_0}, \kappa_{t_1}, ..., \kappa_{t_{N-1}}$. Any of these light sources can either exhibit context-preserving opacity-modulation or act as a conventional light source. The indices of all context-preserving light sources are members of the index set L.

We redefine Equation 4.5 to take into account all context-preserving light sources:

Figure 4.6 – Context-preserving volume rendering using two light sources. The context-preserving light source is placed in front of the face. A conventional light source illuminates the head from the left.

$$m(p) = \prod_{j \in L} |g|^{\left(\kappa_{t_j} \cdot s_j(p) \cdot (1-|p-e|) \cdot (1-\alpha_{i-1})\right)^{\kappa_{s_j}}} \tag{4.6}$$

Furthermore, we need to take the illumination contributions of all light sources into account for the final color $c(p)$ at the sample point p:

$$c(p) = c_{tf}(f) \cdot \sum_{j=0}^{N-1} s_j(p) \tag{4.7}$$

Apart from these minor modifications no other changes to the basic model are required. The advantage of using multiple light sources is that the exploration is not influenced by basic lighting settings. A typical setup might employ one conventional light source for normal illumination and one context-preserving light source used for examining the data, as shown in Figure 4.6.

4.3.3 Data-Dependent Parameters

The two parameters of our model κ_t and κ_s allow intuitive control of the visualization: κ_t is used to interactively browse through the volume, similar to the variation of the depth of a clipping plane; κ_s normally remains fixed during this process and is only later adjusted to achieve a visually pleasing result.

While we have found that these parameters provide sufficient control in most cases, a possible extension is to make them data dependent, i.e., they are defined by specifying a transfer function. This increases the flexibility of the method, but also raises the burden on the user, as transfer function specification is a complex task. Thus, we propose a hybrid solution between both approaches. We keep the global constants κ_t and κ_s, but their values are modulated by rather simple data-dependent functions. In Equation 4.5, κ_t is replaced by $\kappa_t \cdot \lambda_t(p)$ and κ_s is replaced by $\kappa_s \cdot \lambda_s(p)$. Both λ_t and λ_s are real-valued functions in the range $[0..1]$. For example, the user can specify zero for λ_t to make some regions impenetrable. Likewise, setting λ_s to zero for certain values ensures pure gradient-magnitude opacity-modulation. If one of these functions has a value of one everywhere, the corresponding global parameter remains unchanged. These modulation functions can be based on the scalar value, segmentation information (selection membership), or similar properties.

Figure 4.7 (a) shows the visible human male CT data set rendered using just the global parameters, while in Figure 4.7 (b) bone is made impenetrable by setting λ_t to zero for the corresponding values.

4.3.4 Implementation

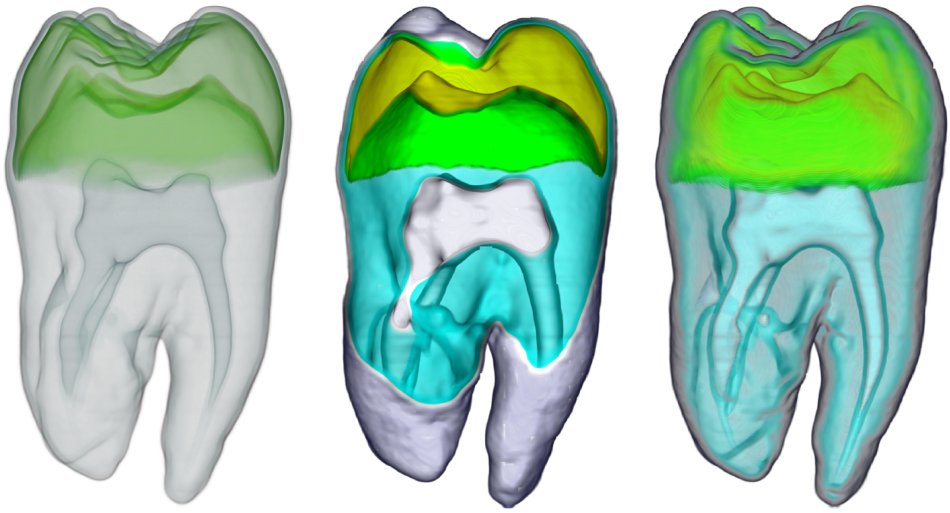
The implementation of the context-preserving volume rendering model is straight-forward, as it only requires a simple extension of the compositing routine of an existing volume rendering algorithm. The model uses only quantities which are commonly available in every volume renderer, such as gradient direction and magnitude and the depth along a viewing ray. It is well suited for implementation on current GPUs. We have also integrated this method into a high-quality software volume ray casting system for large data [41, 43].

4.3.5 Results

We experimented with the presented model using a wide variety of volumetric data sets. We have found that our approach makes transfer function specification much easier, as there is no need to pay special attention to opacity. Normally, tedious tuning is required to set the appropriate opacity in order to provide good visibility for all structures of interest. Using the context-preserving volume rendering model, we just assign colors to the structures

(a) (b)

Figure 4.7 – Context-preserving volume rendering of the visible human male CT data set. (a) Only global parameter settings are used. (b) Bone is made impenetrable by using data-dependent parameters.



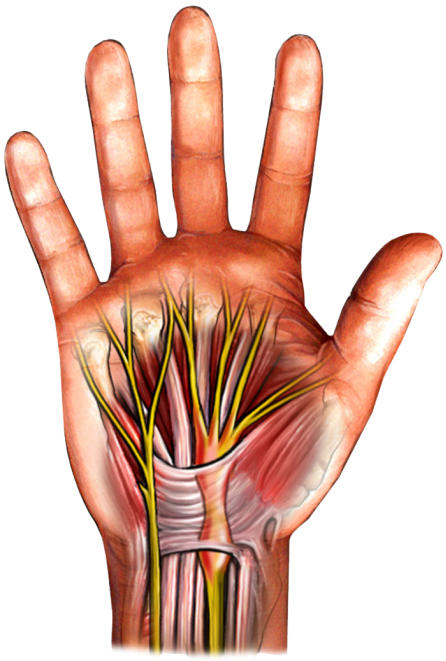
(a) (b) (c)

Figure 4.8 – CT scan of a tooth rendered with three different techniques. (a) Gradient-magnitude opacity-modulation. (b) Direct volume rendering with one clipping plane. (c) Context-preserving volume rendering.

and use the parameters κ_t and κ_s to achieve an insightful rendering. Opacity in the transfer function is just used to suppress certain regions, such as background. This contrasts the usual direct volume rendering approach, where opacity specification is vital in order to achieve the desired visual result. In many cases, however, good results are difficult and laborious to achieve with conventional methods. For example, for structures sharing the same value range, as it is often the case with contrast-enhanced CT scans, it is impossible to assign different opacities using a one-dimensional transfer function. If one object is occluding the other, setting a high opacity will cause the occluded object to be completely hidden. Using high transparency, on the other hand, will make both objects hardly recognizable. Our method inherently solves this issue, as it bases opacity not only on data values, but also includes a location-dependent term. In the following, we present some results achieved with our model in combination with a one-dimensional color transfer function. No segmentation was applied.

Figure 4.8 shows a human tooth data set rendered with gradient-magnitude opacity-modulation, direct volume rendering using a clipping plane, and the context-preserving volume rendering model using the same transfer function. Gradient-magnitude opacity-modulation shows the whole data set but the overlapping transparent structures make the interpretation of the image a difficult task. On the other hand, it is very difficult to place a clipping plane in a way that it does not remove features of interest. Using context-preserving volume rendering, the clipping depth adapts to features characterized by

(a)



(b)

Figure 4.9 – Comparing context-preserving volume rendering to illustration. (a) Context-preserving volume rendering of a hand data set. (b) Medical illustration using ghosting.

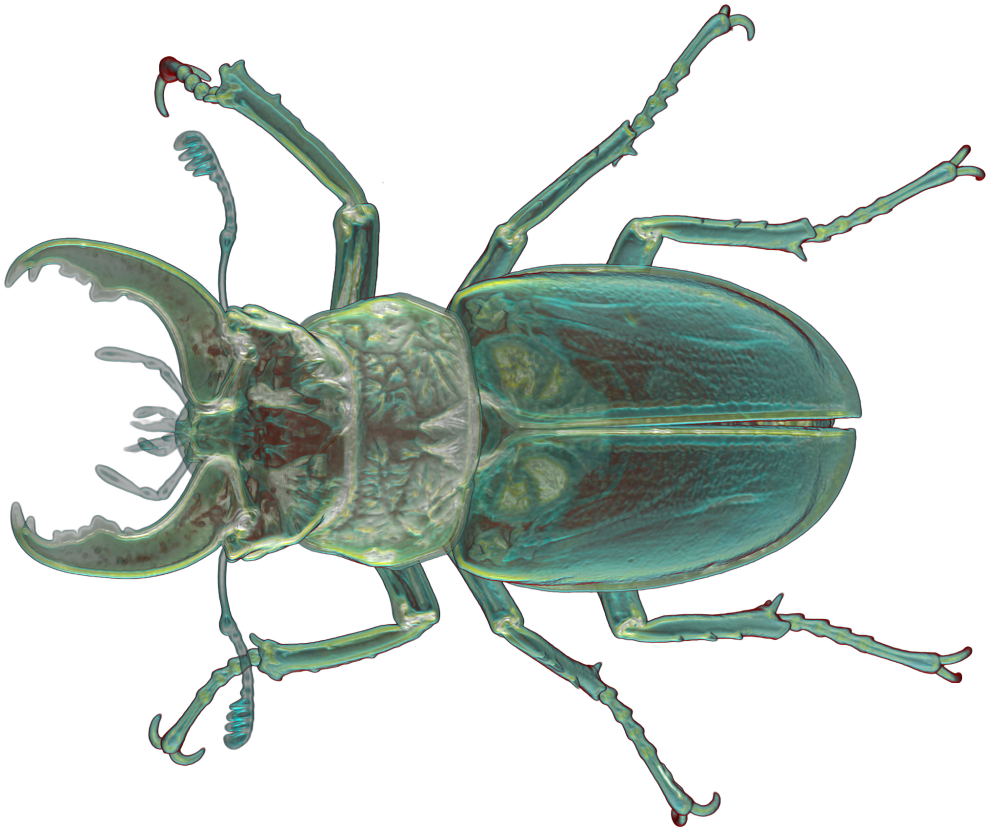


Figure 4.10 – Context-preserving volume rendering of a CT scan of a stag beetle.

varying lighting intensity or high gradient magnitude. Figure 4.9 (a) shows a CT scan of a human hand rendered using our model. The resulting image has a strong resemblance to medical illustrations using the ghosting technique, as can be seen by comparing Figure 4.9 (a) and (b). By preserving certain characteristic features, such as creases on the skin, and gradually fading from opaque to transparent, the human mind is able to reconstruct the whole object from just a few hints while inspecting the detailed internal structures. This fact is commonly exploited by illustrators for static images. For interactive exploration the effect becomes even more pronounced and causes a very strong impression of depth.

In Figure 4.10, a context-preserving volume rendering of a CT scan of a stag beetle is depicted. Highly detailed internal structures can be clearly identified while the translucent exterior provides the necessary context. The shading-dependent term in our model causes reduced transparency near the edges which serves as a cue for the relative depth of different structures.

In Figure 4.11, context-preserving volume rendering is applied to a contrast-enhanced CT scan of a human body. Two light sources are used: a conventional

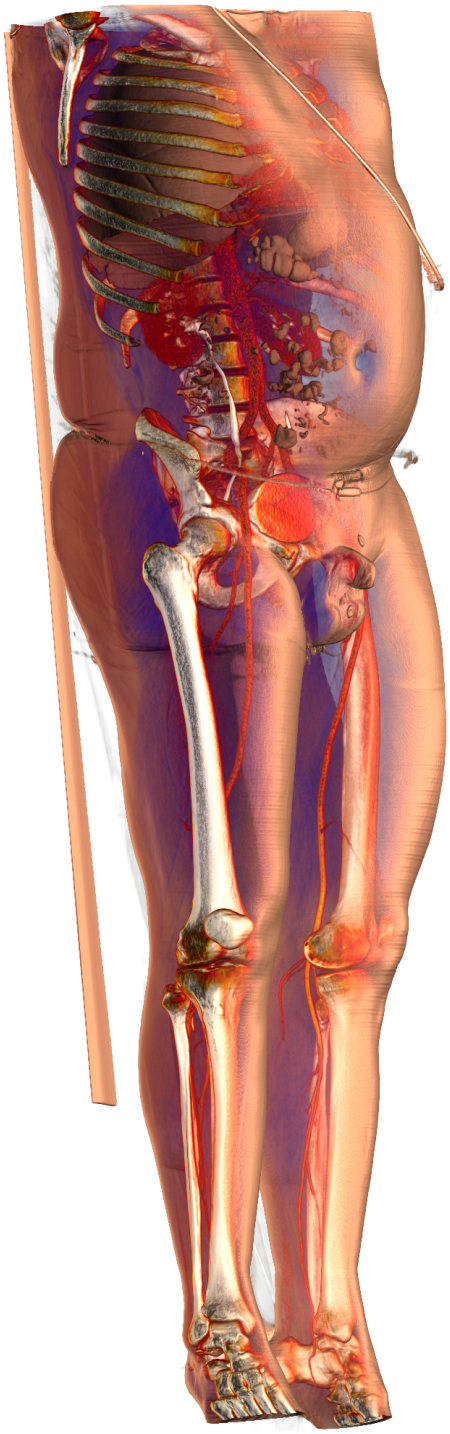


Figure 4.11 – Context-preserving volume rendering of a full-body CT angiography scan.

light source is illuminating the body from the front and a context-preserving light source is used to reveal the interior from the side. Our method allows to create a highly detailed ghosted view of the human body without any segmentation.

A typical application of traditional clipping planes is volume rendering of stacks of histological cross-sections. For such data sets it is not easily possible to assign different opacities to particular structures using transfer functions. Often it is only possible to separate the actual object from its background. Without prior segmentation, a common strategy is to display the entire volume employing clipping planes to explore its interior. As the context-preserving volume rendering model uses relative local quantities to compute the opacity of a sample, it does not suffer from the same problems as global transfer functions which assign opacities based on absolute values. The scalar gradient is replaced by an appropriate metric for photographic data, such as the color distance gradient proposed by Ebert at al. [34].

Figure 4.12 shows that our model allows exploration of color volumes without prior segmentation. Different structures are accentuated as the model parameters are modified. This approach might not completely alleviate the need for segmentation in some applications, but it can act as a tool for exploring and previewing complex spatial relationships as part of a segmentation pipeline.

One usage scenario for the presented visualization technique is an interactive segmentation application for photographic data. The unsegmented data can be visualized three-dimensionally as the segmentation is in progress. This supports the labeling of complex structures which is often difficult based on the original slices. Already segmented features can be made impenetrable (see Section 4.3.3), while the rest of the data is displayed using context-preserving volume rendering. An example is shown in Figure 4.13. With our method, effective volume visualizations of partly segmented data can be generated easily. Visual properties of already segmented objects can be controlled as usual, while visibility of the remaining data is determined through the context-preserving rendering model.

4.3.6 Discussion

Context-preserving volume rendering presents an alternative to conventional clipping techniques. It provides a simple interface for examining the interior of volumetric data sets. In particular, it is well-suited for medical data which commonly have a layered structure. Our method provides a mechanism to investigate structures of interest that are located inside a larger object with similar value ranges, as it is often the case with contrast-enhanced CT data. Landmark features of the data set are preserved. Our approach does not require any form of pre-processing, such as segmentation. One key feature of

(a)

(b)

Figure 4.12 – Using context-preserving volume rendering for exploring photographic data. (a) A low value of κ_t reveals superficial blood vessels. (b) A higher value of κ_t displays muscle and brain tissue.

(a) (b)

Figure 4.13 – Context-preserving volume rendering of the Whole Frog Project [79] photographic data set. (a) Context-preserving volume rendering of the unsegmented data. (b) Nerves have been segmented, are made impenetrable, and are artificially colored.

this model is that it effectively simplifies opacity definition during the transfer function specification.

Context-preserving volume rendering does not pose any restrictions on the type of transfer function used. The model could be applied without changes to modulate the opacity retrieved from a multi-dimensional transfer function. Likewise, the modulation functions λ_t and λ_s could be multi-dimensional. For reasons of simplicity, we have only considered simple one-dimensional transfer functions in the description.

Further degrees of freedom of the method are provided by its close connection to the illumination model. By changing the direction of a directional light source, for example, features can be interactively highlighted or suppressed, based on their orientation. Modifying the diffuse and specular factors will result in a variation of directional dependency, while adjusting the ambient component has a global influence. As means of changing these illumination properties are included in every volume visualization system, this additional flexibility will not increase the complexity of the user interface.

4.4 Exploded Views

Occlusion is an important problem when rendering truly three-dimensional information in scientific visualization, such as, for example, medical data acquired using CT or MRI. Because of occlusion, normally not all of the data can be shown concurrently. Frequently, the user wants to examine an object

of interest within the volumetric data set. In many cases depicting this focus object on its own is not sufficient – the user is interested in exploring it within the context of the whole data set. To solve the problem of occlusion the context can be assigned a different - more sparse - visual representation, for example by reducing its opacity. This adjustment can even be performed locally, so the representation only changes for those parts of the context which actually occlude the focus [8, 102, 108]. In illustrations, cutaways and ghosting techniques are used for this purpose. However, the drawback of these approaches is that parts of the context information are still removed or suppressed. If it is instructive to retain the context even when it occludes the focus structure, illustrators often employ exploded views.

Basically, in an exploded view the object is decomposed into several parts which are displaced so that internal details are visible (see Figure 4.14). This does not only give an unobstructed view on the focus but also potentially reveals other interesting information, such as cross-sections of the split object. The advantage of exploded views is that they simultaneously convey the global structure of the depicted object, the details of individual components, and the local relationships among them.

Our contribution is a new technique for generating exploded views based on a three-dimensional force-directed layout. We present an approach that is capable of producing high quality exploded depictions of volume data at interactive frame rates. One application of our method is the generation of highly detailed anatomic illustrations from scanned data (see Figure 4.1 and Figure 4.15).

4.4.1 Exploded View Generation

We present an approach for the automated generation of exploded views from volume data which does not rely on extensive object information. The technique distinguishes between focus and context using a fuzzy degree-of-interest function. Rather than manually specifying a transformation for each part of the context, an automatic technique which produces a three-dimensional layout of the parts is discussed. Our approach is also capable of re-arranging the parts dynamically based on the viewpoint. We further employ a simple interaction metaphor for specifying part geometry.

Our approach distinguishes between two basic objects derived from the volumetric data set. The *selection* (see Figure 4.16 (a)) is the current focus object specified by a selection volume. The selection volume defines a real-valued degree-of-interest function [27]. A sample of the selection volume at a specific position indicates the degree-of-interest for the corresponding data sample, where one means most interesting and zero means least interesting. The selection object comprises all data samples with non-zero degree-of-interest. The advantage of this definition is that it allows a smooth transition between focus and context.

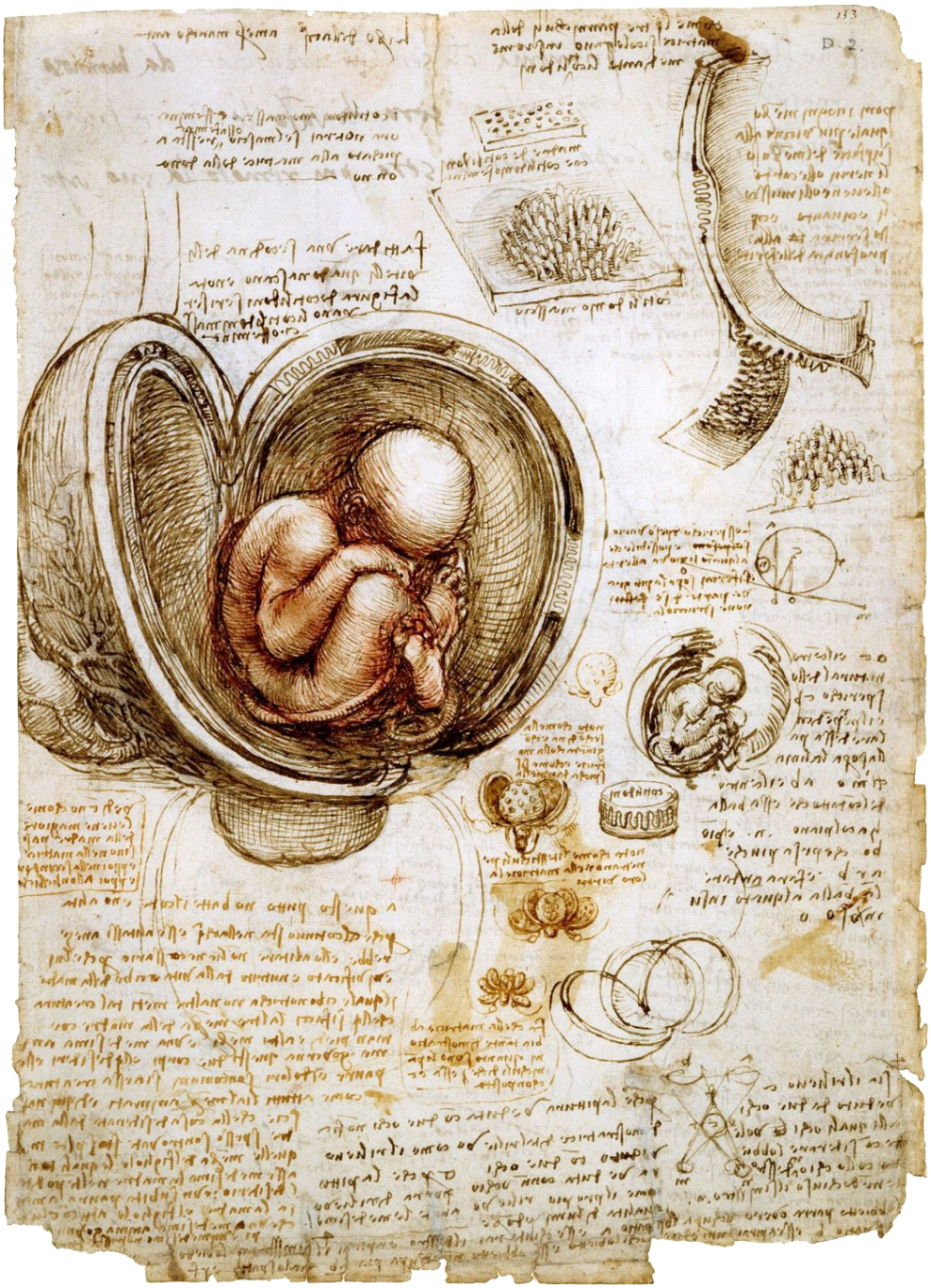


Figure 4.14 – *The Babe in the Womb* (1511) by Leonardo da Vinci is an early example for the use of exploded views – note the smaller depictions which show the use of different explosion setups.

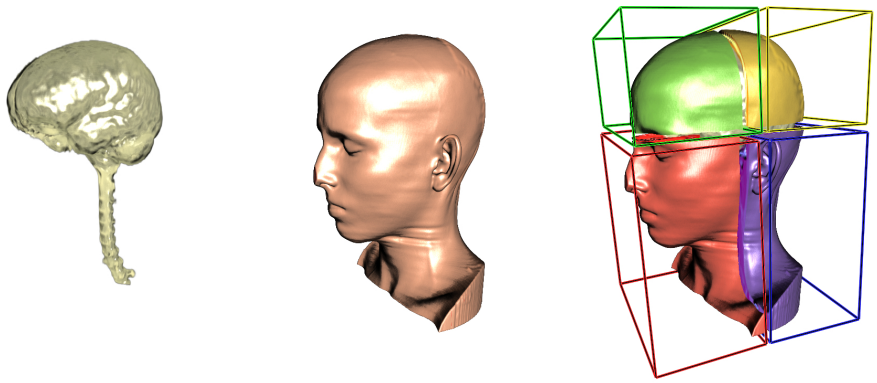
(a) (b)

Figure 4.15 – (a) Plastinated anatomic model in Gunther von Hagens' *Body-worlds*[1] exhibition. (b) Interactive exploded-view illustration generated with our approach.

Everything that is not selected is part of the *background* (see Figure 4.16 (b)) which represents the context. Segments of the background object undergo a transformation while the selection remains static. We divide the space covered by the background into an arbitrary number of non-intersecting parts P_i (see Figure 4.16 (c)). Each part is defined by its geometry and its transformation. For simplicity, we introduce the restriction that each part is convex – concave objects can be formed by grouping together several convex parts. In general, the geometry of a part does not correspond to the shape of the actual object contained in the part (which is determined by the selection volume, the data volume, and the specified transfer function). It merely bounds the space that can be occupied by this object. It is therefore sufficient to represent the part geometry by a bounding polygonal mesh. Using this setup we can generate exploded views where the parts are moved away to reveal the selection. However, it can be very tedious and time-consuming to manually specify the transformation for each part. We want a simple global mechanism to specify how "exploded" a view should be. Therefore, we introduce a *degree-of-explosion* parameter. When the degree-of-explosion is zero all parts remain untransformed. By increasing the degree-of-explosion, the user can control how much of the selection is revealed.

While it would be possible to use an ad-hoc method for displacing parts according to the degree-of-explosion, we choose to employ a force-based approach. In graph drawing, force-directed layout techniques model connectivity information through physical forces which can be simulated [32, 36]. Because of the underlying analogy to a physical system, force-directed layout methods tend to meet various aesthetic standards, such as efficient space filling, uniform edge lengths, and symmetry. They also have the advantage of

[1] `http://www.bodyworlds.com`



(a) (b) (c)

Figure 4.16 – Object setup for exploded views. (a) selection object. (b) background object. (c) background object decomposed into parts.

enabling the visualization of the layout process with smooth animations. For these reasons, we control our explosion using a rigid-body physics engine[2]. Our goal is not to simulate physical reality, which would require a far more sophisticated model including tissue properties, non-linear deformation, and many other aspects. We rather want to supply the user with a simple and intuitive interface to interactively generate exploded visualizations of volumetric data sets. New types of explosions can be generated just by adding additional forces and constraints. Furthermore, the laws of Newtonian physics are generally well understood by humans which aids comprehension of the resulting motions.

4.4.2 Part Geometry

An important step in generating an exploded view is specifying the part geometry. We provide a simple interface for rapid interactive decomposition of a volumetric data set. Our approach is based on the *splitting metaphor*: the user starts out with a single part which corresponds to the bounding box of the background object. By interactive splitting of this part along arbitrary planes as well as grouping and hiding parts the user can define complex part geometries with immediate feedback. Our interface provides three simple tools to split parts:

Axis splitter. By clicking on a point on the screen, the user splits the first part that is intersected by the corresponding viewing ray. The part is split along a plane which passes through the intersection point. Its normal is

[2] `http://www.newtondynamics.com`

the cross product between the viewing direction and the horizontal or vertical axis of the projection plane.

Depth splitter. The user clicks on a point. A viewing ray is cast which records the first intersection with the background object. The corresponding part is then split along a plane at the depth of the intersection point. The plane is parallel to the projection plane.

Line splitter. The user can draw a line segment. For each part it is determined if the projection of the part intersects the line segment. All parts which intersect the line segment are split along a plane which projects to the line.

As exploded views frequently employ splits based on object symmetry, these tools provide an intuitive way of specifying and refining part geometry. Despite the small set of operations, the concept is quite powerful as it operates in a view-dependent manner. The user can interactively rotate the volume and partition it in a natural way. In addition to this interface, our approach could straight-forwardly employ automatically defined part geometries, for example by using a pre-computed curve-skeleton.

4.4.3 Force Configuration

Force-directed layout approaches arrange elements such as the nodes of a graph by translating the layout requirements into physical forces. A simple setup uses repulsive forces between all nodes and attractive forces between nodes which are connected by an edge. A simulation is performed until the system reaches a state of minimal energy. The corresponding node positions constitute the layout. Our problem is similar. We want to arrange three-dimensional objects in such a way that they do not occlude another object, but with as little displacement as possible. Like in an atomic nucleus or a planetary system we want to achieve a steady state where the attractive forces and the repulsive forces are in equilibrium. For this reason we define the following forces based on our requirements:

Explosion force. We want to generate a force that drives the specified parts away from our selection object. The idea is to generate a force field which describes the characteristics of the selection object. Each point of the selection exerts a distance-based force on every part. As there is no analytic description of the selection, we use sampling to generate a discrete number of points which generate the force field. We will refer to these points as explosion points. In order to keep the number of explosion points low, we use an octree-based approach: We generate two min-max octrees; one for the data volume and one for the selection volume. Each node stores the minimum and maximum data and

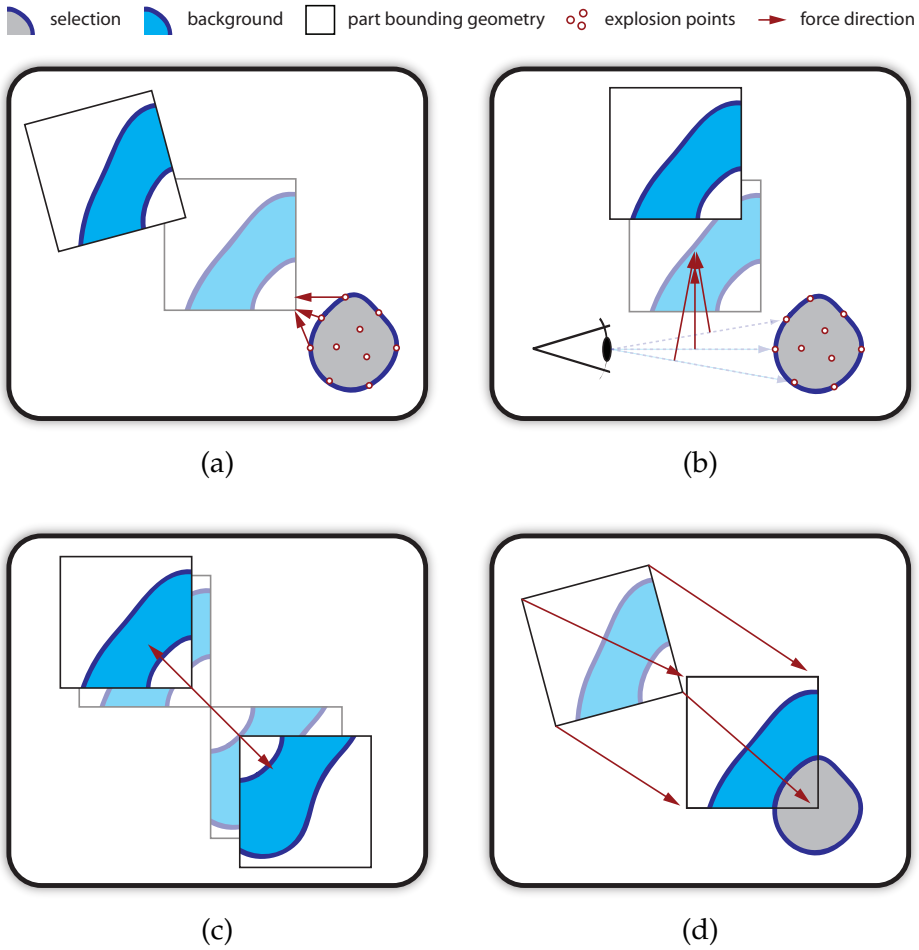
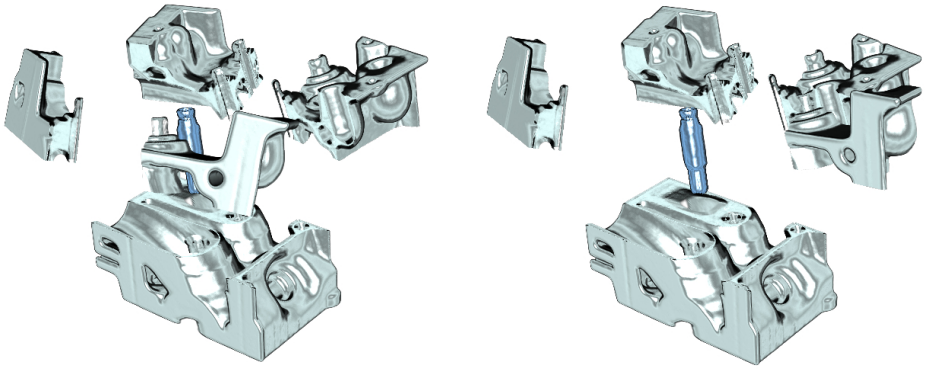


Figure 4.17 – The four different forces for our force-directed layout. (a) Explosion force. (b) Viewing force. (c) Spacing force. (d) Return force.

selection values, respectively, of the represented region. We traverse the two octrees simultaneously and generate an explosion point for each homogeneous octree node that contains both visible data values under the current transfer function and nonzero selection values. We add a small random bias to the position to prevent artifacts due to the regular structure of the octree. The explosion point is also weighted according to the size of the region corresponding to the octree node. Each explosion point exerts a force F_e on every part P_i:

$$F_e = \frac{c_e}{e^{|r|}} \cdot \frac{r}{|r|} \tag{4.8}$$

where r is the vector from the explosion point to the closest point of the part geometry of P_i and c_e is a scaling factor. The force is applied to



(a) (b)

Figure 4.18 – View-dependent exploded views. (a) Exploded view without viewing force – a part occludes the selection (dark blue). (b) Exploded view with viewing force – the occlusion is resolved.

the closest point of the part geometry and can therefore also generate a torque. The exponential fall-off is chosen to limit the force's influence to a region nearby the explosion point. The total explosion force is normalized by dividing it by the number of explosion points. The explosion force is illustrated in Figure 4.17 (a).

Viewing force. So far we have only considered view-independent explosions, i.e., the movement of parts does not take into account the current viewpoint. In traditional illustration this problem typically does not occur as the viewpoint is fixed and the exploded view is specifically generated to be most appropriate for this single viewpoint. In an interactive system, however, we must consider that the user can rotate the camera arbitrarily. For this reason we introduce a view-dependent force which attempts to arrange parts so that they do not occlude the selection for the current viewing transformation. We follow the work of Carpendale et al. [11, 12] who use similar techniques for the layout of three-dimensional graphs.

We project each of the explosion points to the image plane. For a part P_i, we cast a ray from the eye point to each explosion point and determine the point along the ray which is closest to the center of P_i. The force F_v is then:

$$F_v = \frac{c_v}{|r|} \cdot \frac{r}{|r|} \tag{4.9}$$

where r is the vector from the closest point along the viewing ray to the center of the part and c_v is a scaling factor. The total force for a part is normalized by dividing it by the number of explosion points. Figure 4.17 (b) illustrates the viewing force.

Figure 4.18 shows an example for the influence of the viewing force. In Figure 4.18 (a) the explosion force displaces the parts but disregards the viewpoint. The occlusion is resolved in Figure 4.18 (b) by adding the viewing force.

Spacing force. In order to prevent clustering of parts, we also add a repulsive force F_s. For a part P_i, the spacing force exerted by another part P_j is:

$$F_s = \frac{c_s}{|r|^2} \cdot \frac{r}{|r|} \tag{4.10}$$

where r is the vector from the center of P_j to the center of P_i and c_s is a constant scaling factor. The total spacing force for a part is normalized by dividing it by the number of parts. The spacing force is illustrated in Figure 4.17 (c).

Return force. This attractive force tries to move the parts towards their original location. It counteracts all the other forces, which generally displace a part form its initial position. Each vertex of the part geometry is connected with its original (i.e., untransformed) position. The force F_r is realized as a logarithmic spring:

$$F_r = c_r \ln(|r|) \cdot \frac{r}{|r|} \tag{4.11}$$

where r is the vector from the vertex's current position to its original location and c_r is a constant factor. The logarithmic relationship of the force's magnitude to the distance tends to produce less oscillation than the linear relationship of Hooke's law. The total return force for a part is normalized by dividing it by the number of vertices. Figure 4.17 (d) illustrates the return force.

The scaling factors of explosion force, viewing force, and spacing force, c_e, c_v, and c_s, are scaled with the global degree-of-explosion parameter, while c_r remains constant:

$$c_{\{e,v,s\}} = doe \cdot \delta_{\{e,v,s\}} \tag{4.12}$$

where doe is the degree-of-explosion and $\delta_{\{e,v,s\}} \in [0..1]$ specifies the relative contribution of the corresponding force. This allows the user to modify the influence of the individual forces, e.g. to reduce view dependency or to increase spacing. The algorithm is insensitive to changes in $\delta_{\{e,v,s\}}$. In our tests, a setting of $\delta_e = \frac{1}{2}$, $\delta_v = \frac{1}{3}$, and $\delta_s = \frac{1}{6}$ has proven to be a universally good choice. The user mostly interacts with the degree-of-explosion. Figure 4.19 shows a simple part configuration for different degrees-of-explosion.

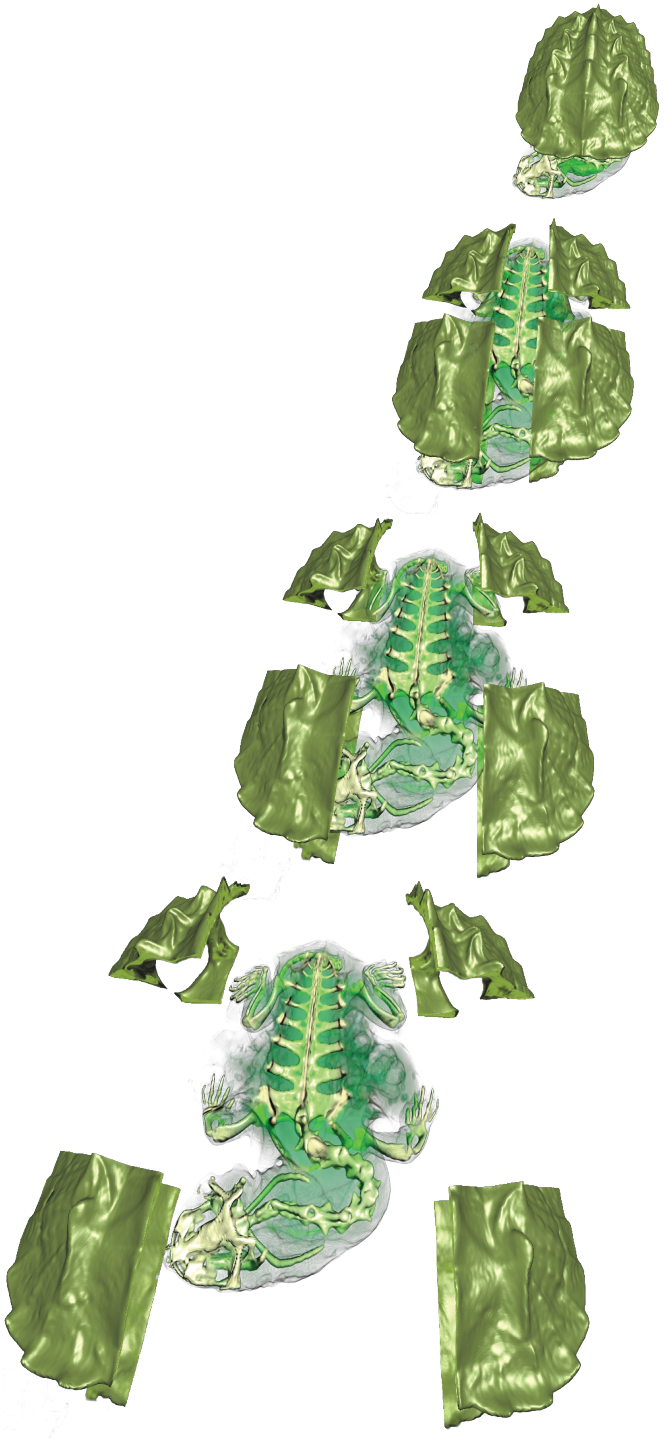


Figure 4.19 – Exploded view of a turtle with increasing degree-of-explosion from top to bottom. The body of the turtle is selected and the shell is divided into four parts.

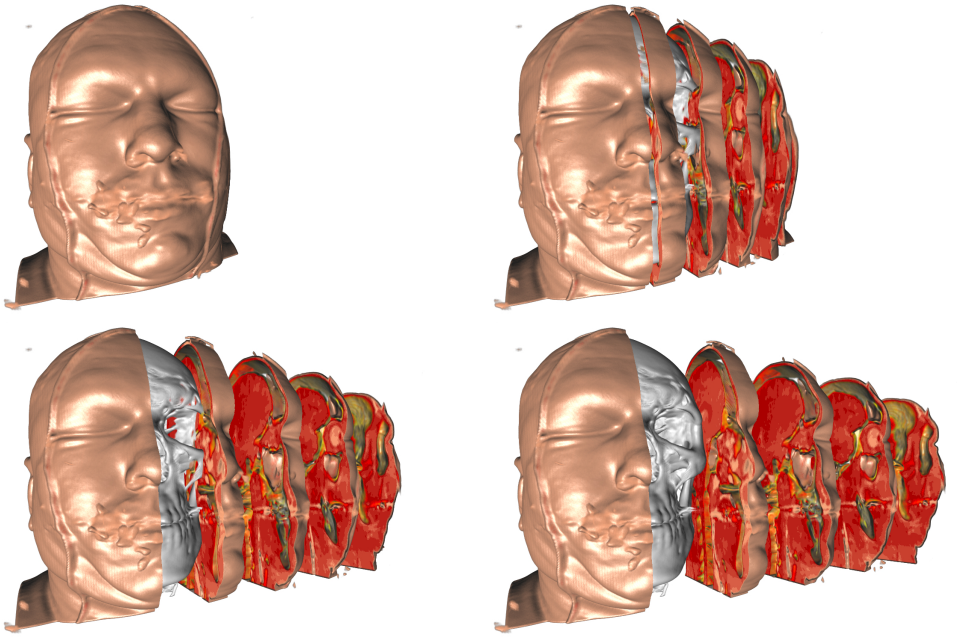


Figure 4.20 – Exploded view using constraints to limit part movement. The skull is selected. The left part of the face is static, the remaining parts are connected by a slider joint which limits their movement to a translation along one axis.

In addition to the basic forces discussed in this section, specific applications may employ further forces. For example, if available, connectivity information between certain parts could be modeled by additional spring forces.

4.4.4 Constraint Specification

While the force configuration discussed in the previous section can be used to generate expressive exploded view visualizations, it is sometimes useful to constrain the movement of parts. Therefore, our approach allows the interactive addition of joints which restrict the relative movement of parts. Available joints include sliders, hinges, ball joints, and universal joints. Additionally, the user can provide an importance for individual parts by modifying their mass. Parts with higher masses will be less affected by the individual forces and, thus, by the explosion. The user can restrict a part from being displaced by assigning an infinite mass. This is particularly useful to easily create break-away illustrations where typically only one section of the object is moved away.

An example for the use of constraints is shown in Figure 4.1 where two hinges are used. In Figure 4.20, the left part of the face has been assigned infinite mass. The right portion of the face is divided into several parts which are connected by a slider joint. As the degree-of-explosion is increased, these parts move along the free axis to reveal the skull.

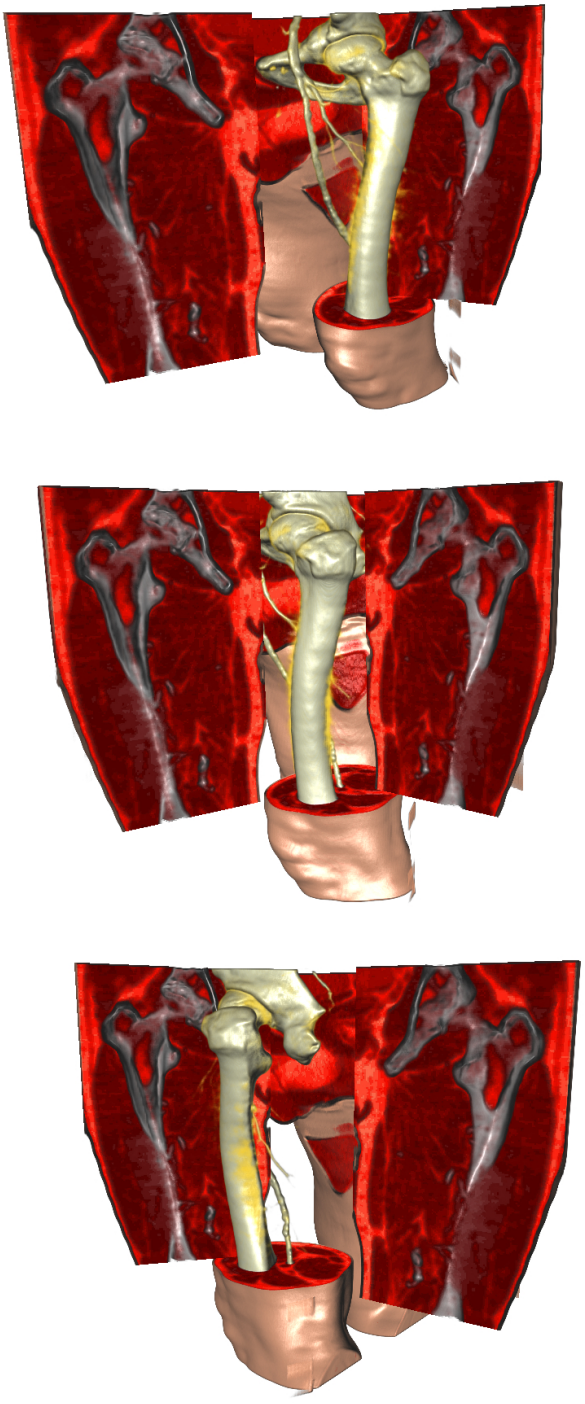


Figure 4.21 – Interaction between constraints and viewing force. All parts except the two upper portions of the leg are static. These two parts are connected by hinges similar to a swing door. As the camera rotates the viewing force causes the parts to orient themselves towards the viewer.

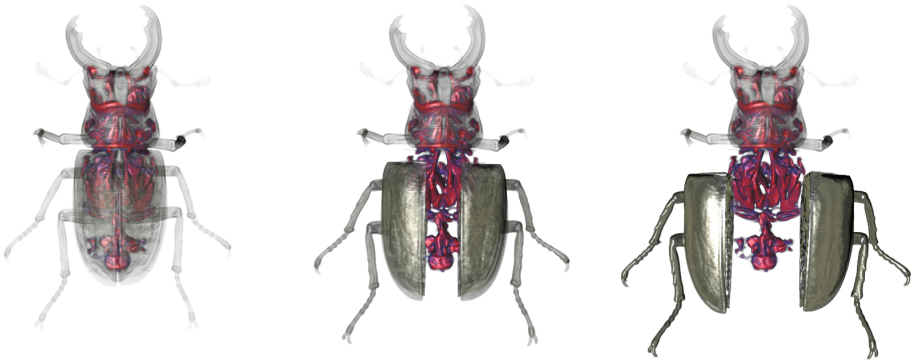


Figure 4.22 – Modulating transparency by the viewing force. As the two lower parts move away, their transparency reduces since the viewing force gets weaker. The upper part remains transparent because it is static – therefore the viewing force stays constant.

By specifying constraints the user can effectively add structural information that is missing from the raw data set. It is easily possible to generate interactive illustrations which allow exploration within the constraints specified by the designer. An interesting component in this context is the viewing force. Although the movement of a part is constrained, it is still affected by the viewing force and therefore moves within the given limits to reveal the selection. An example is shown in Figure 4.21 where two parts are connected by a hinge joint. As the camera rotates, the effect of the viewing force causes the parts to orient themselves towards the viewer.

Constraining part movements may result in arrangements with partial occlusions of the selection object. Different visual representations can be employed to resolve these conflicts. Based on the viewing force that acts on a part we can modify the sparseness of the representation, for example by modifying its transparency. An example of this behavior is shown in Figure 4.22.

4.4.5 Implementation

For rendering an exploded view we need to be able to render a volumetric data set consisting of a background and a selection object. The background object is decomposed into several non-intersecting convex parts which can have arbitrary affine transformations assigned to them. The geometry which defines these parts will be referred to as *part bounding geometry*. The selection object also has its assigned transformation and can intersect any part.

To enable empty space skipping, we further assume that we have geometry enclosing the visible volume under the current transfer function for both background and selection object (*object bounding geometry*). The use of this kind of bounding structures for empty space skipping is very common in

(a) – perform visibility sorting of the parts;
(b) – generate initial entry and exit points;
(c) – perform initial ray casting;
forall *parts* P_i *in front-to-back order* **do**
  (d) – generate entry and exit points for P_i;
  (e) – perform ray casting for P_i;
end

Figure 4.23 – Basic rendering algorithm for exploded views.

volume rendering. They are frequently based on hierarchical data structures. In our implementation, we use min-max octrees for both data volume and selection volume to enable fast updates whenever to transfer function changes.

Our GPU-based ray casting algorithm makes use of conditional loops and dynamic branching available in Shader Model 3.0 GPUs. It was implemented in C++ and OpenGL/GLSL. A basic overview is given in Figure 4.23. We start by performing a visibility sort of the parts (Figure 4.23 (a)). Next, we generate the entry and exit points for the segments of the selection located in front of any part (Figure 4.23 (b)) and perform the ray casting step for these regions (Figure 4.23 (c)). These two steps are actually simplified cases of the general iteration steps (Figure 4.23 (d) and (e)). We then iterate through the parts in front-to-back order. For each part P_i, we first establish the entry and exit points of the viewing rays for both background and selection object (Figure 4.23 (d)). Then we use this information for performing ray casting of the part (Figure 4.23 (e)). Figure 4.24 illustrates the algorithm.

Entry and Exit Point Generation

Generally, the background entry and exit buffers always contain the entry and exit points of the viewing rays for the intersection between background object bounding geometry and the part geometry. Essentially, we are using the depth buffer to perform a CSG intersection between these objects which can be simplified since the part geometry is always convex. As portions of the selection can be located in regions which are not contained in any part, the entry and exit buffers for the selection need to be generated in a slightly different way.

At startup, we generate four off-screen buffers which can be bound to a texture. For this purpose, we use the framebuffer object OpenGL extension. In these buffers we store the ray entry and exit points for both background and selection. A fragment program is bound which writes out the volume texture coordinates under the current object transformation to the red, green, and blue components and the fragment depth in viewing coordinates to the alpha component. The volume texture coordinates are later used for computing the ray direction while the depth is used in order to optimize compositing. We

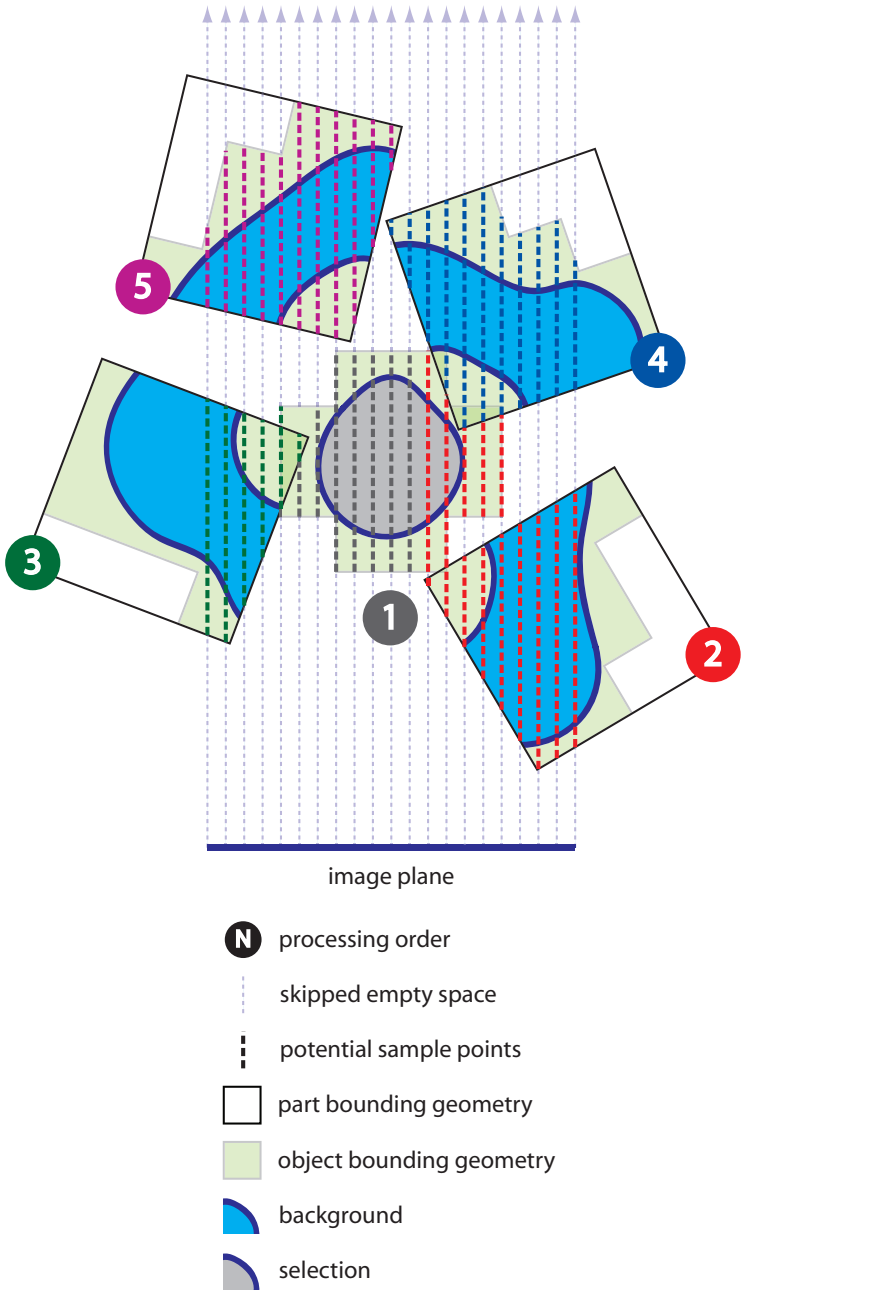


Figure 4.24 – Example of our exploded-view ray casting approach. The parts are processed in front-to-back order. Empty space skipping is performed based on object and part bounding geometries. The potential sample positions (not taking into account early ray termination) are shown for each part.

then perform the following operations to set up the entry and exit buffers for selection and background:

Background. For the exit points, the depth buffer is cleared to one and the alpha component of the color buffer is cleared to zero. Color writes are disabled. The depth test is set to ALWAYS and the front faces of the part geometry are rendered. Then color writes are enabled again, the depth test is set to GREATER, and the back faces of the background object bounding geometry are rendered. Finally, the depth test is set to LESS and the part geometry's back faces are rendered.

For the entry points, we clear the depth buffer to zero and the alpha component of the color buffer to one, disable color writes, and set the depth test to ALWAYS. Then the back faces of the part geometry are rendered. Next, color writes are enabled again, the depth test is set to LESS and the front faces of the background object bounding geometry are rendered. Finally, the depth test is set to GREATER and the front faces of the part geometry are rendered.

Selection. For the exit points the depth buffer is cleared to zero. Then the back faces of the selection object bounding geometry are rendered with the depth test set to GREATER. As it is possible that portions of the selection are not included in any part, we then set the depth test to LESS and render the front faces of all part geometries located behind the current part.

For the entry points the depth buffer is cleared to one. The depth test is set to LESS and the front faces of the selection object bounding geometry are rendered. Then the depth test is set to GREATER and the front faces of the part bounding geometry are rendered.

We also need to handle the case when portions of the selection are located in front of all parts, which is the reason why steps (b) and (c) are required in Figure 4.23. For the generation of initial entry and exit points (Figure 4.23 (b)), this is done analogously to the general iteration step (Figure 4.23 (d)) with the only difference that the background does not have to be taken into account. Thus, the selection entry points do not need to be clipped. The selection exit points are clipped against all part geometries.

Multi-Object Ray Casting

The ray casting pass uses the entry and exits points for rendering the volumetric object contained in the current part. The volume texture coordinates stored in the red, green, and blue components of the entry and exit point buffers are used to compute the ray direction. The depth value stored in the alpha component determines which objects need to be composited. If the intervals of

background and selection do not overlap, they can be composited sequentially. If they overlap, however, multi-object compositing must be performed in the intersection region, i.e., two rays have to be traversed simultaneously. The contributions of both objects at a sample point can be combined using fusion functions [10], intersection transfer functions [4], or alternating sampling [42].

The pseudocode given in Figure 4.25 shows the determination of the intervals from the entry and exit points. The functions `CompositeBackground` and `CompositeSelection` perform single volume ray casting for background and selection, respectively. `CompositeBackgroundSelection` performs multi-volume compositing. The functions `BackgroundToSelection` and `SelectionToBackground` transform between the background and the selection coordinate systems. This is necessary as the background and selection entry and exit points are given for the respective object transformation.

For the initial ray casting step (Figure 4.23 (c)), this procedure is not necessary as only the selection has to be rendered. To perform the ray casting for a part P_i (Figure 4.23 (e)), we bind a fragment program which implements the algorithm shown in Figure 4.25 and render the front faces of the part geometry. The result is blended into a framebuffer object for subsequent display.

4.4.6 Results

As the parts are non-intersecting, visibility sorting can be performed at object level rather than at primitive level. Since the number of parts will be relatively low, this step introduces practically no overhead. We use a GPU-based visibility sorting approach which employs occlusion queries [40]. For fast rendering of the object and part bounding geometry, we employ vertex buffer objects, i.e., the geometry is uploaded to GPU memory whenever it is modified (e.g., transfer function change) and can be subsequently rendered at very high frame rates.

Our ray casting shader contains dynamic branching and conditional loops which could have a significant overhead. In our benchmarks, however, we have noticed that the impact of these operations is comparably low. This might be due to the fact that there is high coherence in the branches taken between fragments and the approach therefore benefits from branch prediction. To verify this, we have compared our exploded-view renderer with a reference implementation of a conventional single-pass GPU ray caster. Both implementations use identical compositing and shading routines, but the standard ray caster ignores part transformations and the selection object. The selection object is placed inside the background object (see Figure 4.16 (a)) and the transfer function is set to a steep ramp (see Figure 4.16 (b)). For increasing numbers of parts we measured the performance for an unexploded view (i.e., the generated image is equivalent to the reference ray caster) and a fully exploded view. The results of this comparison are given in Table 4.1.

```
if b_B.depth < f_B.depth ∧ b_S.depth < f_S.depth then
    if b_S.depth < f_S.depth then
        CompositeBackground (f_B, b_B);
    else if b_B.depth < f_B.depth then
        CompositeSelection (f_S, b_S);
    else
        if f_B.depth < f_S.depth then
            if b_B.depth < f_S.depth then
                CompositeBackground (f_B, b_B);
                CompositeSelection (f_S, b_S);
            else
                f'_S = SelectionToBackground (f_S);
                CompositeBackground (f_B, f'_S);
                if b_B.depth < b_S.depth then
                    b'_B = BackgroundToSelection (b_B);
                    CompositeBackgroundSelection (f'_S, b_B, f_S, b'_B);
                    CompositeSelection (b'_B, b_S);
                else
                    b'_S = SelectionToBackground (b_S);
                    CompositeBackgroundSelection (f'_S, b'_S, f_S, b_S);
                    CompositeBackground (b'_S, b_B);
                end
            end
        else
            if b_S.depth < f_B.depth then
                CompositeSelection (f_S, b_S);
                CompositeBackground (f_B, b_B);
            else
                f'_B = BackgroundToSelection (f_B);
                CompositeSelection (f_S, f'_B);
                if b_B.depth < b_S.depth then
                    b'_B = BackgroundToSelection (b_B);
                    CompositeBackgroundSelection (f_B, b_B, f'_B, b'_B);
                    CompositeSelection (b'_B, b_S);
                else
                    b'_S = SelectionToBackground (b_S);
                    CompositeBackgroundSelection (f_B, b'_S, f'_B, b_S);
                    CompositeBackground (b'_S, b_B);
                end
            end
        end
    end
end
```

Figure 4.25 – Multi-object ray casting algorithm: f_B and b_B are the ray's entry and exit points for the background object, f_S and b_S for the selection object

parts	frames/second	
	unexploded	exploded
1	8.47 (94.4%)	7.56 (84.3%)
2	7.48 (83.4%)	7.52 (83.8%)
4	6.73 (75.0%)	6.61 (73.7%)
8	6.06 (67.6%)	5.26 (58.6%)
16	5.05 (56.3%)	4.67 (52.1%)
32	4.07 (45.4%)	3.93 (43.8%)
64	2.67 (29.8%)	2.53 (28.2%)

Table 4.1 – This table gives the frame rates for unexploded and exploded view rendering for different part counts. Numbers in brackets denote the performance as compared to the reference ray caster which achieved 8.97 frames/second. The viewport size is 512×512 with an object sample distance of 1.0. The data set dimensions are $256 \times 256 \times 166$. Transfer function and selection are specified as in Figure 4.16. Test system: Intel Pentium 4, 3.4 GHz CPU, NVidia GeForce 6800 GT GPU.

We see that our approach scales well – the frame rate drops sublinearly with the number of parts and the performance for a single part is almost identical. Considering the greatly increased flexibility of our rendering approach, we believe that these results are quite promising.

4.4.7 Discussion

Exploded views are a powerful concept for illustrating complex structures. We have presented a novel approach for generating exploded views from volumetric data sets. Our method attempts to make as little assumptions as possible while still automating laborious tasks. Instead of manually displacing parts, the user defines constraints which control the part arrangement. View-dependent explosions result in a dynamic part arrangement within the specified constraints while the user explores the object. Coupled with fast high-quality rendering, our approach for exploded-view volume-visualization features an intuitive direct manipulation interface. Possible future work includes the integration of our interactive approach with methods for automated skeleton extraction. One could imagine a system where the user can design illustration templates including joints and other constraints. This structure could then be matched with the skeleton extracted from another data set. Approaches for automatically extracting view-dependent part geometry based on concepts such as viewpoint entropy are another interesting direction for further research.

4.5 Summary

In this chapter, high-level abstraction techniques for volumetric data were discussed. High-level abstraction techniques alter object visibility according to the communicative intent of the illustration. Their aim is to provide unobstructed views of important structures while still retaining as much context as possible. The context-preserving volume rendering model is one such technique which was inspired by the common use of ghosted views in illustrations. The interior of highly lit regions is made transparent to illustrate structures behind while preserving important details. The second technique presented featured a novel approach for generating exploded views from volume data. The method attempts to make as little assumptions as possible while still automating laborious tasks. Instead of manually displacing parts, the user defines constraints which control the part arrangement. View-dependent explosions result in a dynamic part arrangement within the specified constraints while the user explores the object.

Confidence in nonsense is a requirement for the creative process.

— *Maurits Cornelis Escher*

CHAPTER

5 Summary and Conclusions

FOR centuries, the quality of an illustration heavily depended on the illustrator's ability to create a mental model of the subject. However, the complexity of many specimens makes this an elaborate and time-consuming process. This thesis proposed the use of measured data acquired by three-dimensional imaging modalities as a basis for generating illustrations. The concept of a direct volume illustration system which aims to produce results with the aesthetic quality of traditional illustrations while allowing for fully dynamic three-dimensional interaction was introduced. The aim of the presented work was to demonstrate, through the development of state-of-the-art volume visualization techniques, that it is possible and feasible to create expressive illustrations directly from volumetric data.

Abstraction, an important tool for illustrators in communicating the thematic focus to the viewer, plays a major role in the illustration process. Low-level abstraction techniques control how objects are represented. Depending on the communicative intent of an illustration, they can be used to put subtle emphasis on important structures. Two novel interactive techniques for low-level abstraction of volumetric data were introduced:

Stylized shading using style transfer functions, a new concept to parameterize visual appearance using a compact image-based representation of rendering styles, allows flexible control over the rendering of volume data sets.

Volumetric halos permit the generation of advanced illumination effects such as soft shadowing and glow in a localized manner. This technique provides an effective way of enhancing complex three-dimensional structures.

High-level abstraction techniques focus on which structures should be visible and recognizable. They are essential for the depiction of three-dimensional data as they allow to resolve conflicts between importance and visibility. Two new high-level abstraction techniques were presented:

Ghosted views simultaneously depict focus and contextual information. A context-preserving volume rendering model selectively reduces the opacity of regions with low information content to make important structures more visible.

Exploded views decompose a volume data set into several portions. These parts are automatically arranged in a three-dimensional layout that provides an unoccluded view on a focus object while retaining context information.

It was shown that abstraction techniques based on a volumetric representation integrated into a direct volume illustration framework can not only recreate the appearance of traditional illustrations, but also allow for three-dimensional interaction by exploiting the flexibility and performance of current graphics hardware. The proposed system has a wide variety of potential applications such as the production of high-quality scientific education material based on measured data and the increasingly important area of patient education. Detailed illustrations based on real patient data could be used to explain pathologies or surgical interventions to laymen.

While this thesis presented substantial progress towards these goals, several areas require further research. For instance, the investigation of alternative input technologies such as touch screens and pen interfaces might lead to more effective user interaction, particularly for novices. New metaphors are needed to integrate these input devices effectively. Furthermore, multi-user support could enable the collaborative design of illustrations.

The multitude of books is making us ignorant.

— *Voltaire*

Bibliography

[1] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3D illustrations. *Journal of the WSCG*, 13(1):1–8, 2005.

[2] A. Appel, F. J. Rohlf, and A. J. Stein. The haloed line effect for hidden line elimination. In *Proceedings of ACM SIGGRAPH 1979*, pages 151–157, 1979.

[3] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. In *Proceedings of IEEE Symposium on Volume Visualization 1998*, pages 39–46, 1998.

[4] S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005*, pages 671–678, 2005.

[5] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.

[6] S. Bruckner and M. E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351, 2007.

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

[8] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving volume rendering. In *Proceedings of EuroVis 2005*, pages 69–76, 2005.

[9] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

[10] W. Cai and G. Sakas. Data intermixing and multi-volume rendering. *Computer Graphics Forum*, 18(3):359–368, 1999.

[11] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Distortion viewing techniques for 3-dimensional data. In *Proceeding of the IEEE Symposium on Information Visualization 1996*, pages 46–53, 1996.

[12] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, 1997.

[13] P. Cavanagh. Pictorial art and vision. In R. A. Wilson and F. C. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*, pages 644–646. MIT Press, Cambridge, MA, 1999. ISBN 0471360112.

[14] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proceedings of the International Workshop on Volume Graphics 2003*, pages 35–44, 2003.

[15] J. Claes, F. Di Fiore, G. Vansichem, and F. Van Reeth. Fast 3D cartoon rendering with improved quality by exploiting graphics hardware. In *Proceedings of Image and Vision Computing New Zealand 2001*, pages 13–18, 2001.

[16] F. M. Corl, M.R. Garland, and E. K. Fishman. Role of computer technology in medical illustration. *American Journal of Roentgenology*, 175(6): 1519–1524, 2000.

[17] N. Cornea, D. Silver, and P. Min. Curve-skeleton applications. In *Proceedings of IEEE Visualization 2005*, pages 95–102, 2005.

[18] C. Correa and D. Silver. Dataset traversal with motion-controlled transfer functions. In *Proceedings of IEEE Visualization 2005*, pages 359–366, 2005.

[19] C. Correa, D. Silver, and M. Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1069–1076, 2006.

[20] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M. E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3):452–460, 2001.

[21] M. de la Flor. *The Digital Biomedical Illustration Handbook*. Charles River Media, 1st edition, 2004. ISBN 1584503378.

[22] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH 1998*, pages 189–198, 1998.

[23] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of ACM SIGGRAPH 2000*, pages 145–156, 2000.

[24] P. Desgranges, K. Engel, and G. Paladini. Gradient-free shading: A new method for realistic interactive volume rendering. In *Proceedings of Vision, Modeling, and Visualization 2005*, pages 209–216, 2005.

[25] J. Diepstraten, D. Weiskopf, and T. Ertl. Transparency in interactive technical illustrations. *Computer Graphics Forum*, 21(3):317–325, 2002.

[26] C. A. Dietrich, L. P. Nedel, S. D. Olabarriaga, J. L. D. Comba, D. J. Zanchet, A. M. Marques da Silva, and E. F. de Souza Montero. Real-time interactive visualization and manipulation of the volumetric data using GPU-based methods. In *Proceedings of Medical Imaging 2004*, pages 181–192, 2004.

[27] H. Doleisch and H. Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. *Journal of WSCG*, 10(1):147–154, 2002.

[28] F. Dong and G. J. Clapworthy. Volumetric texture synthesis for non-photorealistic volume rendering of medical data. *The Visual Computer*, 21(7):463–473, 2005.

[29] D. Dooley and M. F. Cohen. Automatic illustration of 3D geometric models: Lines. In *Proceedings of the Symposium on Interactive 3D graphics*, pages 77–82, 1990.

[30] D. Dooley and M. F. Cohen. Automatic illustration of 3D geometric models: Surfaces. In *Proceedings of IEEE Visualization 1990*, pages 307–314, 1990.

[31] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Proceedings of ACM SIGGRAPH 1988*, pages 65–74, 1988.

[32] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42: 149–160, 1984.

[33] D. S. Ebert and P. Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *Proceedings of IEEE Visualization 2000*, pages 195–202, 2000.

[34] D. S. Ebert, C. J. Morris, P. Rheingans, and T. S. Yoo. Designing effective transfer functions for volume rendering from photographic volumes. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):183–197, 2002.

[35] S. K. Feiner and D. D. Seligmann. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer*, 8 (5&6):292–302, 1992.

[36] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[37] N. Gagvani and D. Silver. Parameter-controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

[38] N. Gagvani and D. Silver. Animating volumetric models. *Graphical Models and Image Processing*, 63(6):443–458, 2001.

[39] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of ACM SIGGRAPH 1998*, pages 447–452, 1998.

[40] N. K. Govindaraju, M. Henson, M. Lin, and D. Manocha. Interactive visibility ordering and transparency computations among geometric primitives in complex environments. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games 2005*, pages 49–56, 2005.

[41] S. Grimm, S. Bruckner, A. Kanitsar, and E. Gröller. Memory efficient acceleration structures and techniques for CPU-based volume raycasting of large data. In *Proceedings of the IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics 2004*, pages 1–8, 2004.

[42] S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Gröller. Flexible direct multi-volume rendering in interactive scenes. In *Proceedings of Vision, Modeling, and Visualization 2004*, pages 386–379, 2004.

[43] S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Gröller. A refined data addressing and processing scheme to accelerate volume raycasting. *Computers & Graphics*, 28(5):719–729, 2004.

[44] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of IEEE Visualization 2003*, pages 301–308, 2003.

[45] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005.

[46] K. Hartmann, B. Preim, and T. Strothotte. Describing abstraction in rendered images through figure captions. *Electronic Transactions on Artificial Intelligence*, 3(A):1–26, 1999.

[47] H. Hauser, L. Mroz, G.-I. Bischi, and M. E. Gröller. Two-level volume rendering - fusing MIP and DVR. In *Proceedings of IEEE Visualization 2000*, pages 211–218, 2000.

[48] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3): 242–252, 2001.

[49] G. T. Herman and H. K. Liu. Three-dimensional display of human organs from computed tomograms. *Computer Graphics and Image Processing*, 9(1):1–21, 1979.

[50] J. Hladůvka, A. König, and M. E. Gröller. Curvature-based transfer functions for direct volume rendering. In *Proceedings of the Spring Conference on Computer Graphics 2000*, pages 58–65, 2000.

[51] E. R. S. Hodges, editor. *The Guild Handbook of Scientific Illustration*. John Wiley & Sons, 2nd edition, 2003. ISBN 0471360112.

[52] K. H. Höhne, M. Bomans, M. Riemer, R. Schubert, U. Tiede, and W. Lierse. A volume-based anatomical atlas. *IEEE Computer Graphics and Applications*, 12(4):72–78, 1992.

[53] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings of IEEE Visualization 1997*, pages 421–424, 1997.

[54] S. Islam, S. Dipankar, D. Silver, and M. Chen. Spatial and temporal splitting of scalar fields in volume graphics. In *Proceedings of the IEEE Symposium on Volume Visualization and Graphics 2004*, pages 87–94, 2004.

[55] M. Kersten, J. Stewart, N. Troje, and R. Ellis. Enhancing depth perception in translucent volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1117–1124, 2006.

[56] Y. Kim and A. Varshney. Saliency-guided enhancement for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):925–932, 2006.

[57] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization 2003*, pages 513–520, 2003.

[58] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of IEEE Visualization 2001*, pages 255–262, 2001.

[59] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[60] 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.

[61] O. Konrad-Verse, B. Preim, and A. Littmann. Virtual resection with a deformable cutting plane. In *Proceedings of Simulation und Visualisierung 2004*, pages 203–214, 2004.

[62] J. Krüger, J. Schneider, and R. Westermann. ClearView: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):941–948, 2006.

[63] C. H. Lee, X. Hao, and A. Varshney. Light collages: Lighting design for effective visualization. In *Proceedings of the IEEE Visualization 2004*, pages 281–288, 2004.

[64] C. H. Lee, X. Hao, and A. Varshney. Geometry-dependent lighting. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):197–207, 2006.

[65] A. Leu and M. Chen. Modelling and rendering graphics scenes composed of multiple volumetric datasets. *Computer Graphics Forum*, 18(2): 159–171, 1999.

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

[67] W. E. Loechel. The history of medical illustration. *Bulletin of the Medical Library Association*, 48(2):168–171, 1960.

[68] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH 1987*, pages 163–169, 1987.

[69] J. Loviscach. Stylized haloed outlines on the GPU. ACM SIGGRAPH 2004 Poster, 2004.

[70] A. Lu and D.S. Ebert. Example-based volume illustrations. In *Proceedings of IEEE Visualization 2005*, pages 655–662, 2005.

[71] A. Lu, C. J. Morris, D. S. Ebert, P. Rheingans, and C. Hansen. Non-photorealistic volume rendering using stippling techniques. In *Proceedings of IEEE Visualization 2002*, pages 211–218, 2002.

[72] A. Lu, C. J. Morris, J. Taylor, D. S. Ebert, C. Hansen, P. Rheingans, and M. Hartner. Illustrative interactive stipple rendering. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):127–138, 2003.

[73] T. Luft, C. Colditz, and O. Deussen. Image enhancement by unsharp masking the depth buffer. In *Proceedings of ACM SIGGRAPH 2006*, pages 1206–1213, 2006.

[74] E. B. Lum and K.-L. Ma. Lighting transfer functions using gradient aligned sampling. In *Proceedings of IEEE Visualization 2004*, pages 289–296, 2004.

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

[76] M. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization 2003*, pages 401–408, 2003.

[77] Z. Nagy, J. Schneider, and R. Westermann. Interactive volume illustration. In *Proceedings of Vision, Modeling, and Visualization 2002*, pages 497–504, 2002.

[78] C. Niederauer, M. Houston, M. Agrawala, and G. Humphreys. Non-invasive interactive visualization of dynamic architectural environments. In *Proceedings of the Symposium on Interactive 3D Graphics 2003*, pages 55–58, 2003.

[79] W. Nip and C. Logan. Whole frog technical report. Technical Report LBL-35331, University of California, Lawrence Berkeley Laboratory, 1991.

[80] Y. Ostrovsky, P. Cavanagh, and P. Sinha. Perceiving illumination inconsistencies in scenes. *Perception*, 34(11):1301–1314, 2005.

[81] S. Owada, F. Nielsen, K. Nakazawa, and T. Igarashi. A sketching interface for modeling the internal structures of 3D shapes. In *Proceedings of the International Symposium on Smart Graphics 2003*, pages 49–57, 2003.

[82] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi. Volumetric illustration: Designing 3D models with internal textures. In *Proceedings of ACM SIGGRAPH 2004*, pages 322–328, 2004.

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

[84] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, pages 581–586, 2001.

[85] B. Preim, A. Ritter, and T. Strothotte. Illustrating anatomic models - a semi-interactive approach. In *Proceedings of the International Conference on Visualization in Biomedical Computing*, pages 23–32, 1996.

[86] P. Rheingans and D. S. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.

[87] F. Ritter, C. Hansen, V. Dicken, O. Konrad, B. Preim, and H.-O. Peitgen. Real-time illustration of vascular structures. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):877–884, 2006.

[88] G. Rong and T.-S. Tan. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2006*, pages 109–116, 2006.

[89] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. In *Proceedings of ACM SIGGRAPH 1990*, pages 197–206, 1990.

[90] Z. Salah, D. Bartz, and W. Straßer. Illustrative rendering of segmented anatomical data. In *Proceedings of SimVis 2005*, pages 175–184, 2005.

[91] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):1–12, 1999.

[92] D. D. Seligmann and S. K. Feiner. Supporting interactivity in automated 3D illustrations. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 37–44, 1993.

[93] D. D. Seligmann and S. K. Feiner. Automated generation of intent-based 3D illustrations. In *Proceedings of ACM Siggraph 1991*, pages 123–132, 1991.

[94] M. Sheelagh, T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Making distortions comprehensible. In *Proceedings of the IEEE Symposium on Visual Languages 1997*, pages 36–45, 1997.

[95] V. Singh and D. Silver. Interactive volume manipulation with selective rendering for improved visualization. In *Proceedings of IEEE Symposium on Volume Visualization and Graphics 2004*, pages 95–102, 2004.

[96] V. Singh, D. Silver, and N. Cornea. Real-time volume manipulation. In *Proceedings of the International Workshop on Volume Graphics 2003*, pages 45–51, 2003.

[97] P.-P. Sloan, W. Martin, A. Gooch, and B. Gooch. The lit sphere: A model for capturing NPR shading from art. In *Proceedings of Graphics Interface 2001*, pages 143–150, 2001.

[98] M. C. Sousa and J. Buchanan. Computer-generated graphite pencil rendering of 3D polygonal models. *Computer Graphics Forum*, 18(3): 195–207, 1999.

[99] M. C. Sousa, K. Foster, B. Wyvill, and F. Samavati. Precise ink drawing of 3D models. *Computer Graphics Forum*, 22(3):369–379, 2003.

[100] S. Spitzer, M. J. Ackermann, A. L. Scherzinger, and D. Whitlock. The visible human male: A technical report. *Journal of the American Medical Informatics Association*, 3(2):118–139, 1996.

[101] A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proceedings of IEEE Visualization 2003*, pages 355–362, 2003.

[102] M. Straka, M. Cervenansky, A. La Cruz, A. Köchl, M. Sramek, M. E. Gröller, and D. Fleischmann. The VesselGlyph: Focus & context visualization in CT-angiography. In *Proceedings of IEEE Visualization 2004*, pages 385–392, 2004.

[103] T. Strothotte, editor. *Computational Visualization: Graphics, Abstraction, and Interactivity*. Springer-Verlag, 1st edition, 1998. ISBN 3540637370.

[104] N. Svakhine, D. S. Ebert, and D. Stredney. Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics and Applications*, 25(3):31–39, 2005.

[105] N. A. Svakhine and D. S. Ebert. Interactive volume illustration and feature halos. In *Proceedings of the Pacific Conference on Computer Graphics and Applications 2003*, pages 347–354, 2003.

[106] C. Tietjen, T. Isenberg, and B. Preim. Combining silhouettes, surface, and volume rendering for surgery education and planning. In *Proceedings of EuroVis 2005*, pages 303–310, 2005.

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

[108] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[109] S. W. Wang and A. E. Kaufman. Volume sculpting. In *Proceedings of the Symposium on Interactive 3D Graphics 1995*, pages 151–156, 1995.

[110] D. Weiskopf, K. Engel, and T. Ertl. Volume clipping via per-fragment operations in texture-based volume visualization. In *Procceedings of IEEE Visualization 2002*, pages 93–100, 2002.

[111] D. Weiskopf, K. Engel, and T. Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):298–312, 2003.

[112] A. Wenger, D. F. Keefe, and S. Zhang. Interactive volume rendering of thin thread structures within multivalued scientific data sets. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):664–672, 2004.

[113] B. Wilson, E. B. Lum, and K.-L. Ma. Interactive multi-volume visualization. In *Proceedings of the International Conference on Computational Science 2002*, pages 102–110, 2002.

[114] R. Yagel, A. Kaufman, and Q. Zhang. Realistic volume imaging. In *Proceedings of IEEE Visualization 1991*, pages 226–231, 1991.

[115] X. Yuan and B. Chen. Illustrating surfaces in volume. In *Proceedings of Joint IEEE/EG Symposium on Visualization 2004*, pages 9–16, 2004.

[116] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[117] J. Zhou, A. Döring, and K. D. Tönnies. Distance based enhancement for focal region based volume rendering. In *Proceedings of Bildverarbeitung für die Medizin 2004*, pages 199–203, 2004.

A life spent making mistakes is not only more honorable, but more useful than a life spent doing nothing.

— *George Bernard Shaw*

Curriculum Vitae

Contact Information

Name	Stefan Bruckner
Address	Castellezgasse 20/1/1, 1020 Wien, Austria
Phone	+43 676 671 2743
E-Mail	bruckner@cg.tuwien.ac.at

Personal Details

Date of Birth	June 2nd, 1980
Place of Birth	Oberwart, Austria
Citizenship	Austrian
Gender	Male
Languages	German (native), English (fluent)

Education

Sep. 1986 - Jun. 1990	Primary School Loipersdorf-Kitzladen, Austria
Sep. 1990 - Jun. 1994	Secondary School Oberschützen, Austria
Sep. 1994 - Jun. 1999	Federal Higher Technical Institute Pinkafeld, Austria **Secondary school education (computing and organization).** Successfully completed the final project *Hierarchical Data Interface for Geodesy-Software*. Graduation with highest distinction.
Oct. 1999 - Jun. 2004	Vienna University of Technology, Austria **Master's studies (computer science).** Successfully completed the master's thesis *Efficient Volume Visualization of*

Large Medical Datasets at the Institute of Computer Graphics and Algorithms. Advisors: Dr. Sören Grimm, Prof. Eduard Gröller. Graduation with highest distinction.

since Jul. 2004 — Vienna University of Technology, Austria
Doctoral studies (computer science). Working on the dissertation *Interactive Illustrative Volume Visualization* at the Institute of Computer Graphics and Algorithms. Advisor: Prof. Eduard Gröller.

Employment History

Summer 1998 — rmDATA GmbH, Austria
Intern. Design and implementation of a printing system for measurement data.

Summer 1999 — rmDATA GmbH, Austria
Intern. Development of an interactive editor for measurement data using an object-oriented database interface.

Summer 2000 — Efficient Marketing GmbH, Austria
Intern. Design and implementation of a web-based front-end for data-warehousing applications.

Summer 2001 — UPC Telekabel GmbH, Austria
Intern. Migration of database applications and development of user-interfaces.

Oct. 2003 - Jun. 2004 — Vienna University of Technology, Austria
Teaching assistant. Supervision of student courses at the Institute of Computer Graphics and Algorithms.

since Jul. 2004 — Vienna University of Technology, Austria
Research assistant. Research and teaching at the Institute of Computer Graphics and Algorithms.

Nov. 2005 - Jan. 2006 — Biotronics3D Ltd., United Kingdom
Consultant. Design and supervision of the implementation of a high-performance rendering system for virtual endoscopy.

Additional Qualifications

Jun. 1997 — Cambridge First Certificate in English

Dec. 1998 — Quality Austria Quality Techniques QII Certificate

Awards and Honors

Apr. 2004	Best Paper Award and Best Presentation Award at the Central European Seminar on Computer Graphics 2004
Mar. 2006	Karl-Heinz-Höhne Award for Medical Visualization 2006
Sep. 2007	3rd Best Paper Award at Eurographics 2007

Professional Activities

Contributor to a lecture series on illustrative visualization at several international conferences:

ACM SIGGRAPH 2006	Course on Illustrative Visualization for Medicine and Science
IEEE Visualization 2006	Tutorial on Illustrative Visualization for Science and Medicine
IEEE Visualization 2007	Tutorial on Illustrative Display and Interaction in Visualization
Eurographics 2008	Tutorial on Interactive Tools for Scientific and Medical Illustration Composition

Referee for international journals and conferences in the area of visualization and computer graphics:

Journals	IEEE Transactions on Visualization and Computer Graphics, Computer Graphics Forum, Computers & Graphics
Conferences	IEEE Visualization, Eurographics, EuroVis, Volume Graphics, VMV, IEEE Virtual Reality, Pacific Graphics, SimVis, Computational Aesthetics, WSCG, CGIV

Publications

S. Bruckner, D. Schmalstieg, H. Hauser, and M. E. Gröller. The inverse warp: Non-invasive integration of shear-warp volume rendering into polygon rendering pipelines. In *Proceedings of Vision, Modeling, and Visualization 2003*, pages 529–536, 2003.

S. Bruckner. Efficient volume visualization of large medical datasets. In *Proceedings of the Central European Seminar on Computer Graphics 2004*, 2004. Best paper award.

S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Gröller. A refined data addressing and processing scheme to accelerate volume raycasting. *Computers & Graphics*, 28(5), 2004.

S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Gröller. Vots: Volume dots as a point-based representation of volumetric data. *Computer Graphics Forum*, 23 (4), 2004.

S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Gröller. Memory efficient acceleration structures and techniques for CPU-based volume raycasting of large data. In *Proceedings of the Symposium on Volume Visualization and Graphics 2004*, 2004.

S. Grimm, S. Bruckner, A. Kanitsar, and M. E. Gröller. Flexible direct multi-volume rendering in interactive scenes. In *Proceedings of Vision, Modeling, and Visualization 2004*, pages 386–379, 2004.

E. Coto, S. Grimm, S. Bruckner, M. E. Gröller, A. Kanitsar, and O. Rodriguez. MammoExplorer: An advanced CAD application for breast DCE-MRI. In *Proceedings of Vision, Modeling, and Visualization 2005*, pages 91–98, 2005.

S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005*, pages 671–678, 2005.

S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving volume rendering. In *Proceedings of EuroVis 2005*, pages 69–76, 2005.

S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.

P. Rautek, B. Csebfalvi, S. Grimm, S. Bruckner, and M. E. Gröller. D^2VR: High quality volume rendering of projection-based volumetric data. In *Proceedings of EuroVis 2006*, pages 211–218, 2006.

P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. Evaluation of a bricked volume layout for a medical workstation based on java. *Journal of WSCG*, 15(1-3):83–90, 2007.

S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007. 3rd Best Paper Award at Eurographics 2007.

P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. LiveSync: Deformed viewing spheres for knowledge-based navigation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1544–1551, 2007.

P. Rautek, S. Bruckner, and M. E. Gröller. Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1336–1343, 2007.

S. Bruckner and M. E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13 (6):1344–1351, 2007.