Institut für Computergraphik und Algorithmen
Technische Universität Wien
Karlsplatz 13/186/2
A-1040 Wien
AUSTRIA
Tel: +43 (1) 58801-18601
Fax: +43 (1) 58801-18698

Institute of Computer Graphics and Algorithms
Vienna University of Technology
email:
technical-report@cg.tuwien.ac.at
other services:
http://www.cg.tuwien.ac.at/
ftp://ftp.cg.tuwien.ac.at/

TECHNICAL REPORT

VolumeShop: An Interactive System for Direct Volume Illustration

Stefan Bruckner M. Eduard Gröller

TR-186-2-05-04
April 2005

Keywords: illustrative visualization, volume rendering, focus+context techniques

VolumeShop: An Interactive System for Direct Volume Illustration

Stefan Bruckner M. Eduard Gröller*

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria

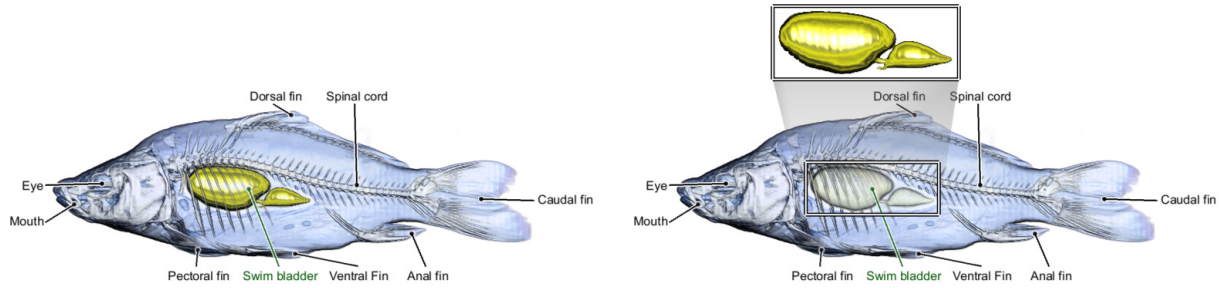


Figure 1: Annotated direct volume illustrations of a carp. (a) The swim bladder is highlighted using cutaways and ghosting. (b) The swim bladder is displayed enlarged.

Abstract

Illustrations play a major role in the education process. Whether used to teach a surgical or radiologic procedure, to illustrate normal or aberrant anatomy, or to explain the functioning of a technical device, illustration significantly impacts learning. Although many specimen are readily available as volumetric data sets, particularly in medicine, illustrations are commonly produced manually as static images in a time-consuming process. Our goal is to create a fully dynamic three-dimensional illustration environment which directly operates on volume data. Single images have the aesthetic appeal of traditional illustrations, but can be interactively altered and explored. In this paper we present methods to realize such a system which combines artistic visual styles and expressive visualization techniques. We introduce a novel concept for direct multi-object volume visualization which allows to control the appearance of inter-penetrating objects via two-dimensional transfer functions. Furthermore, a unifying approach to efficiently integrate many non-photorealistic rendering models is presented. We discuss several illustrative concepts which can be realized by combining cutaways, ghosting, and selective deformation. Finally, we also propose a simple interface to specify objects of interest through three-dimensional volumetric painting. All presented methods are integrated into VolumeShop, an interactive hardware-accelerated application for direct volume illustration.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms

Keywords: illustrative visualization, volume rendering, focus+context techniques

1 Introduction

A considerable amount of research 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 we base our paper on this large body of research, our 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

*e-mail: {bruckner | groeller}@cg.tuwien.ac.at

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 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. [4] 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. 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, a specific pathology. 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 paper is subdivided as follows: In Section 2 we discuss related work. Section 3 gives a conceptual overview of our approach. In Sections 4, 5, and 6, we cover in detail the three fundamental building blocks of our direct volume illustration system, multi-object volume rendering, illustrative enhancement, and selective illustration, respectively. Section 7 discusses strategies and results for an efficient implementation of the presented concepts. The paper is concluded in Section 8.

2 Related Work

Non-photorealistic or illustrative rendering methods are a very active field of research. In volume visualization, Levoy [14] was the first to propose modulation of opacity 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, Rheingans and Ebert [19] present several illustrative techniques which enhance features and add depth and orientation cues. They also propose to locally apply these methods for regional enhancement. Using similar methods, Lu et al. [15] developed an interactive volume illustration system that simulates traditional stipple drawing. Csébfalvi et al. [5] 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. Lum and Ma [16] present a hardware-accelerated approach for high-quality non-photorealistic rendering of volume data. Exploring the variety of traditional illustration styles, selective emphasis of certain structures is an important technique. The concept of two-level volume rendering, proposed by Hauser et al. [8], allows focus+context visualization of volume data. Different rendering styles, such as direct volume rendering and maximum intensity projection, are used to emphasize objects of interest while still displaying the remaining data as context. Methods for combining multiple volume data sets have been investigated in the context of multi-modal data. For instance, Cai and Sakas [2] discuss different methods for data intermixing in volume rendering. Wilson et al. [24] propose a hardware-accelerated algorithm for multi-volume visualization. Leu and Chen [13] present a system for modeling scenes consisting of multiple volumetric objects which is restricted to non-intersecting volumes. The approach by Grimm et al. [7] uses alternating sampling for combining multiple volumes in dynamic scenes. An automated way of performing clipping operations has been presented by Viola et al. [22]. Inspired by cut-away 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. Konrad-Verse et al. [10] perform clipping using a mesh which can be flexibly deformed by the user with an adjustable sphere of influence. Zhou et al. [26] propose the use of distance to emphasize and de-emphasize different regions. Lum an Ma [17] use two-dimensional scalar-based lighting transfer functions to enhance material boundaries using illumination. Volume sculpting, proposed by Wang and Kaufman [23], enables interactive carving of volumetric data. Islam et al. [9] discuss

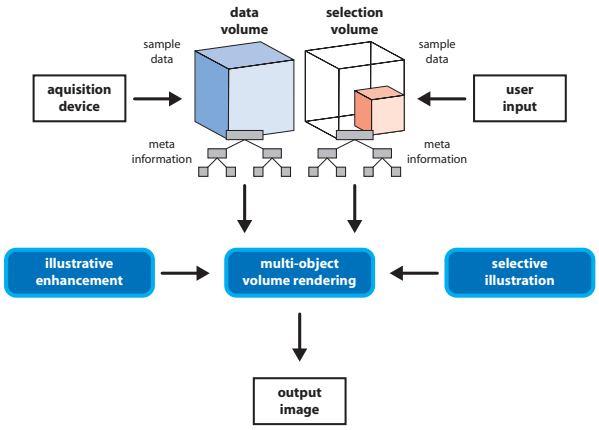


Figure 2: Conceptual overview of our direct volume illustration environment.

methods for spatial and temporal splitting of volume data sets.

3 Overview

The architecture of VolumeShop, our direct volume illustration system, discriminates between two basic types of volumes: data volumes and selection volumes. A data volume stores the actual scalar field, for example acquired by a CT scanner. 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". While both, multiple data and selection volumes can be defined, only one pair is active at a time. Both 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 volume due to its sparse nature. Furthermore, several pieces of meta information (e.g., min-max octrees, bounding boxes, and transformations) are stored for both volumes and updated on modification. This allows, for instance, the quick extraction of tight bounding volumes, which are used to skip empty space during rendering. At the heart of the system lies a multi-object volume rendering algorithm which is responsible for the concurrent visualization of multiple user-defined volumetric objects. It makes use of illustrative enhancement methods and selective illustration techniques defining the visual appearance of objects. A conceptual overview of this interaction is given is Figure 2. In the following sections, we will describe each of these components in detail.

4 Multi-Object Volume Rendering

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. We also want to permit arbitrary intersections between objects and control how the intersection regions are visualized.

Our 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 original location of the selection, 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 is 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 a value in $[0,1]$ which indicates 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 3, first row) with their respective membership functions μ_S, μ_G, and μ_B:

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

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

To control the appearance of our three objects, selection, ghost, and background, we define color and opacity transfer functions based on the values of f_V, 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 3, second row):

$$\begin{aligned} \mu_{V_S}(p) &= \alpha_S(f_V(T(p))) \\ \mu_{V_G}(p) &= \alpha_G(f_V(p)) \\ \mu_{V_B}(p) &= \alpha_B(f_V(p)) \end{aligned} \tag{2}$$

For each of our three objects we can now define an *objectset* as the intersection between the corresponding selection and volume set (see Figure 3, third row):

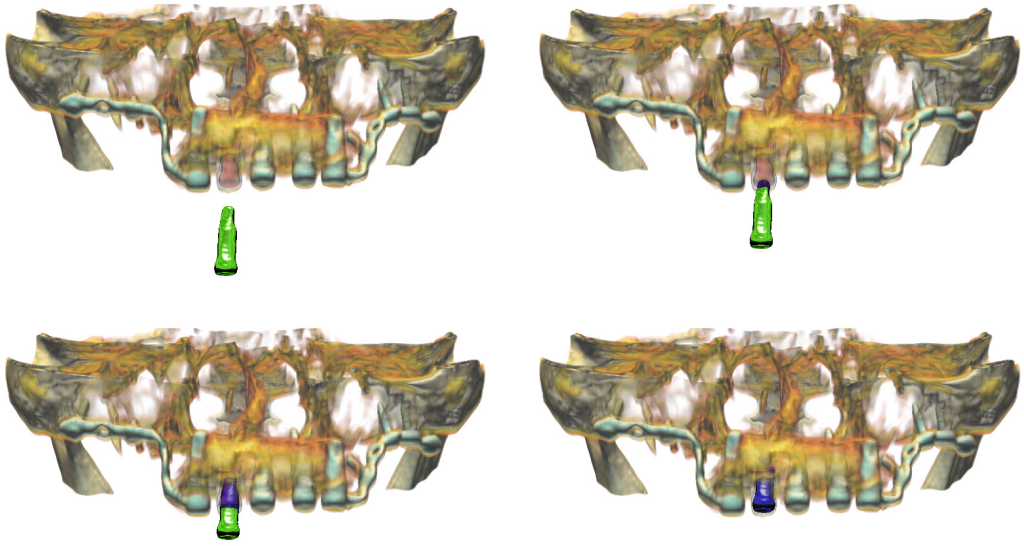


Figure 4: Using intersection transfer functions to illustrate implant placement in the maxilla. As the selection (green) is moved into the ghost (faint red), the intersection transfer function causes it to be displayed in blue.

$$\begin{array}{rcl} S & = & S_S \cap V_S \\ G & = & S_G \cap V_G \\ B & = & S_B \cap V_B \end{array} \tag{3}$$

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 visa 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 at a point p as a weighted sum using the respective membership functions as weights:

$$c(p) \quad = \quad \frac{\sum_{i=1}^{n} c_i(p) \cdot \mu_i(p)}{\sum_{i=1}^{n} \mu_i(p)} \tag{4}$$

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:

$$\alpha(p) \quad = \quad \mu_{X_1 \cup X_1 \cup \ldots \cup X_n}(p) \tag{5}$$

Using Equations 4 and 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. Frequently, for example, illustrators emphasize inter-penetrating 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 for color and opacity. Thus, we define two new sets V_{GS} and V_{BS} with the following membership functions:

$$\begin{array}{rcl} \mu_{V_{GS}}(p) & = & \alpha_{GS}(f_V(p), f_V(T(p))) \\ \mu_{V_{BS}}(p) & = & \alpha_{BS}(f_V(p), f_V(T(p))) \end{array} \tag{6}$$

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:

$$\begin{array}{rcl} \mu_{\widehat{GS}}(p) & = & \begin{cases} 0 & \mu_{GS}(p) = 0 \\ \mu_{S_G \cap S_S \cap V_{GS}}(p) & otherwise \end{cases} \\ \mu_{\widehat{BS}}(p) & = & \begin{cases} 0 & \mu_{BS}(p) = 0 \\ \mu_{S_B \cap S_S \cap V_{BS}}(p) & otherwise \end{cases} \end{array} \tag{7}$$

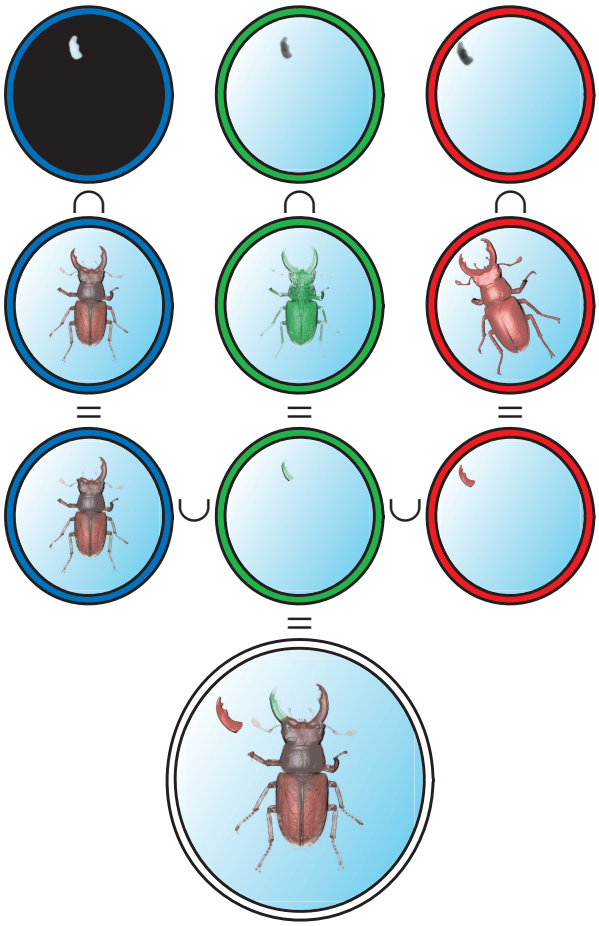


Figure 3: Overview of the basic multi-object combination process for background (left column), ghost (middle column), and selection (right column): the intersection between selection sets (first row) and volume sets (second row) results in object sets (third row) which are then combined.

To evaluate the combined color and opacity at a point p in space now, we use Equation 4 and 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 is used for the intersection and the maximum operator for the union of two fuzzy sets [25].

The intersection transfer functions can be used to control the color and opacity in the region of intersection between two objects based on the scalar values of both objects. In our implementation we provide a default setting which is a opacity-weighted average between the one-dimensional color transfer functions of the two respective objects (background and selection, or ghost and selection). Further, we provide presets where the opacity is computed from the one-dimensional opacity transfer functions by one of the compositing operators derived by Porter and Duff [18]. The color can be specified arbitrarily. Additionally, the user can paint on the two-dimensional function using a gaussian brush to highlight specific scalar ranges. Figure 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.

While we use the concept presented in this section for concurrent visualization of multiple objects derived from the same data set, this restriction is not necessary - objects could also be derived from multiple data sets. The approach could be straight-forwardly used for general multi-volume visualization. However, we note that the use of intersection transfer functions might not be feasible in a setup consisting of a large number of objects. Increasing the number of objects will quickly lead to a combinatorial explosion in the number of possible regions of intersection. In such a case the objects for which such a fine-grained control is required should be limited by application-specific constraints.

5 Illustrative Enhancement

Illustration is closely related to non-photorealistic rendering methods, many of which attempt to mimic artistic styles and techniques. In this section we present a simple approach which integrates several presented models and is thus well-suited for a volume illustration system. Most illumination models use information about the angle between normal, light vector and viewing vector 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. Numerous distinct approaches have been presented that use these quantities in different combinations to achieve a wide variety of effects. Our goal is to present a computationally inexpensive method which integrates many of these models.

We define a two-dimensional lighting transfer function. The arguments of this function are the dot product between the normalized gradient and the normalized light vector and the dot product between normalized gradient

and the normalized half-way vector. A two-dimensional lookup table stores the ambient, diffuse, and specular lighting contributions for every $\hat{N} \cdot \hat{L}$ and $\hat{N} \cdot \hat{H}$ pair. Additionally, a fourth component used for opacity enhancement is stored.

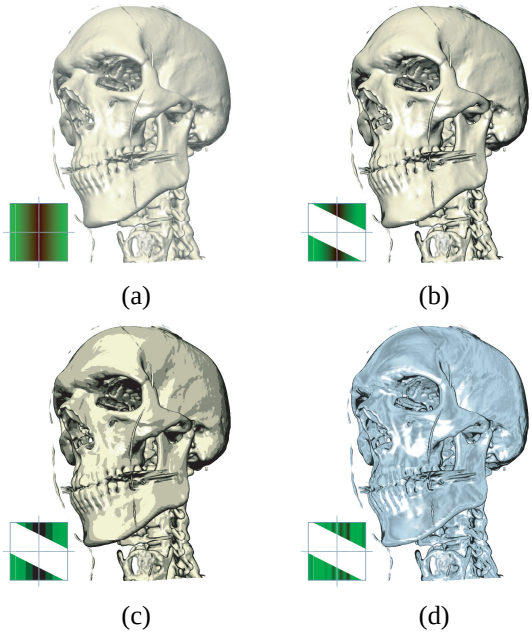
Shading is then performed by using these four values in the following way to compute the shaded color c_s and shaded opacity α_s:

$$\begin{aligned} c_s &= (s_a(\hat{N} \cdot \hat{L}, \hat{N} \cdot \hat{H}) + s_d(\hat{N} \cdot \hat{L}, \hat{N} \cdot \hat{H})) \cdot c_u + s_s(\hat{N} \cdot \hat{L}, \hat{N} \cdot \hat{H}) \\ \alpha_s &= (\min(1, s_\alpha(\hat{N} \cdot \hat{L}, \hat{N} \cdot \hat{H}) + (1 - |N|)))^{-1} \cdot \alpha_u \end{aligned} \tag{8}$$

where c_u and α_u are the unshaded color and opacity, $\hat{N}$ is the normalized gradient, $\hat{L}$ is the normalized light vector, $\hat{H}$ is the normalized half-way vector, and s_a, s_d, and s_s are the shading transfer function components for ambient, diffuse, and specular lighting contributions. The opacity enhancement component of the transfer function denoted by s_α is combined with the gradient magnitude $|N|$ to modulate the unshaded opacity value (we assume that the gradients have been scaled such that $|N|$ is between zero and one).

We use the terms ”ambient”, ”diffuse”, and ”specular” to illustrate the simple correspondence in case of Phong-Blinn lighting. However, the semantics of these components are defined by the model used for generation of the lighting transfer function. Thus, a lighting transfer function might use these terms to achieve effects completely unrelated to ambient, diffuse, and specular lighting contributions. In a similar matter, when examining Equation 8 it can be seen that the ambient and diffuse components could be combined without loss. We only choose to keep them separate for the sake of consistency and simplicity.

It is straight-forward to use this kind of lighting transfer function for common Phong-Blinn lighting. However, many other models can also be specified in this way and evaluated at constant costs. For example, contour lines are commonly realized by using a dark color where the dot product between gradient and view vector $\hat{N} \cdot \hat{V}$ approaches zero, i.e., these two vectors are nearly orthogonal. If we have $\hat{N} \cdot \hat{L}$ and $\hat{N} \cdot \hat{H}$ with $\hat{H} = \widehat{L+V}$, then $\hat{N} \cdot \hat{V} = \hat{N} \cdot \hat{L} + 2(\hat{N} \cdot \hat{H})$. We can thus create a lighting transfer function where we set ambient, diffuse and specular components to zero where $\hat{N} \cdot \hat{L} \approx -2(\hat{N} \cdot \hat{H})$. 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 transfer function (e.g., using a gaussian) with no additional costs during rendering. Using the opacity enhancement component of the lighting transfer function also allows for a meaningful combination of contour enhancement and transparency: the opacity of contour regions is increased, but only where the gradient magnitude is high. Without taking the gradient magnitude into account opacity enhanced contour lines would lead to a cluttered image in translucent views. This is due to rapidly varying gradient directions in nearly homogeneous regions. Pure gradient-magnitude opacity-enhancement without directional dependence just requires a constant s_α. Other methods, such as cartoon shading [3] or metal shading [6] can be realized straight-forwardly and combined with effects like contour enhancement. Figure 5 shows an image rendered using four different lighting transfer functions (standard Phong-Blinn lighting, Phong-Blinn lighting with contour enhancement, cartoon shading with contour enhancement, and metal shading with contour enhancement). Figure 6 illustrates how separate lighting transfer functions for background, ghost, and selection can be used to put emphasis on one of these objects.



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

Figure 5: The same data set rendered with four different lighting transfer functions (the lighting transfer function for each image are displayed in the lower left corner - ambient, diffuse, specular, and opacity enhancement are encoded in the red, green, blue, and alpha channel, respectively). (a) Standard Phong-Blinn lighting. (b) Phong-Blinn lighting with contour enhancement. (c) Cartoon shading with contour enhancement. (d) Metal shading with contour enhancement.

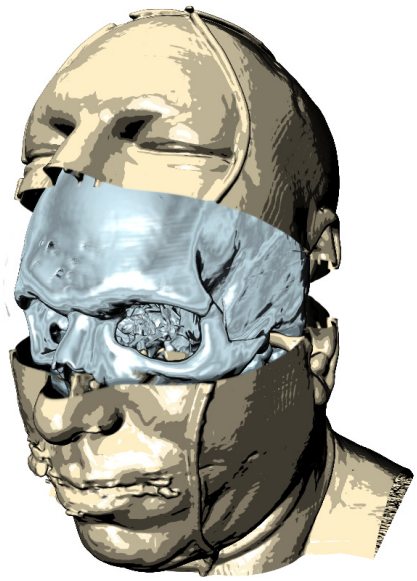


Figure 6: Using different lighting transfer functions for background, ghost, and selection. The background (lower part of the head) and the selection (upper part of the head) use cartoon shading, while the ghost (threshold is set to display the skull) is illuminated using metal shading. The selection has been moved to achieve an effect similar to volume splitting [9].

6 Selective Illustration

In this section we present techniques for selective illustration. Selective illustration techniques are methods which aim to emphasize specific user-defined features in a data set using visual conventions commonly employed by human illustrators. They are closely related to focus+context approaches frequently found in information visualization. The general idea is to highlight the region of interest (focus) without completely removing other information important for orientation (context).

6.1 Volume Painting

Volume Segmentation, i.e., the identification of individual objects in volumetric data sets is an area of extensive research, especially in medical imaging applications. Approaches range from very general methods to algorithms specifically designed to identify certain structures. An important criterion is the exactness of the segmentation, i.e., the ratio between correctly and incorrectly classified voxels. In practise, due to limited information, this criterion is difficult to measure. For volume illustration, however, voxel-exact classification of features is not necessarily of primary concern. Rather, it is important that the illustrator can quickly and easily add and remove structures of interest to and from the selection. Furthermore, as our approach is based on a fuzzy selection function, this fuzzyness should be also supported by the selection definition method. For this reason, we use a simple three-dimensional volumetric painting approach for selection definition. 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). We have found that this approach is intuitive and capable of achieving good results in a short time: the user specifies a transfer function which displays the object of interest and then just paints on it until it is fully selected. However, it is clear that a real-world application should also include more sophisticated algorithms. Just like image editing software normally supports manual and semi-automatic selection mechanisms (e.g., the common ”magic wand tool”), a volume illustration system should include volume painting as well as region growing or watershed segmentation.

6.2 Cutaways and Ghosting

Cutaways (also referred to as cut-away views) are an important tool commonly employed by illustrators to display specific features occluded by other objects. The occluding object is cut out to reveal the structure of interest. Viola et. al. [22] introduced importance-driven volume rendering, a general framework how to determine which object is to be cut by using an importance function. Our simplified three-object setup allows to define this importance statically, which enables us to skip costly importance compositing and thus allows for an efficient implementation. Cutaways are only performed on the background and can be independently defined for ghost and selection.

Ghosting refers to a technique which is frequently used in conjunction with cutaways. Instead of removing the occluding regions completely, opacity is selectively reduced in a way which attempts to preserve features such as edges. This tends to aid mental reconstruction of these structures and generally gives a better impression of the spacial location of the object in focus. In

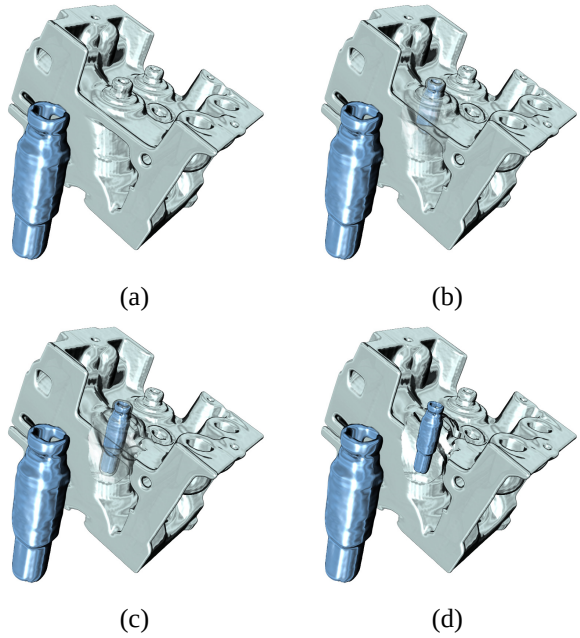


Figure 7: Different degrees of ghosting - from no ghosting (a) to full cutaway (d).

our approach, the user can smoothly control the degree of ghosting from no ghosting (opacity is not reduced at all) to full cutaway view (occluding structures are completely suppressed) as shown in Figure 7. This is achieved by combining a user-defined ghosting factor with the opacity-enhancement component of the lighting transfer function. Thus, for a lighting transfer function which enhances the opacity of contours, increasing the degree of ghosting will preserve these regions. Again, in the context of importance-driven volume rendering this approach can be seen as a special level-of-sparseness which is designed to closely correspond to traditional illustration techniques.

6.3 Visual Conventions and Interaction

As the selection can undergo a user-defined transformation there are a number of possibilities how to combine the effects of transfer functions, cutaways and ghosting, and spacial displacement. In its simplest form, this can be used to illustrate the removal or insertion of an object. Furthermore, ”magic views” on a structure of interest can be generated, where the object is displayed using a different degree of detail, orientation, or rendering style.

Illustrators commonly employ certain visual conventions to indicate the role of an object in their works. In our illustration environment, we provide the user with

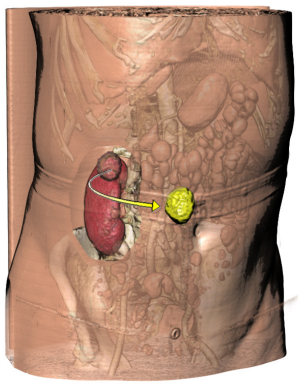


Figure 8: Illustrating a tumor resection procedure using an automatically generated arrow.

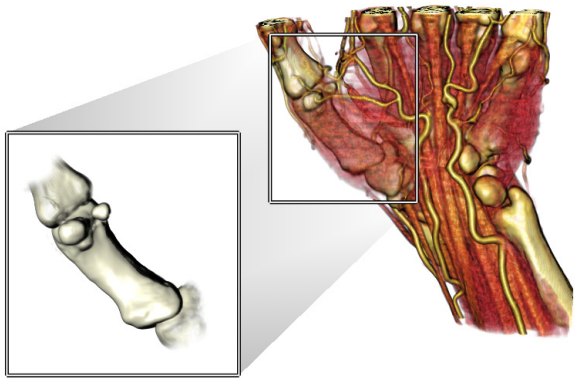


Figure 9: Detailed depiction of a hand bone using a fan.

different kinds of visual enhancements inspired by these conventions:

Boxes: For three-dimensional interaction, bounding boxes provide useful cues on the position and orientation of an object if occlusions are handled correctly. The display of bounding boxes is most useful when the selection is arranged during the design of an illustration. For the presentation of the illustration, however, the bounding boxes can be distracting and potentially occlude important details.

Arrows: Arrows 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). Analogously, we use arrows to depict the translation between ghost and selection, i.e., the arrow is automatically drawn from the object’s original position to its current location. To avoid very short arrows in case the se-

lection and the ghost project to nearby positions in image space, we use the screen-space depth difference to control the curvature of the arrow. This leads to the kind of bent arrows frequently found in illustrations. Figure 8 shows an example for the use of arrows.

Fans: A fan is a connected pair of shapes, such as rectangles or circles, used to indicate a more detailed or alternative depiction of a structure. It can be easily constructed by connecting the screen-space bounding rectangles of ghost and selection. In combination with cutaways and ghosting, this type of enhancement can lead to very expressive visualizations, depicting, for example, two different representations of the same object (see Figure 9).

Apart from controlling visual appearance, it is useful to provide different interaction types based on the role of an object in the illustration. A selection can be in one of three states which influence the way it behaves in relation to the remaining scene:

Integrated: The selection acts as a full part of the three-dimensional scene. This is intuitive, but has certain drawbacks. For example, when the viewpoint is rotated, the selection's movement is dependent on its distance to the origin. It can easily move out of the viewport or can be occluded by other objects.

Decoupled: The opposite to the integrated approach is to fully decouple the selection from the scene. It can be independently manipulated and is not affected by the viewing transformation. This is, for instance, useful when it is required to depict an object at a specific orientation regardless of the viewing transformation.

Pinned: A useful hybrid between the two modes above is to allow the object to be pinned to its current position in image space. Its on-screen location remains static, but it is still affected by rotations. A rotation of the viewpoint causes the same relative rotation of the object. For example, this can be used to generate a special view which always shows the part of an object facing away from the viewer in the background object.

6.4 Annotations

Hand-made illustrations in scientific and technical textbooks commonly use labels or legends to establish a coreferential relation between pictorial elements and textual expressions. As we allow multiple selections to be defined, annotations are important for both, recreating the appearance of static illustrations, and simplifying orientation in our interactive environment. For placing annotations we need their screen-space bounding rectangles and anchor points. We use the following guidelines to derive a simple layout algorithm for optically pleasing annotation placement (for a more complete description of annotation layout styles and guidelines refer to [1]):

- Annotations must not overlap.
- Connecting lines between annotation and anchor point must not cross.
- Annotations should not occlude any other structures.
- An annotation should be placed as close as possible to its anchor point.

In many textbook illustrations, annotations 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. Thus, the position of an annotation is defined by one value in the range $[0, 1]$. Based on its location in parametric space, a label is always placed in such a way that it remains outside the convex hull. All annotations are initially placed at the position along the 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 labels overlaps its predecessor, it is moved 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 are handled by disabling annotations based on priority. We use the screen-space depth of the anchor point to define these priorities, i.e., annotations 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 (usually no more than 30 annotations

Figure 10: Annotated illustration of a human foot - the current selection is highlighted.

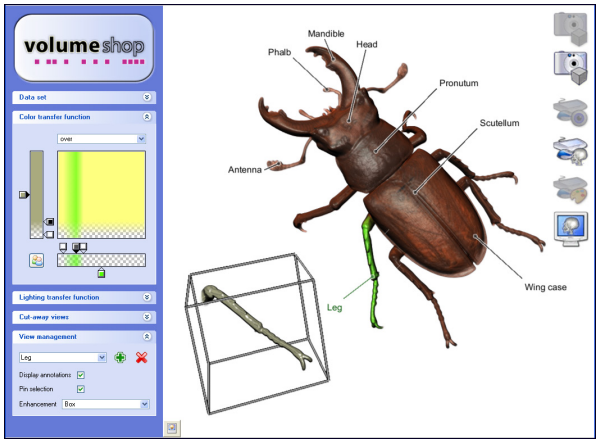


Figure 11: Screenshot of VolumeShop during operation.

are used in a single illustration) and generally leads to a visually pleasing layout. Due to the initialization of annotation locations at the position on the silhouette closest to the anchor point, the annotations generally move smoothly in animated views. Discontinuities only occur when intersections and overlaps need to be resolved. As some annotated structures might not be visible from every viewpoint, we use the screen-space depth of the anchor point to control the opacity of the connection line between anchor point and label. Figure 10 shows an annotated illustration of a human foot highlighting the current selection.

7 Implementation

In this section, we will briefly describe the implementation of our algorithm for illustrative multi-object volume rendering with support for cutaways and ghosting. It was integrated into VolumeShop, a prototype application for interactive direct volume illustration (see Figure 11). VolumeShop has been implemented in C++ and Cg using OpenGL. While we have clear indications that the current version of NVidia's Cg compiler does not produce optimal code in all circumstances, we have refrained from hand-optimizing assembly language shaders for the sake of portability.

It is possible to implement all presented methods in one single rendering pass. However, this would introduce considerable computational overhead, as, for example, multi-object compositing would have to be performed for every sample point even if it only intersects one object. While current graphics hardware supports dynamic branching, it still introduces severe performance penalties. It is therefore favorable to choose a multi-pass approach. A well-established strategy is to use the early-z culling capability of modern hardware for computational masking. Employing this approach we can identify those regions where less work has to be performed and use simplified vertex and fragment programs in these areas.

We can quickly extract bounding volumes for background, ghost, and selection by traversing our hierarchical data structures and rendering the corresponding geometry. Initially, we set up two depth maps by rendering the bounding volumes of ghost and selection each into a separate depth texture with the depth test set to LESS. These depth maps are used in the subsequent rendering passes to discard fragments, thus emulating a two-sided depth test. For smooth cutaways we additionally filter these depth maps using a large kernel.

In principle, our implementation comprises three volume rendering passes using three sets of vertex and fragment programs with increasing complexity:

Background pass: The first volume rendering pass is responsible for the background object. We set the depth test to LESS and render the bounds of the background object into the depth buffer. Depth buffer writes are then disabled to take advantage of early-z culling and the depth test is set to GREATER. Thus, empty space up to the fist intersection point of a viewing ray with the background bounding volume is skipped without executing the fragment program. We then render view-aligned slices in back-to-front order and perform shading

in a fragment program. Shadow mapping hardware is used to discard fragments whose depth is greater or equal than the corresponding value of the ghost or selection depth texture. Thus, regions which might contain the ghost and/or the selection are effectively cut out from the background object.

Ghost pass: In the second volume rendering pass we start by clearing the depth buffer and rendering the bounding volume of the ghost object with the depth test set to GREATER. Then depth buffer writes are disabled again and the depth test is set to LESS. The fragment program needs to perform shading for background and ghost. Fragments whose depth value is greater or equal than the corresponding value of the selection depth map are discarded. If cutaways are enabled then the opacity of the background is additionally modulated by a user-defined ghosting factor for fragments whose depth value is greater or equal than the corresponding value of the ghost depth map.

Selection Pass: For the final pass we render the selection bounds into the cleared depth buffer with the depth test set to GREATER. Depth buffer writes are then disabled again and the depth test is set to LESS. The selection transformation is handled by passing in two sets of texture coordinates: one unmodified set for background and ghost, and an additional set for the selection which is transformed accordingly. In the fragment program we need to perform shading for background, selection, and ghost. We also handle background/selection and ghost/selection intersections by using the colors and opacities defined in the intersection transfer functions. For cutaways, the background's opacity is additionally modulated for fragments whose depth value is greater or equal than the corresponding values in one or both of the depth maps.

For handling of intersections with opaque geometry an additional depth map is generated before the background pass. The color contributions of the geometry are blended into the frame buffer. The depth texture is used in all three rendering passes to discard fragments which are occluded by geometry. Visual enhancements are either displayed as real three-dimensional objects with correct intersection handling (e.g., bounding boxes) or as overlays (e.g., fans).

As larger selections will require more fragments to be processed in the more complex rendering passes, the performance of the presented algorithm mainly depends on the size of the selection. Thus, if no selection has been defined we achieve almost the same frame rates as conventional slice-based volume rendering due to the effectivity of early-z culling. Selections, by definition, typically will be rather small compared to the background. Additionally, if we can determine that the selection does not intersect background or ghost (e.g., by means of a simple bounding box test) we execute a simplified fragment program in the selection pass.

selection	frame rate
none	8.28
16^3	8.04
32^3	6.81
64^3	4.86

Table 1: Performance results for rendering 444 slices of the UNC CT head (256^3) using different selection sizes.

For obtaining performance results we used the following setup: The chosen data set was the standard UNC CT head (256^3) rendered using $\sqrt{3 \cdot 256^2} \approx 444$ slices - a realistic number for high-quality rendering. The selection was set to a cube sized 16^3, 32^3, and 64^3 voxels centered in the middle of the data set. The selection transformation was set to identity. The transfer functions for background, ghost, and selection were set to zero opacity for values up to 1228 and to an opacity of one for all values above. The frame rates given in Table 1 are average figures for three 360° rotations about the x-,y-, and z-axis for a 512^2 viewport. An Intel Pentium 4 3.4 GHz GPU and a NVidia GeForce 6800 GT GPU were used to obtain these measurements.

These results indicate that our approach is well-suited for high-quality rendering in interactive applications. In the future, we expect to further increase the rendering performance by integrating early ray termination as proposed by Krüger and Westermann [11].

8 Conclusion and Future Work

In this work, we introduced the general concept of a direct volume illustration environment. Based on this concept, VolumeShop, an interactive system for the generation of high-quality illustrations from volumetric data, has been developed. An intuitive three-object setup for the interaction with volumetric data was discussed. We contributed a general technique for multi-object volume rendering which allows for emphasis of intersection regions via two-dimensional transfer functions. Furthermore, we introduced a unified approach to efficiently integrate different non-photorealistic illumination models. Techniques for selective illustration

were presented which combine cutaways and ghosting effects with artistic visual conventions for expressive visualization of volume data. In addition, we proposed volume painting as an interactive selection method and presented an algorithm for automated annotation placement. A hardware-accelerated volume renderer was developed which combines the presented techniques for interactive volume illustration.

While we believe that the results achieved with our prototype system are promising, a lot of work remains to be done. In the future we aim to integrate further artistic styles and techniques for the creation of aesthetically pleasing illustrations [20]. We also want to investigate methods for automatically guiding viewpoint specification [21] and light placement [12]. Finally, improved interaction metaphors and techniques could significantly contribute to the usability of a volume illustration system.

Acknowledgements

The carp data set is courtesy of Michael Scheuring, University of Erlangen, Germany. The stag beetle data set has been provided by Georg Glaeser, Vienna University of Applied Arts, Austria and Johannes Kastner, Wels College of Engineering, Austria. The visible human data set is courtesy of the Visible Human Project, National Library of Medicine, USA. The engine block data set is courtesy of General Electric, USA.

References

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization 2003*, pages 287–292, 2003.

[12] 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.

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

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

[15] 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.

[16] E. B. Lum and K.-L. Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the International Symposium on Non-photorealistic Animation and Rendering 2002*, pages 67–74, 2002.

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

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

[19] 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.

[20] 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.

[21] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings of Vision Modeling and Visualization 2001*, pages 273–280, 2001.

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

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

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

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

[26] 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.