Enhanced Perception in Volume Visualization

Doctoral Thesis

Jose Díaz Iriberri

Barcelona, February 2013

Doctoral dissertation submitted for
the International Doctorate Mention

Universitat Politècnica de Catalunya

Department of Software (LSI)
PhD Programme in Computing

Advisor : Pere Pau Vázquez Alcocer
Co-Advisor: Isabel Navazo Álvaro

Abstract

Due to the nature of scientific data sets, the generation of convenient visualizations may be a difficult task, but crucial to correctly convey the relevant information of the data. When working with complex volume models, such as the anatomical ones, it is important to provide accurate representations, since a misinterpretation can lead to serious mistakes while diagnosing a disease or planning surgery. In these cases, enhancing the perception of the features of interest usually helps to properly understand the data.

Throughout years, researchers have focused on different methods to improve the visualization of volume data sets. For instance, the definition of good transfer functions is a key issue in *Volume Visualization*, since transfer functions determine how materials are classified. Other approaches are based on simulating realistic illumination models to enhance the spatial perception, or using illustrative effects to provide the level of abstraction needed to correctly interpret the data.

This thesis contributes with new approaches to enhance the visual and spatial perception in *Volume Visualization*. Thanks to the new computing capabilities of modern graphics hardware, the proposed algorithms are capable of modifying the illumination model and simulating illustrative motifs in real time.

In order to enhance local details, which are useful to better perceive the shape and the surfaces of the volume, our first contribution is an algorithm that employs a common sharpening operator to modify the lighting applied. As a result, the overall contrast of the visualization is enhanced by brightening the salient features and darkening the deeper regions of the volume model.

The enhancement of depth perception in *Direct Volume Rendering* is also covered in the thesis. To do this, we propose two algorithms to simulate ambient occlusion: a screen-space technique based on using depth information to estimate the amount of light occluded, and a view-independent method that uses the density values of the data set to estimate the occlusion. Additionally, depth perception is also enhanced by adding halos around the structures of interest.

Maximum Intensity Projection images provide a good understanding of the high intensity features of the data, but lack any contextual information. In order to enhance the depth perception in such a case, we present a novel technique based on changing how intensity is accumulated. Furthermore, the perception of the spatial arrangement of the displayed structures is also enhanced by adding certain colour cues.

The last contribution is a new manipulation tool designed for adding contextual information when cutting the volume. Based on traditional il-

lustrative effects, this method allows the user to directly extrude structures from the cross-section of the cut. As a result, the clipped structures are displayed at different heights, preserving the information needed to correctly perceive them.

Resumen

Debido a la naturaleza de los datos científicos, visualizarlos correctamente puede ser una tarea complicada, pero crucial para interpretarlos de forma adecada. Cuando se trabaja con modelos de volumen complejos, como es el caso de los modelos anatómicos, es importante generar imágenes precisas, ya que una mala interpretación de las mismas puede producir errores graves en el diagnóstico de enfermedades o en la planificación de operaciones quirúrgicas. En estos casos, mejorar la percepción de las zonas de interés, facilita la comprensión de la información inherente a los datos.

Durante décadas, los investigadores se han centrado en el desarrollo de técnicas para mejorar la visualización de datos volumétricos. Por ejemplo, los métodos que permiten definir buenas funciones de transferéncia son clave, ya que éstas determinan cómo se clasifican los materiales. Otros ejemplos son las técnicas que simulan modelos de iluminación realista, que permiten percibir mejor la distribución espacial de los elementos del volumen, o bien los que imitan efectos ilustrativos, que proporcionan el nivel de abstracción necesario para interpretar correctamente los datos.

El trabajo presentado en esta tesis se centra en mejorar la percepción de los elementos del volumen, ya sea modificando el modelo de iluminación aplicado en la visualización, o simulando efectos ilustrativos. Aprovechando la capacidad de cálculo de los nuevos procesadores gráficos, se describen un conjunto de algoritmos que permiten obtener los resultados en tiempo real.

Para mejorar la percepción de detalles locales, proponemos modificar el modelo de iluminación utilizando una conocida herramienta de procesado de imágenes (*unsharp masking*). Iluminando aquellos detalles que sobresalen de las superfícies y oscureciendo las zonas profundas, se mejora el contraste local de la imagen, con lo que se consigue realzar los detalles de superfície.

También se presentan diferentes técnicas para mejorar la percepción de la profundidad en *Direct Volume Rendering*. Concretamente, se propone modificar la iluminación teniendo en cuenta la oclusión ambiente de dos maneras diferentes: la primera utiliza los valores de profundidad en espacio imagen para calcular el factor de oclusión del entorno de cada píxel, mientras que la segunda utiliza los valores de densidad del volumen para aproximar dicha oclusión en cada vóxel. Además de estas dos técnicas, también se propone mejorar la percepción espacial y de la profundidad de ciertas estructuras mediante la generación de halos.

La técnica conocida como *Maximum Intensity Projection* (MIP) permite visualizar los elementos de mayor intensidad del volumen, pero no aporta ningún tipo de información contextual. Para mejorar la percepción de la profundidad, proponemos una nueva técnica basada en cambiar la forma en la que se acumula la intensidad en MIP. También se describe un esquema

de color para mejorar la percepción espacial de los elementos visualizados.

La última contribución de la tesis es una herramienta de manipulación directa de los datos, que permite preservar la información contextual cuando se realizan cortes en el modelo de volumen. Basada en técnicas ilustrativas tradicionales, esta técnica permite al usuario estirar las estructuras visibles en las secciones de los cortes. Como resultado, las estructuras de interés se visualizan a diferentes alturas sobre la sección, lo que permite al observador percibirlas correctamente.

Acknowledgements

The work presented in this thesis would not have been possible without the assistance of different people, particularly my advisors. Therefore, my most sincere thanks are due to Pere Pau Vázquez and Isabel Navazo, for their patient guidance and their valuable support during the last four and a half years. My special thanks are also extended to Eva Monclús, for her assistance with the volume rendering engine used to develop the algorithms presented in the thesis.

To my past and present colleagues of the MOVING Research Group, I would like to express my gratitude for creating such an amazing atmosphere to work and have fun in. As well, I want to thank the members of the Institute of Computer Graphics and Algorithms (Vienna University of Technology), with whom I spent four fantastic months.

For their ideas and the fruitful discussions we had, I wish to thank all the co-authors of the papers published throughout the thesis. The contribution of the volunteers who took part in different user studies has also been appreciated. Finally, I want to thank my father for providing the statistical knowledge whenever it was needed.

This thesis has been funded by a grant of the Spanish Ministry of Science and Innovation (FPI-BES-2008-002404).

Barcelona, February 2013

Jose Díaz

Contents

1 Introduction

Graphical representations have been used for centuries to provide knowledge of real world phenomena. From the days of Leonardo Da Vinci and his anatomical illustrations to the present computer generated visualizations, the process of creating such graphical depictions has radically changed, but its main goal has been preserved: providing means for a better understanding of real evidences. The way to obtain the data to be displayed has also evolved over the years. In the old days, the main source of information was the simple observation of visible occurrences. Nowadays, thanks to scientific and technological advances, natural phenomena can be simulated or acquired by sophisticated devices, which also provide information of invisible elements to the human eye, like microscopic particles or thermal information.

Due to the increasing complexity and the large amounts of data provided by current acquisition devices, new approaches are in constant development to effectively visualize the information from such data sets. In fact, thanks to the accuracy of the collected information, *Scientific Visualization* has evolved, not just to present graphical results, but also to validate hypotheses from the generated images. A clear example of the benefits provided by current approaches is *Medical Visualization*, which plays an important role in the diagnosis of diseases, treatment planning, and to provide intraoperative support in surgery.

Despite the diversity of information that can be acquired with scanners and other devices, there are a wide range of scientific and technological fields where the data has a volumetric nature. For instance, the information collected to study the air flow around objects in aerodynamics or the meteorological data to generate climate visualizations. Due to the high complexity that volumetric data may present, the generation of convenient visualizations can be a difficult task, but crucial to correctly understand the phenomenon being analyzed. This is especially true when working with anatomical data sets, where a misinterpretation of the data or a lack of accuracy can lead to serious mistakes while diagnosing or planning surgery. The subfield of *Computer Graphics* responsible for generating such good views is called *Volume Visualization*.

Among the topics of research in the visualization field, most of them focus on improving the perception of certain features or revealing occluded structures that are worth displaying. For instance, the overall appearance of the resulting visualization is mainly due to the transfer function applied, since it is responsible for assigning optical properties to data samples. For this reason, the research on the definition of good transfer functions is critical, because they allow us to distinguish between different materials, and determine how these materials are shown to the viewer.

Other topics of research are based on simulating a realistic shading model in order to clarify ambiguous visualizations. For example, the addition of lighting effects such as shadows or the amount of ambient light occluded by the elements being visualized, have proven their effectiveness in enhancing the depth perception and provide good clues for a better understanding of the spatial information.

A different way to enhance the perception of the features of interest is by simulating the effects applied by traditional artists in technical and anatomical illustrations. Thus, there are different illustrative motifs that can be simulated, such as cutaways and ghosted views to explore the internal parts of a volume, or the generation of halos around different structures in order to emphasize them. The branch of volume visualization in charge of simulating such effects is called *Illustrative Visualization*, and it is usually used when a certain level of abstraction is needed to understand complex volumetric data sets.

1.1 Motivation

Nowadays, thanks to the computing capabilities of modern equipment, better solutions and more complex approaches can be proposed to answer the open questions of a wide range of scientific and technological fields, not just by processing large amounts of measured data, but also by simulating the behaviour of the real phenomena to be analyzed. Among these fields, *Computer Graphics* has been drastically improved due to the design and development of *graphics processing units* (GPUs), which have relieved *central processing units* (CPUs) of an important part of the graphics related computations, speeding up the whole visualization process.

Taking the computing capabilities of modern GPUs into account, the main goal of this thesis is to propose, design and develop new approaches, in order to enhance the visualization of volumetric data sets by providing comprehensible visualizations of the data in real time. Concretely, throughout the development of the thesis, our research has been focused on improving the perception of different features in two ways: by simulating realistic shading effects and by adding traditional illustrative motifs.

1.2 Addressed problems and contributions

The work presented in this thesis focuses on addressing the following issues in the volume visualization field:

- **The enhancement of local features.** A well-known approach to visualize volmetric data is *Direct Volume Rendering* (DVR), which is based on the definition of transfer functions that assign optical properties to data values. Using these optical properties, mainly a color and an opacity for each value, the visualization algorithm is capable of generating images from volumetric data sets. Due to the fact that the definition process is focused on providing good views of certain structures of interest, small features may be difficult to distinguish. Based on *volumetric unsharp masking* [105] and a multi-scale volume hierarchy of the original data set, one of the contributions presented in this thesis is a new approach to enhance the perception of such small details in real time [26].

- **The enhancement of depth perception in *Direct Volume Rendering*.** Due to the complexity of volumetric information, understanding the spatial arrangement of some structures may present a certain difficulty. In order to enhance the perception of these structures, direct volume rendering images can be improved by adding additional depth cues. For instance, the simulation of shadows or other realistic shading effects and the application of illustrative motifs, have proven their effectiveness for enhancing the depth perception of volume models. This thesis presents two new data structures based on *summed area tables* [24] to improve the depth perception in two different ways: by modifying the shading estimating the ambient occlusion and by generating coloured halos around the structures of interest. Concretely, we propose the use of 2D *summed area tables* to simulate the aforementioned effects in image space [30], and also the computation of the ambient occlusion information in a volumetric basis by using 3D *summed area tables* [29].

- **The enhancement of depth perception and the spatial context in *Maximum Intensity Projection*(MIP).** *Maximum Intensity Projection* is another visualization approach but in this case, tailored to display the maximum intensity values of volumetric data sets. As a result, MIP images are similar to X-rays and they are often used by radiologists. Unfortunately, MIP images lack depth cues and spatial information, so radiologists normally have to contrast MIP results with DVR images, in order to identify the real location of the high density values. Another contribution of this thesis is a new approach

to enhance MIP visualizations with no need to compare with additional images, but just by adding some depth and spatial cues in a similar way to DVR [28].

- **The generation of context-preserving cross-sections of volume models.** Clipping the volume for inspecting the inner structures is a widely-used operation in volume visualization. However, this method may remove some contextual information that is worth preserving. One important issue of clipping techniques is that structures are removed by the clipping geometry and thus, the loss of context may make them difficult to identify. Inspired by traditional illustration effects, this thesis describes a new approach that visualizes the clipped structures at different heights from the cross-section of the clipped region [27]. Thus, the information needed to correctly perceive the shape, position and orientation of these structures is preserved.

1.3 About this document

The remainder of this document is organized as follows: the next chapter reviews some basic knowledge in the volume visualization field. Expert readers may freely skip this chapter since only introductory concepts are presented in it. Chapter 3 reviews the related work on the different topics of research covered in the thesis, mainly the issues stated in the previous section. The next four chapters describe the main features and contributions of the proposed approaches. Thus, the feature enhancement technique is described in Chapter 4, while the two different methods to enhance depth perception in DVR are presented in Chapter 5. After that, the technique to enhance the depth and spatial perception in MIP is described in Chapter 6. The last contribution of the thesis, the method to enhance the visualization of clipped structures, is explained in detail in Chapter 7. Finally, the conclusions of the thesis are presented in Chapter 8.

2 Preliminaries

Visualization in *Computer Graphics* is a broad field that is difficult to define. However, there are certain ideas that always arise when thinking about what *Visualization* is. For instance, the communication of visual information, the creation of visual representations to amplify cognition, etc. Depending on the data to be visualized, it is generally accepted that it can be divided into two main fields: *Information Visualization* and *Scientific Visualization*. The former is responsible for providing images from abstract data without a specific spatial representation, such as textual and economical information or the distribution of information on the Internet. The latter provides visual representations of scientific data with an n-dimensional nature, such as medical, geographical or meteorological information. The work presented in this thesis belongs to the scientific visualization field and the goal of this chapter is intended to provide a brief overview of the process to transform volumetric data into such graphical depictions.

2.1 The visualization pipeline

The visualization process to generate the final image (see Figure 2.1), usually includes the following stages, as stated by Haber and McNabb in [35]:

- **Acquisition:** input data can be simulated or measured with acquisition devices, such as Computer Tomography (CT) or Magnetic Resonance Imaging (MRI). The result is a set of discrete sample points distributed in three-dimensional space, where each sample can contain different information. For instance, CT provides a single density value of the measured material, whereas time-varying data stores more than one value for each sample position. The most used representation for such volumetric datasets is a regular grid in the three-dimensional space, where each discrete sample is represented by a single volume element called *voxel*.

- **Filtering:** due to the technical limitations of current acquisition devices and external conditions related to the acquisition process, raw

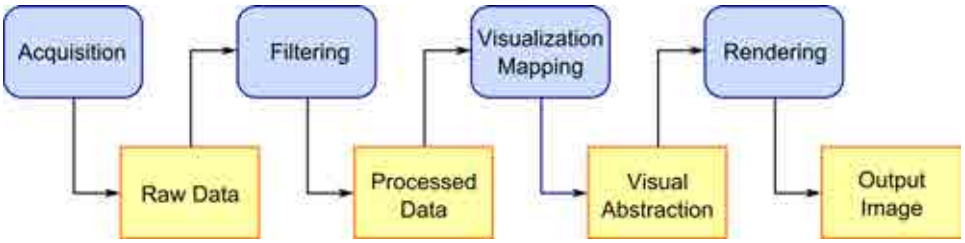


Figure 2.1: *The general visualization pipeline. After the acquisition or simulation of the input data, a filtering stage is often needed to reduce noise artifacts. Then, the resulting data is mapped to some entities which can be directly visualized by the rendering stage, producing the output image of the visualization process.*

data often need to be filtered in order to improve the quality of the resulting visualization. For example, smoothing filters can reduce noise artifacts produced by motion and breathing when scanning human beings.

- **Visualization mapping:** data processed in the previous stage cannot be directly visualized. Therefore, data values need to be mapped to certain entities and attributes which have a visual representation. They include geometric primitives, like points and lines, or optical properties like colors, transparency or surface textures. The result of this stage is a visual abstraction of the data which can be transformed into an image.

- **Rendering:** the last stage of the visualization pipeline is in charge of generating the final image from the visual abstraction of the data, given a certain viewpoint. To this end, typical rendering operations are performed, such as viewing transformations, occlusion removal, primitives rasterization, shading computation, etc.

The work presented in this thesis is mainly focused on the visualization mapping and rendering stages, since the different proposed approaches either assign optical properties to the data or modify the shading computations to better perceive certain features of the volume. Data used to test the proposed methods were provided from *Computer Tomography* (CT) and *Magnetic Resonance Images* (MRI) and, as a consequence, input data were stored as single density values along a set of regularly spaced samples within a volumetric region. These data sets will be referred in the rest of the thesis as volume models or volumetric data sets.

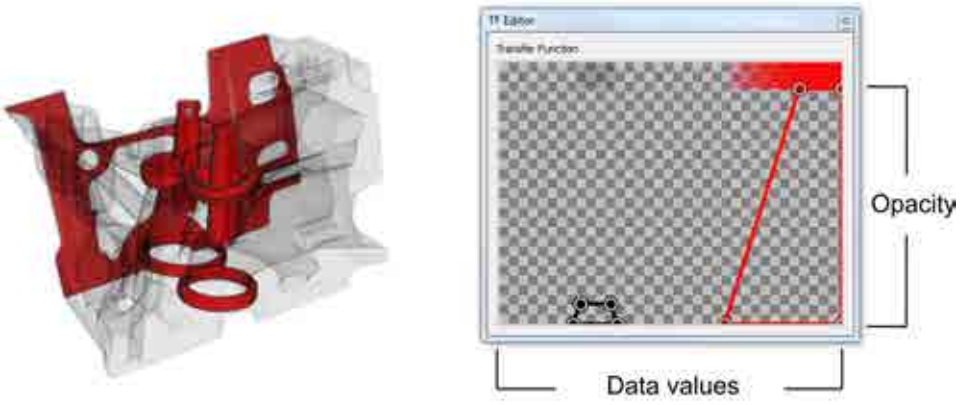


Figure 2.2: *Visualization of an engine model given a user-defined 1D transfer function. In this case, black color and low opacity has been assigned to density values representing the outer surface, while inner structures are opaque and red. Note that X and Y axes of the transfer function editor codify density values and opacities respectively.*

2.2 The transfer function

As we have seen in the previous section, the main goal of the visualization mapping is to transform the sampled data into different entities that could be visually represented. In order to correctly extract the information of the input data, the mapping process can not be performed arbitrarily, since it is required to assign similar graphical attributes to data values with similar properties. As a consequence, the visualization mapping involves a classification of the data.

In volume visualization, the mapping between data values and optical properties is provided by the so-called *transfer functions*. The most basic ones, also known as 1D transfer functions, are based on assigning colors and opacities to different ranges of the domain of the input data, in order to distinguish between the materials contained in the volumetric dataset (see Figure 2.2). In this way, transfer functions provide a classification of data values. More accurate classifications can be obtained by taking into account not just single data values when defining the transfer function, but also additional properties extracted from the input data such as the gradient magnitude [50].

The volumetric data set codifies a continuous space into discrete sample points within the 3D space, so the interpolation of new sample values is needed to reconstruct and visualize the volume. The mapping provided by the transfer function can be applied as a pre-classification or post-classification process. Pre-classification mapping assigns the optical properties defined by the transfer function to the discrete sample points, and then,

interpolates these properties to obtain the ones belonging to the points between samples. On the contrary, post-classification mapping applies the transfer function after the interpolation of data values, which leads to different results. The pre-classification process does not reproduce high frequencies, which generate noticeable artifacts in the final image. On the contrary, post-classification reproduces high frequencies obtaining better results. The main difference between both processes is when the mapping is performed and, as a consequence, in which stage of the visualization pipeline the transfer function is applied: whereas pre-classification belongs to the visualization mapping stage, post-classification is performed in the rendering stage.

2.3 Volume segmentation

Volume segmentation is the process of adding semantic information to the data set by labelling each voxel as belonging to a certain structure. When working with segmented volumes, transfer functions can be used to assign different optical properties to the segmented structures, in order to distinguish them in the resulting image.

Although a segmentation process may be a complex and time-consuming task, it usually provides more accurate results than the classification derived from transfer functions. For instance, in an anatomical data set of the hand, the phalanges present similar density values, and it would be difficult to separate them just by using transfer functions. This is due to the fact that segmenting a volume does not depend only on data values, but also on some other information related with the shape of the structures. For this reason and in order to obtain accurate results, the segmentation process often requires the participation of a specialist with a good knowledge of the elements contained in the volume.

There is a wide range of methods to segment volumetric data sets, such the image-based approaches described in [43] or the model-based algorithms presented in [39]. Although the segmentation topic is out of the scope of this thesis, Chapter 7 describes a new visualization technique that presents an on-the-fly segmentation process based on the *region-growing algorithm* [90]. This segmentation approach consists on placing different seed points within the volume and iteratively adding their neighbouring voxels until a certain finishing condition is fulfilled. This condition, which is normally a user-defined threshold for density values, determines which range of densities must be considered as belonging to the same structure as the seed points. In order to deal with the segmentation information, we use a second volume model with the same resolution as the original data set, which stores, for each voxel, the identifier of the structure it belongs to.

2.4 Rendering methods

The last stage of the visualization pipeline is responsible for generating the final image of the input data. In order to do this, the previous stage provides a visual abstraction that can be displayed on screen. The different methods employed for rendering volume models can be classified in the following categories: *Indirect Volume Rendering*, which often require a polygonal representation of the data, and *Direct Volume Rendering*, that generates the final image by estimating the light propagation within the volume.

2.4.1 Indirect volume rendering

The two most popular approaches that can be considered as indirect methods to visualize volumetric datasets are the *plane-oriented* and the *surface-oriented* techniques.

The plane-oriented visualization approach, widely used in the medical field and known as *cine mode*, is based on displaying single slices of the volume to examine them subsequently. These slices may be aligned with the three principal axes of the volume (X,Y,Z), providing saggital, coronal and axial views. Unfortunately, since the structures of interest might not be aligned with these axes, other orientations can be chosen to better perceive certain structures. In these cases, the resulting single slice comes from the intersection of the volume model with a plane, whose orientation is defined by its normal vector.

On the contrary, the main purpose of surface-oriented methods is the extraction and visualization of polygonal surfaces that represent the boundaries between a feature of interest and its surrounding data. These boundaries, typically determined by voxels with the same density values, define the so-called *isosurfaces*. Among the different methods to obtain a polygonal representation of a given isosurface [116], the most popular one is the *Marching Cubes* algorithm, proposed by Lorensen and Cline in 1987 [65]. This approach is based on triangulating each individual volume cell, which is a cubic region determined by eight adjacent data samples, guided by a set of fifteen possible triangulations. The main issue of Marching Cubes is the generation of surface artifacts due to ambiguous triangulations, which are mainly addressed with subsequent versions of this algorithm such as *Marching Tetrahedrons* [98]. By extracting isosurfaces in the visualization mapping stage (see Figure 2.1), the geometric primitives that represent the polygonal surfaces are rasterized in the rendering stage producing the final image.

2.4.2 Direct volume rendering

Direct Volume Rendering techniques do not require an intermediate representation as surface-oriented approaches do, because the image is directly generated from the data. This is due to the fact that direct volume rendering considers a volumetric dataset as a set of particles with certain optical properties representing a participating media. Hence, the output image is generated by estimating the light propagation within the volume. In order to approximate light transfer at a reasonable computational cost, the following optical models [71] are usually used:

- **Absorption Only**: this model considers that particles only absorb light.
- **Emission Only**: here, only the emission of light is considered.
- **Emission-Absorption**: in this case, particles emit and absorb light, but neither light scattering nor indirect illumination are considered.
- **Single Scattering and Shadowing**: this model considers single scattering of light that comes form an external light source and shadows.
- **Multiple Scattering**: this models evaluates the emission, absorption and scattering of light.

Among these optical models, the most used in direct volume rendering is the emission-absorption model, because it provides a good balance between generality and computational cost. In this case, the light energy I, also called the *radiance*, can be approximated by integrating along the light direction from a starting point ($s = s_0$) to an endpoint ($s = D$), using the so-called *volume-rendering integral*:

$$I(D) = I_0 * e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} q(s) * e^{-\int_{s_0}^{D} \kappa(t)dt} ds \tag{2.1}$$

where I_0 is the light that arrives to the initial position s_0 and $I(D)$ represents the radiance that leaves the volume at the endpoint D. The terms K and q are the absorption and emission coefficients from the optical properties assigned by the transfer function, and $e^{-\int_{s_0}^{D} \kappa(t)dt}$ represents the corresponding transparency of the material between s_0 and D. Note that by evaluating this equation, the absorption is estimated by the first addend while the second one approximates the emission.

Although scattering effects are not considered in the emission-absorption model, they can be approximated in a similar way to surface-oriented methods, that is, by applying local illumination models such as *Phong* [81] or the *Blinn-Phong* [8] models. In this case, the gradient of the scalar field which represents the data is used as the normal vector for the shading computation. By adding shading effects, a more realistic look is provided to the final visualization.

In order to numerically evaluate Equation 2.1, several approximations can be used. A popular one is based on dividing the integration domain into several discrete intervals, and estimating the radiance by means of the color and opacity provided by the transfer function:

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} (1 - \alpha_j) \tag{2.2}$$

being $c_0 = I(s_0)$. In this case c_i and $1 - \alpha_j$ represent the color and transparency of a certain discrete interval, respectively. This equation can be iteratively computed by splitting the summations and multiplications in simpler operations, just by compositing the colors and opacities of the samples along a viewing ray. The detailed derivation from Equation 2.1 to Equation 2.2 can be found in [34].

2.4.2.1 Compositing schemes

Depending on how optical properties are composed along viewing rays, there exists two different schemes: *front-to-back* and *back-to-front*. In the front-to-back compositing, viewing rays are traversed from the viewpoint to a given point. In this case, the accumulated color (C_{dst}) and opacity (α_{dst}) at a certain point of the ray are computed as follows:

$$C_{dst} = C_{dst} + (1 - \alpha_{dst}) * \alpha_{src} * C_{src} \tag{2.3}$$

$$\alpha_{dst} = \alpha_{dst} + (1 - \alpha_{dst}) * \alpha_{src} \tag{2.4}$$

where C_{src} and α_{src} are the color and opacity at the current point of the ray. Note that the color can be directly provided by the transfer function, as well as the local illumination model used to compute the shading.

Back-to-front compositing reverse the viewing ray direction, computing the accumulated color with the equation:

$$C_{dst} = (1 - \alpha_{src}) * C_{dst} + \alpha_{src} * C_{src} \tag{2.5}$$

In this case, there is no need to accumulate the opacity along the viewing ray, because it is not needed to obtain the color contribution.

Other volume visualization approaches like *Maximum Intensity Projection*, do not accumulate the color along viewing rays. In this case, the maximum intensity value along the ray is displayed producing similar images to X-rays, and the compositing is performed either back-to-front or front-to-back using the following formula:

$$C_{dst} = max(C_{dst}, C_{src}) \tag{2.6}$$

As a final remark in compositing, an opacity correction is required when varying the sampling rate throughout the volume. This variation can be needed, for instance, when we want to improve the quality of a certain region to better perceive the features contained in it. The work presented in this thesis assumes a regular sampling along viewing rays. Further information on opacity correction is provided in [34].

2.4.2.2 The volume rendering pipeline

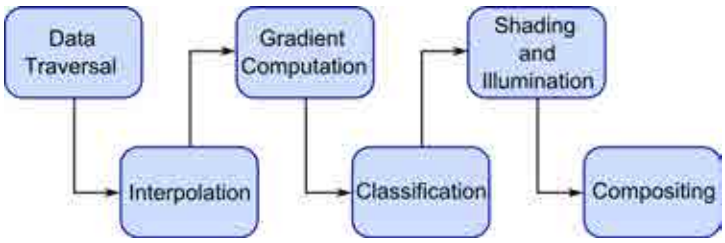


Figure 2.3: *The volume rendering pipeline. In order to evaluate the applied optical model, the volumetric data set must be traversed. By interpolating the density values of the discrete acquired data, the density of sample positions are obtained. After that, gradients are computed and optical properties are assigned to each sample during the classification step. Finally, shading and illumination of the samples are computed and composed along the viewing rays to obtain the final image.*

The evaluation of the optical model typically involves a set of stages, which define the *volume rendering pipeline* (see Figure 2.3):

- **Data Traversal**: in order to compute the accumulated color of each individual viewing ray, different samples of the volumetric data set must be evaluated.

- **Interpolation**: because the samples along the viewing rays might not correspond to discrete samples of the volume model, the density values of the ray samples may be obtained by interpolation from the closest discrete samples.

- **Gradient Computation**: discrete sample values of the original data set represent a scalar field from which a gradient can be computed. These gradients can be used in the volume rendering pipeline with different purposes, such as defining more accurate transfer functions or as a replacement of normal vector in the shading computation. Gradients are typically obtained with discrete filters like the central differences method.

- **Classification**: optical properties are assigned to data samples. This mapping is typically done by means of transfer functions that provide the colors and opacities to evaluate the volume-rendering integral.

- **Shading and Illumination**: as it has been mentioned before, scattering can be approximated by using local illumination models like the ones used in surface rendering approaches.

- **Compositing**: in order to generate the final image of the input data, one of the compositing schemes mentioned in the previous section is applied.

2.4.2.3 Volume rendering methods

Depending on how the volume is traversed, direct volume rendering techniques can be classified as *object-order* or *image-order* approaches. In object-order methods, the volume model is projected onto the image plane, and the color of each individual pixel is computed according to a certain compositing scheme, by blending the optical properties of the projected samples. Examples of object-order approaches are *splatting* [115], where the voxels are projected onto the screen, or *texture slicing*, where a set of 2D slices within the volumetric data set is used to sample the volume. Due to the fact that texture slicing is directly supported by the graphics hardware, this is one of the most used approaches in direct volume rendering.

On the contrary, image-order approaches take the image plane as a starting point of the viewing rays that traverse the volume. The most popular example of these approaches is the *ray casting* algorithm [58], which directly evaluates the volume-rendering integral by casting a ray towards the volume from each pixel of the image plane (see Figure 2.4). According to a determined compositing scheme, the final color of each pixel is computed by accumulating the colors and opacities of the samples along the ray. In order to speed up the ray casting, different acceleration methods can be used, such as the *early-ray termination* or the *empty-space skipping* described in [59]. The former one is based on finishing the ray traversal once the maximum opacity is reached in the front-to-back composition. The main reason to do this is that the color contribution is not modified by subsequent samples when full opacity is reached, so the sampling along the ray may be stopped.

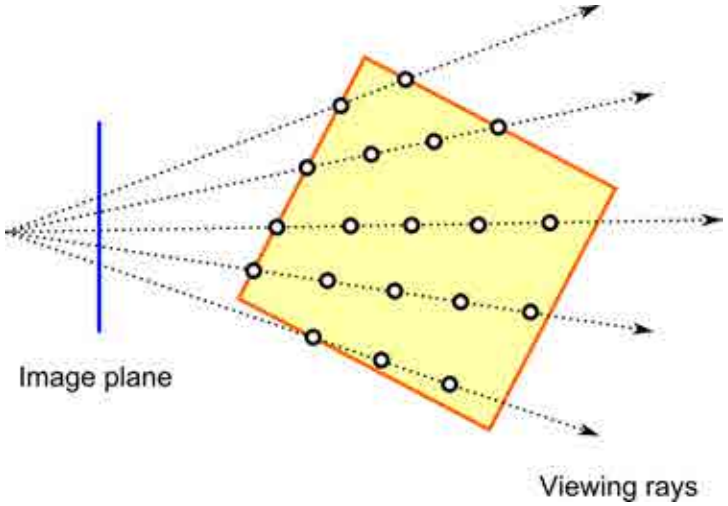


Figure 2.4: *General representation of the ray casting algorithm. A viewing ray, which is traced from each pixel of the image plane towards the volume, samples the data set at discrete positions. By compositing the optical properties of the samples along the rays, the color of each pixel is obtained.*

The latter is based on avoiding the sampling of empty regions that do not contribute to the color computation. Since ray casting can be efficiently implemented in the current GPUs, this method is used as a basis for a wide range of direct volume rendering applications.

2.5 Further reading

This chapter provides a general overview of *Volume Visualization*, intended to familiarize the reader with the different topics that appear in the rest of the thesis. The methods proposed in the next chapters are based on the ray casting algorithm using the front-to-back compositing scheme, and the used volumetric data sets come from CT and MRI images. Furthermore, the emission-absorption model is used to estimate the volume-rendering integral, approximating the scattering of light with the Blinn-Phong illumination model (normally referred as a Phong model in the rest of the thesis). Figure 2.5 shows the basic rendering pipeline to visualize a volume model using a GPU version of the ray casting process: once the original data set is read, a volume model is built from it, which is stored in a 3D texture and passed to the GPU. There the ray casting is performed to generate the final image. The approaches presented in this thesis propose different modifications to this basic pipeline in order to obtain the desired results.

For a detailed overview on *Visualization*, we would recommend this

books: *The Visualization Handbook*, edited by Charles D. Hansen and Christopher R. Johnson [36], *Visualization in Medicine* by Bernhard Preim and Dirk Bartz [82], and *Real-Time Volume Graphics* by Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christoph Rezk-Salama and Daniel Weiskopf [34], which also presents technical details for the implementation of different visualization methods.

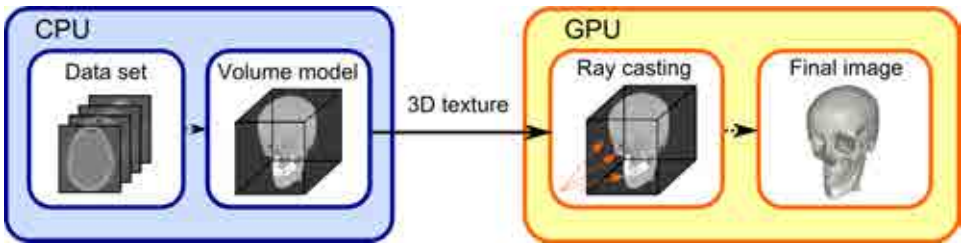


Figure 2.5: *Basic rendering pipeline to visualize a volume data set using the ray casting algorithm on the GPU. Firstly, the CPU reads the original data set and builds a volume model from it, which is stored in a 3D texture and passed to the CPU. There, the ray casting process is performed to obtain the final image.*

3
Previous work

Volume Visualization focuses on the generation of comprehensible images from volumetric datasets. In order to correctly extract and convey the information contained in the original data, a wide range of approaches have been presented in the last decades, each of them facing the problem in a different way. This chapter reviews the most significant work related to the problems addressed in this thesis.

3.1 Enhancement of local details

Local features normally provide a good comprehension of the shape and orientation of surfaces, as well as important clues for a correct interpretation of complex structures. Unfortunately, transfer functions and the illumination models used to visualize the data, usually lack enough accuracy to correctly convey them. In order to address this issue, one can attempt to modify the transfer function definition process by using additional information, instead of just using the original density values. This is the case of the *curvature-based transfer functions* by Kindlman *et al.* [49], which compute high quality curvature measurements based on the second-order derivatives of the volume data and convolution kernels. By using this information when designing the transfer function, feature enhancement is achieved by visualizing contours and emphasizing ridge and valley regions of the displayed isosurfaces. Furthermore, curvature-based transfer functions can also be used to distinguish between real surface features like high curvature edges, and noisy artifacts produced by acquisition devices.

Instead of modifying the transfer function definition process, some metrics can also be used in other stages of the visualization pipeline to determine which features must be enhanced. For instance, the visual saliency, which measures the perceptual importance of a certain feature with respect to its neighbouring elements, is the basis of the quality metric proposed by Jänicke and Chen [46], which may be used not just to guide the visualization process, but also to evaluate the generated views. Saliency is also used in the method proposed by Kim and Varshney [48], that emphasizes local details within selected regions of volume models. In order to do

this, a saliency field associated to the original volume is transformed into a multi-scale emphasis field, which is then used to modify the brightness and eventually the color of the visualized voxels. By using this measure and the center-surrounding operator proposed to obtain the emphasis field from the saliency one, brightness of local details is increased while lowered in their neighbouring regions. The saliency of each voxel can be defined manually, using eye-tracking devices or procedurally detecting local features, as it is done in [67], and assigning an importance value to each of them. Another approach to improve visibility and enhance local features with no need of modifying the transfer function is presented by Marchesin *et al.* in [69, 70]. This method is based on modifying the volume rendering integral in a per-pixel basis by using a relevance function that determines how the opacities contributing to each pixel must be weighted. The authors propose different relevant functions such as a binary classification to avoid the contribution of empty or less important voxels, or using the gradient and second derivatives to better perceive boundaries.

Shading modifications have also proved their effectiveness when enhancing local features. For instance, the *exaggerated shading* proposed by Rusinkiewicz *et al.* [93], which can be applied to polygonal surfaces or volumetric data sets, is based on modifying the local light direction. By computing the lighting at multiple scales (*i.e.* using smoothed versions of the normals or gradients), local features and the overall shape are enhanced at different frequencies regardless the orientation of the surfaces. Another example of such techniques is the *volumetric unsharp masking* proposed by Tao *et al.* [105], that uses a well-known signal-processing filter called unsharp masking to enhance features. By applying this operator to the radiance of the data samples, a lightening of local features is obtained making them more perceptible. However, the main limitation of this approach is the high memory consumption, since two auxiliary volumes are needed to apply unsharp masking: one containing the radiances of each voxel and another one to store a smoothed version of it. A more detailed review of this approach is presented in Chapter 4 of the thesis.

Another way to enhance the local features of surfaces is by using non-photorealistic effects, such as contours, silhouettes or hatching lines. The method presented by Csébfalvi *et al.* [25] belongs to this category and is based on displaying the contours of certain structures of the volume. In this case, contours are rendered depending on gradient magnitudes, instead of density values. Thus, there is no need of defining complex transfer functions, because optical properties are assigned to data samples depending on the gradient magnitude and the angle between the viewing direction and the gradient vector. In order to avoid occlusions, the maximum intensity projection (MIP) compositing scheme is used in this technique, and level lines may also be rendered to improve the spatial perception of certain isosurfaces.

Another non-photorealistic technique is the object space algorithm for

generating hatching lines by Nagy *et al.* [78]. This method presents a combination of line drawings and direct volume rendering (DVR) techniques, based on placing hatching lines along the principal curvature directions of certain structures, before combining them with the rendering of the volume model. The approach by Yuan and Chen [118] behaves in a similar way. In this case, the isosurfaces of interest, represented by feature lines, hatching strokes, ridges and valley lines, are also combined with the visualization algorithm. In this case, surface silhouettes are extracted in image space using image processing filters like edge detectors, which may be modified by introducing a procedural perturbation function to obtain different artistic results. In this way, traditional illustration effects enhance the main features of certain isosurfaces while direct volume rendering provides contextual information of the surrounding structures.

In order to enhance the perception of salient features, other approaches based on modifying the shading or simulating additional illustrative motifs have been proposed. For instance, some of them simulate shadows or the ambient occlusion, while others are based on the generation of halos around the features of interest. Because these approaches are also used to enhance the depth perception in DVR, a summary of the related work on this topics is presented in the next section.

In Chapter 4, we present a new feature enhancement approach [26] that focuses on improving the perception of local details by modifying the shading model, with no need of adding silhouettes or other illustrative effects. Concretely, we propose an extension of the method by Tao *et al.* [105], that reduces memory requirements and preserves the quality of the enhanced images with no need of huge pre-processing computations.

3.2 Depth perception in Direct Volume Rendering

The aforementioned techniques are tailored to enhance local details of surfaces, but in order to better perceive global features such as the spatial arrangement of the structures, other methods are required. A possible way to improve the perception of spatial information is by adding depth cues to the visualization, which has been traditionally done in direct volume rendering by simulating realistic illumination models or applying illustrative effects.

3.2.1 Simulation of realistic illumination models

The simulation of shadows is a widely-used approach for polygonal scenes. However, it may be a complex and computationally expensive task when the scene presents a volumetric nature. This is especially true when dealing with multiple light sources under varying conditions, such as modifying light positions or changing the transfer function. The most intuitive way to

simulate shadows and global illumination effects is by casting a ray from the point to shade to the light source, in order to find out the amount of light arriving at the point, and modulate the shading accordingly. However, the large number of rays required to shade a volume discards the application of this technique to interactively explore the data, as it happens in [117], where a recursive ray tracing algorithm is used to obtain high quality shadows and other effects.

In order to improve performance, researchers have addressed shadowing in different ways, by proposing a wide range of techniques that perform interactively. For example, the method by Behrens and Ratering [4] simulates shadows for texture-based volume rendering without expensive lighting computations. To do this, shadow maps are computed for each individual slice of the volume and rendered in the framebuffer with an specific blending function. Another example for slice-based volume rendering is the approach by Kniss *et al.* [51], that is based on two rendering passes for each slice, one from the light source and the other from the viewpoint, in order to compute the light that arrives to each voxel. By modulating the shading with this amount, hard shadows and translucent materials can be simulated maintaining the interactivity. Another two-pass rendering technique is presented by Zhang and Crawfis in [119], where splatting is used to generate hard and soft shadows. Other examples can be found in the work by Ropinski *et al.* [87] where shadow rays, shadow maps and deep shadow maps [64], which allow the simulation of shadows casted by semitransparent objects, are analyzed and efficiently implemented using the ray casting algorithm.

Instead of computing shadows as the previous approaches do, another way to enhance depth perception while increasing the realism of the shading model is by simulating the occlusion of the ambient light. Based on this principle, different techniques have been presented in the past, such as the *vicinity shading* by Stewart [100], which is a particular case of the *obscurances*, proposed previously by Zhukov [122]. In essence, Stewart describes a model that modulates the shading according to the presence of occluders in the neighborhood of a voxel. In a similar way, obscurances, also called *ambient occlusion* by Landis [57] and others, is a technique that modifies the shading in proportion to the total solid angle occluded by nearby elements (and sometimes not so close geometry). Unfortunately, the computation of the amount of occlusion for each voxel is obtained by discretely sampling different directions within the volume, what makes these techniques computationally expensive and not suited to explore the data interactively.

Since the occlusion computation is the most expensive step, more recent approaches for both polygonal [53, 120, 106, 68] and volume models [92, 88, 41], are based on computing the amount of occlusion in a preprocessing stage and use it to modify the shading in the rendering stage. In the approach by Ruiz *et al.* [92], occlusion information is computed by tracing several rays in a similar way to Stewart's vicinity shading [100]. As

a result, ambient occlusion and color bleeding can be applied interactively. Another example is the *local ambient occlusion* method by Hernell *et al.* [41], which restricts the occlusion calculations to a local spherical neighbourhood around each voxel. An interesting feature of this technique is that fully shadowed regions, which may occlude features of interest while exploring the data, are avoided. A different strategy is proposed by Ropinski *et al.* in [88], where a precomputed set of local histograms is used (as in [79]). Although they are used to apply ambient occlusion on the fly and to perform fast updates when changing the transfer function, the set of histograms may become too big for large data sets. Furthermore, color bleeding and volumetric glows are also simulated with the same data structure.

Other techniques like the slice-based directional occlusion shading by Schott *et al.* [97], simulate ambient occlusion by blurring the opacity of each voxel in an additional buffer, which is used to modulate the shading of the neighbouring voxels in the following slice. An extension to this apporach is described in [110], where the blurring of the opacity is achieved by using an elliptical cone instead of the radial blurring of Schott's approach. By determining the parameters of the cone, mainly the angle and the height (distance to the light), the user can obtain shadows and ambient occlusion from different light directions.

More recent approaches allow the simulation of global illumination effects like scattering. For instance, the shadow volume propagation by Ropinski *et al.* [85], approximates the propagation direction of light to generate shadows and simulate scattering within volumetric datasets. In this case, an auxiliary volume to store luminance and scattering information is computed, and used to modulate the Phong shading at each sample of the viewing rays. The same idea is used by the authors to simulate illumination effects of area light sources in [86]. Another example is the method by Schlegel *et al.* [96], which uses 3D summed area tables to efficiently compute the amount of occlusion due to the ambient or directional lights. As a result, soft shadows, scattering, color bleeding and ambient occlusion can be applied interactively. Shadowing and scattering are also considered in the *image plane sweep volume illumination* by Sundén *et al.* [101]. This method is based on a line-sweep computation of illumination information in image space, that is performed following the projection of the light direction on the screen. Once the lighting information for a line is computed, the shading of the samples is modified accordingly using the ray casting algorithm. The main difference of this technique with other ray casting-based approaches, is that viewing rays are cast from the image plane line by line, once the illumination for a line has been computed.

Other approaches rely on spherical harmonics to simulate global illumination. Some examples are the techniques by Beason *et al.* [3], used to visualize isosurfaces, the hierarchical visibility approximation implemented on the GPU by Ritschel [83], or the recent method by Kronander *et al.* [54],

which provides dynamic illumination effects with an arbitrary number of lights. More sophisticated methods use photon mapping to simulate global illumination in volume rendering, such as the one presented by Jönsson *et al.* in [47].

In order to evaluate the performance of different illumination models to perceive depth and size in volume visualization, the study by Lindemann and Ropinski [63] compares Phong shading to other six approaches [51, 97, 110, 85, 3, 88] that mainly simulate shadows or ambient occlusion. In three different tasks where relative and absolute depth, as well as relative size perception are tested, the results suggest that some of the six illumination models generally perform better than Phong shading to convey depth and size in the displayed images.

In conclusion, although simulating realistic illumination models enhances depth perception in volume visualization, the computation of occlusion information or, in other words, the amount of light that arrives to the point to shade, is a time-consuming task. In order to speed up the process, some of the reviewed approaches compute the required information in pre-processing stages to obtain interactive visualizations. Unfortunately, the memory needed to store the pre-computed information may be very high, constraining the applicability of these techniques to relatively small volume models. Some popular image-based methods applied in polygonal rendering, like the *horizon-based* and the *multi-layered ambient occlusion* by Bavoil *et al.* [2, 1] or the work by Mittring [76] in the *Cryengine 2*, could also be used in volume visualization. These techniques are based on sampling a certain number of pixels to compute the amount of occlusion, and blurring the result to obtain a continuous shading along the image. In this way, an approximation to the ambient occlusion can be computed for polygonal scenes in real time. However, due to the fact that volume rendering impact on performance is higher than polygonal rendering, the number of sampling pixels must be drastically reduced to maintain the frame rate, producing low-quality results. To overcome the existing limitations and to obtain a good balance among performance, pre-computation time and the quality of the results, two different approaches [29] to simulate ambient occlusion are presented in Chapter 5 of this thesis. One of them describes a screen-space approximation of vicinity shading and the other one estimates the occlusion with volumetric information.

3.2.2 Simulation of illustrative effects

As it has been mentioned in section 3.1, traditional illustration effects may also be used to enhance the depth perception. A well-known technique to obtain such a result is the generation of halos around the structures of interest which, by analyzing the region around the object where the halo is applied, provide a better perception of the spatial arrangements at first glance. In

order to simulate this illustrative effect in volume rendering, different techniques have been proposed. For example, the method by Interrante and Grosch [44] applies halos to enhance depth perception in flow visualization. In this approach, halos are generated by means of an auxiliary volume containing a slightly larger scale version of the original data, which determines the voxels that belong to the halo region. By darkening the shading of these voxels, halos are displayed around the original flow lines. Other approaches to generate halos can be found in [33] and [102]. In both papers, gradient-based methods are used to enhance the depth perception and local details. Feature enhancement is achieved by extracting boundaries, silhouettes or drawing sketch lines based on gradient directions and magnitudes. View-dependent halos created in orthogonal planes to the viewing direction are also computed. In this case, the gradient determines the thickness of the halo, while its brightness and opacity are controlled by a scalar value. Another technique to generate halos is the approach by Tarini *et al.* [106], that is applied to impostor-based molecular visualization instead of volume rendering. In this case, halos are drawn in a second rendering pass, when the depth buffer has been updated. This is due to the fact that the opacity of each point of the halo depends on the distance with the atom border and the depth of background elements.

The previous techniques are based on brightening or darkening the color of the image in the regions of influence of the halos, but no hue changes are considered, as it is done in the volumetric halos by Bruckner and Gröller [12]. The main idea of this approach is to process the volume model in view-aligned slices and perform three main steps for each slice. First of all, the structures to highlight are classified depending on their density value, direction and position, and several seeds are placed around them. After that, the halo region of influence is generated by filtering the seeds of the previous stage. Then colors and opacities are assigned to the halo regions and a final compositing step adds the obtained halos to the volume rendering of the volumetric data set.

Chapter 5 of this thesis proposes a new data structure based on summed area tables that is used to simulate ambient occlusion and halos. In our case, depth-based colored halos around the structures of interest are generated when visualizing the volume with the ray casting algorithm.

3.3 Depth perception in Maximum Intensity Projection

The use of the maximum operator as a compositing scheme makes the structures with high signal intensities visible [77], producing similar results to X-rays. As a consequence, Maximum Intensity Projection (MIP) images are widely used in medicine for vascular visualization. The main issue of MIP is that the maximum operator does not provide any contextual cue, so

physicians often have to change the point of view by rotating the volume or compare MIP with DVR renderings to perceive the spatial arrangement of the structures.

An effective way to convey the spatial context is by means of adding depth cues. As a consequence, several attempts have been carried out to enhance depth perception in MIP. For instance, Heidrich *et al.* [38] propose a method to improve the perception of depth using polygonal surfaces. The core idea is to extract the required isosurfaces from the volume and modify the vertex coordinates of every polygon before projecting them on the framebuffer. By replacing the z-coordinate of each vertex with their associated isovalues, the depth buffer is modified so that geometry with higher densities are closer to the viewpoint. Then, the z-buffer is used to enhance the MIP visualization. A different approach is the *Local Maximum Intensity Projection* proposed by Sato *et al.* [94], which overcomes the limitations of MIP with no need of extracting polygonal surfaces, by selecting local maximum values along the viewing rays. By displaying the first local maximum above a pre-defined threshold or the maximum intensity value along the ray if no local maximum is found, this technique effectively enhance the contours and provides the spatial relationships between vascular vessels from CT and MRI angiographies.

Ropinski *et al.* [89] enhance the depth perception in angiography images in a different way. This technique is based on using a set of methods such as edge enhancement to distinguish contours, changing the color or applying the depth of field effect. The color modification, called *pseudo chromadepth*, is based on the fact that the lens of the eye refracts colored light with different wavelengths at different angles and, as a result, blue objects are perceived as being far away, while red objects are perceived as being closer. By assigning reddish tones to closer objects and blueish tones to further ones, the depth perception is enhanced. On the contrary, the depth of field effect presents a blurred visualization of the further parts, which are determined by defining a distance threshold.

Maximum Intensity Difference Accumulation (MIDA) [14] is a visualization approach that can be seen as an intermediate step between both of MIP and DVR. In the case of MIP, depth and spatial cues are added by simulating a similar shading to DVR. In the case of DVR, high intensity values behind opaque structures are preserved as MIP does. In order to achieve this result, MIDA is based on modifying the compositing scheme of the volume rendering pipeline, by taking into account local intensity maxima along the viewing rays. Thus, instead of using the maximum operator, colors and opacities are accumulated along the rays as it is done in DVR, but modulating both properties when a local maximum is found. An extension of this technique is proposed in [112], where the modulation only affects the opacity contribution, which is driven by different importance measurements derived from gradient magnitudes, depth, intensity information or a combination of

these values.

The loss of information is the main issue that has to be taken into account when enhancing MIP visualizations, since adding color cues or modifying the shading, may conceal some information that is worth preserving. In order to enhance depth perception in MIP and reveal the arrangement of the rendered structures with the minimum loss of information, we describe in Chapter 6 a new method called Depth-enhanced Maximum Intensity Projection (DeMIP) [28], which is based on slightly modifying the MIP shading taking into account the depth of closer structures with a similar material to the one with maximum intensity. Furthermore, some color schemes are also provided to correctly convey the spatial context.

Recently, Zhou *et al.* have presented the *Shape-enhanced Maximum Intensity Projection* approach [121], which is inspired by DeMIP. The main difference with our technique is that the Phong shading model is applied using the gradient of the closest samples along viewing rays, instead of modifying MIP according to their depth. After that, a tone reduction operator [32] is performed to preserve the MIP local contrast in the shaded image, and depth-based color cues are also used to better perceive depth information.

For a deeper work on depth perception in *Computer Graphics*, the interested reader can refer to the early experiments by Wanger *et al.* [113], or the more recent work by Pfautz [80].

3.4 Visualization of inner structures

In scientific visualization, features of interest are usually situated in the interior of the volume. For example, if we consider anatomical datasets, we would like to obtain good views of the elements placed inside the body, not just the skin and other features that are visible to the naked eye. In order to correctly visualize the inner structures of volumetric data, different methods have been presented in the past, which can be mainly classified in the following categories: *cutaway* views, *focus+context* visualization, *importance-driven* volume rendering and *deformation-based* approaches.

Widely used in technical and anatomical illustrations, **cutaways** are based on removing certain regions of the outer structures in order to reveal the interior. This effect can be achieved by simply defining a clipping plane or more complex cutting geometries as it is done by Weiskopf *et al.* in [114]. This work describes different depth-based clipping approaches based on analyzing the depth information of the cutting geometry to decide which part of the volume has to be clipped. Furthermore, a volumetric clipping method is also presented. In this case, the clipping object is voxelized and used as a 3D mask in order to determine which voxels of the original volume must be removed. Another example of cutaway views can be found in the anatomical atlases by Höhne *et al.* [42], where the inner parts of anatomical

datasets are visualized by using cutting planes. An interesting feature of this approach is the *selective cutting* tool, which allows the user to cut the volume layer by layer, excluding the internal parts from being cut.

The virtual resection technique by Konrad-Verse *et al.* [52] also performs cuts of anatomical models. This method generates a deformable clipping plane from user-defined resection lines on the surface of the organ to be cut. Then, the deformable plane can be manipulated using the mouse in order to perform the desired resection. A completely different way to explore the inner parts using clipping planes is the *HingeSlicer* of McInerney and Broughton [73]. This method is based on creating custom cross-sectional views of the volume by using a 3D slice plane widget. This cutting structure is basically formed by a clipping plane that can be divided into portions, which may be displaced in order to obtain complex clipping geometries.

Although they do not belong to the volume visualization field, and therefore, they are out of the scope of this thesis, some interactive approaches for generating cutaways of polygonal models have also been proposed, such as the ones presented in [15, 31, 74]. Furthermore, the method by Li *et al.* [62], allows the user to modify the appearance of the elements in the clipped region by means of a rigging system that defines how the cutaway affects each structure.

Most clipping techniques and cutaways do not preserve contextual information, which is important to provide a better understanding of the inner structures. In order to overcome this limitation, **focus+context methods** visualize certain features of the context without occluding the internal elements. To obtain such a result, the opacity of the structures placed between the viewpoint and the features of interest may be modified, like in Bruckner *et al.* [9]. Considering that highly illuminated regions of the volume correspond to flat surfaces facing light sources, and less illuminated areas contain features that are worth preserving, this technique modifies the compositing scheme to increase the transparency of flat regions while preserving local details. To this end, the opacity of each sample along viewing rays is modulated taking into account the shading intensity, the gradient magnitude, the distance to the viewer and the accumulated opacity of the ray. Similar results are obtained with the *ClearView* technique [55], which is based on modifying the opacity of different segmented layers within a user-defined focus region. In order to do this, the different layers of the dataset are rendered in a first step, whereas the transparency of each layer is modulated afterwards, taking some importance measures into account. The importance of each layer can be determined by curvature information and the distance between context and focus layers. A different way to face the problem is presented by Chan *et al.* in [16]. In this method, different quality measures are applied to enhance the perception of semi-transparent layers in direct volume rendered images. To this end, a set of measures based on the visibility, shape and transparency of the structures is proposed and used afterwards

to automatically define the parameters involved in the rendering stage. As a result, internal elements are effectively revealed with a semi-transparent look.

Focus and context structures are sometimes determined by importance measures. The core idea of **importance-driven volume visualization** is to transfer importance to visibility, so a visibility priority is assigned to each object. In this way, when objects with different priorities occlude some others, the object with maximum priority is rendered. An example of importance-driven visualization is the work by Viola *et al.* [108], that proposes a set of rendering styles to visualize internal structures according to their priority. An importance value is assigned to each object and, instead of using constant optical properties, levels of sparseness are used to render the objects. These levels are based on color and opacity modulations, the screen-door transparency effect (*i.e.* a wire mesh with holes) or volume thinning, which are used to visualize the structures traversed by the viewing rays according to a determined importance-compositing scheme. Composition may be done in a similar way to MIP, where the maximum priority object along the ray is densely rendered while less important objects are rendered with a high level of sparseness, or according to the importance of a certain object, with respect to the sum of importances of the others. A final compositing scheme is also proposed where the visibility of the focus object is constantly preserved.

Note that the previous approaches require tools which allow the user to easily determine the features of interest, importance information, and eventually, the focus of attention. In order to obtain good visualizations, the definition of the region of interest is crucial and often a complex stage. Moreover, most of the previous methods work with pre-segmented datasets and users select by hand the important structures. In order to do this, different tools may be used, like the volume painting metaphor described in [10]. In this method, the user clicks on a pixel and selects the closer structure projected on this pixel, by using a 3D volumetric brush. A different tool to define the region of interest is presented by Chen *et al.* in [17], which allows the user to generate complex cutaways by directly carving, peeling and cutting the volume. The core idea of these manipulation tools is placing a set of points on the surface and determine if their associated voxels must be removed. By considering each point as an energy field that smoothly decreases to zero, they obtain soft boundaries on the clipped regions. This paper also presents an interactive segmentation process based on the region growing algorithm [90], which uses sketch lines to place different seeds within the structure to segment.

Other approaches based on **volume deformation** also permit the interactive manipulation of volume elements. For example, the methods by Mensmann *et al.* [75] or Correa *et al.* [22], which generate sophisticated cutaways performed by surgical tools. Another example of volume deforma-

tion is the peel-away method proposed by Birkeland *et al.* [6]. In this case, a deformation template containing the mapping between the original and the new positions of the voxels after the deformation is computed on-the-fly. Furthermore, by rotating the template, internal structures can be visualized without occlusions after peeling away the outer surface of the volume. McGuffin *et al.* [72] also use deformations to visualize inner structures. This approach describes a system that provides several manipulation tools to split and open a volume, apply peeling operations or displace away the occluding structures. The main difference of this method with others is that voxels are rendered as points at their associated 3D positions in order to preserve the simplicity and flexibility of the system when deforming the volume.

Volume splitting and volume deformation were also used in [45] and [23] respectively. While the former one reviews different splitting operations for volume models and their applications, the second one is based on deforming volumetric datasets applying algebraic operations on displacement maps. The main contribution of this second technique is that complex deformations can be simulated with no need of using computationally expensive physically-based methods, but also by combining simple primitive displacements.

Exploded views is another technique used by traditional illustrators. In this case, occluding structures are decomposed and moved aside to show the internal parts. While different approaches have been presented to generate exploded views of polygonal models [61, 107], Bruckner and Gröller [11] describe an interactive and editable method for volumetric datasets. In this technique, the focus object is selected using the volume painting metaphor presented in [10], and the rest of the volume is divided in several parts that are displaced according to a force-based model determined by the focus structure. Furthermore, users can define how the volume is split and introduce different constrains in order to control the spatial arrangement of the different parts after the explosion effect.

More examples of illustrative effects employed to visualize the interior of the volume are provided by Svakhine *et al.* in [103]. This work describes a framework to create anatomical illustrations by highlighting the focus structure while deemphasizing less important regions. To this end, the user selects segmented objects as a focus, defines a focal area, or a specific range of values from the transfer function, and applies different rendering methods to each part. For instance, removing the occluding layers or displaying only the silhouette of the context object.

Another technique inspired in traditional illustrations is the recent work by Birkeland *et al.* [5], which preserves the amount of information revealed in the neighbourhood of a clipping plane by using an elastic membrane to clip the volume. This membrane adapts itself to the anatomical structure by defining a potential field that guides the membrane clipping. However, this approach may require a time-consuming user intervention, since the

operation on the potential field sometimes requires the manual addition of anchor points.

Summarizing, different approaches have been presented to explore the internal parts of a volumetric data set. Some of them are more simple, but have more limitations, such as the clipping techniques. On the contrary, more sophisticated approaches improve the results of clipping methods by increasing the complexity of their algorithms, but affecting the performance or usability. In order to improve the results of clipping techniques while preserving contextual information and their ease of use, we propose in Chapter 7 a new manipulation tool to generate enhanced structure-aware cross-sections of the volume [27]. Our approach is based on mouse dragging, which avoids interaction with several planes or complex bounding geometries. Thanks to this feature, even inexperienced users are able to generate anatomical illustrations efficiently.

4

Enhancement of local features

Transfer functions play a major role in *Visualization*, since they are in charge of assigning visual properties to the measured values. During the transfer function definition process, which normally requires human intervention, users focus on obtaining good depictions of certain features of interest, which can be either isolated structures or even the whole volume. The main consequence of this process is that local features, which are useful to convey the shape and detail of surfaces, may be difficult to perceive. For instance, in the images of Figure 4.1, we can see how the edges and some local superficial details of the engine shown in (a), are better perceived when applying a feature enhancement algorithm (b).

During the last decades, researchers have presented different approaches to obtain such enhanced visualizations. For instance, feature enhancement can be achieved by defining transfer functions that take into account additional information such as the curvature [49]. Other approaches that effectively enhance local features are based on lighting modifications [93], traditional illustration effects [78], or the use of well-known 2D image processing tools designed to sharpen details and increase the contrast of images. This is the case of the method presented by Tao *et al.* [105], which apply *unsharp masking* to the visualization of volume models, producing high-quality renderings by increasing the luminance of local details. Unfortunately, the main drawback of this technique is the high memory consumption, which limits its applicability to relatively small data sets.

In order to overcome the limitations of [105], this chapter presents a new feature enhancement technique that improves the perception of local details not just by lightening salient features but also by increasing the darkening of shadowed regions to improve the local contrast. Furthermore, feature enhancement by means of a harmonic color modification is also proposed. Based on applying unsharp masking to generate the aforementioned effects, our technique reduces the memory requirements of [105] by using one or two levels of a multi-scale volume hierarchy of the original data, in order to obtain the parameters involved in the unsharp masking computation.

The remainder of the chapter is organized as follows: Section 4.1 reviews some preliminary concepts while the proposed method is introduced in Sec-

(a) Phong shading (b) Feature-enhanced image

Figure 4.1: *Comparison between the Phong shaded visualization (a) of a volume model and the feature-enhanced result (b) obtained using the technique proposed in this chapter. Note how local details depicting the shape of the surface are clearly enhanced by using volumetric unsharp masking.*

tion 4.2. After that, the results obtained with different volumetric data sets are presented in Section 4.3. In order to evaluate our enhancement method, we carried out an informal user study that is described in Section 4.4. Finally, discussion about certain features of the approach and the conclusions are presented in Sections 4.5 and 4.6 respectively.

4.1 Preliminaries

This section introduces some basic topics that appear throughout the chapter. Concretely, as the basis of our proposal, the definition of unsharp masking and the multi-scale volume hierarchy used to reduce memory requirements are presented.

4.1.1 Unsharp masking

The unsharp mask is a filter that amplifies high-frequency components of a signal S, by adding back the difference between S and a smoothed version S' of the original signal. It can be achieved by using in the following equation:

$$Unsharp(S) = S + \lambda(S - S') \tag{4.1}$$

where λ is a scaling value that determines the intensity of the effect. When using unsharp masking as an image processing tool, *laplacian* or *gaussian*

(a) Source image (b) Sharpened image

Figure 4.2: *Enhanced image using unsharp masking. Note how the edges and other details are sharpened in (b), improving the overall appearance of the statues and the background vegetation with respect to (a).*

filters are usually used to obtain the smoother version of the signal (S'). The result of unsharp masking an image is that edges are enhanced, improving the perception of local details and the overall appearance, as it is shown in Figure 4.2.

Besides using unsharp masking in image processing, this filter has also been applied to enhance the visualization of 3D geometry [19, 66, 84] and volumetric data sets [105]. In this latter case, the *radiance* of each sample along the viewing rays is considered as a signal S, and the process to apply unsharp masking involves the following steps: firstly, the radiance of each voxel of the model is computed and stored in memory. Then, a smoother version of this radiance volume is obtained by means of a gaussian filter. And finally, the radiance of the samples is unsharped in the visualization process, obtaining the enhanced result. The main limitation of this process is the high amount of memory consumption, since it requires the storage of two additional volumes with the same resolution as the original dataset: the computed radiance volume and its smoothed version.

4.1.2 Multi-scale volume hierarchy

Our proposal of *volumetric unsharp masking* consists on using a multi-scale volume hierarchy of the original data set to reduce memory requirements and increase the local contrast. By creating a multi-scale volume hierarchy, different versions of the volume can be used to compute the smoother radiances required by the unsharp masking process. As it is described in

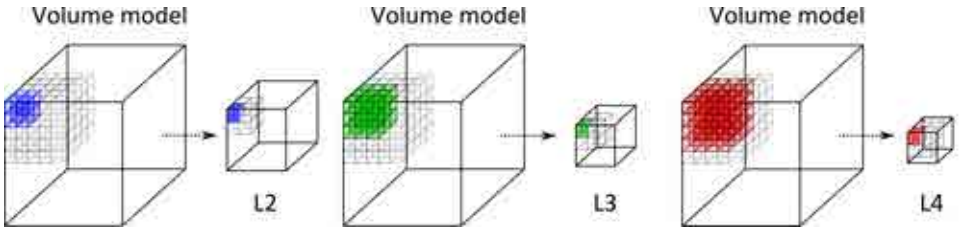


Figure 4.3: *Hierarchy building process. Each level of the hierarchy is computed by dividing the dimensions of the original volume by a given integer. The figure shows how many voxels are taken into account to compute the value of a single voxel in levels 2, 3 and 4.*

Section 4.2, just one or two levels of the hierarchy are used to emphasize local details so, at the most, two additional volumes of lower resolution than the original one must be stored in memory.

The use of a multi-scale volume hierarchy can be seen as a variation of traditional mipmapping. In *Computer Graphics*, a mipmap (sometimes spelled mip map) is a hierarchy of textures built from a source one, meant to increase texturing speed. This hierarchy is built iteratively by halving each dimension of the original texture, so each level reduces by 4 (2D mipmaps) or 8 (3D mipmaps) the size of the previous level. In our multi-scale hierarchy, the dimensions of the original data set can be divided by any integer ranging from 2 to the minimum dimension of the original volume. This is due to the fact that depending on the size of the features to enhance, levels not included in the classical mipmap may produce more accurate results, as it is analyzed in Section 4.2.1. The construction process of levels 2 to 4 is shown in Figure 4.3, where the dimensions of the original volume are divided by 2, 3 and 4 respectively. The fact that there is no need of storing the whole hierarchy in memory and that non-power-of-two levels can be computed, are the main differences between our multi-scale volume hierarchy and classical 3D mipmaps [60].

4.2 Volumetric unsharp masking

The purpose of our approach is the enhancement of local features to obtain visualizations like the one shown in Figure 4.1. To this end, we propose a new method based on the ray casting algorithm, where unsharp masking is applied during the shading computation of the samples along the viewing rays.

The general pipeline of our application is depicted in Figure 4.4, and can be briefly described as follows: firstly, the volume hierarchy from the original data set is computed on the CPU. For the volumes used to test our method, levels 2, 3, 4, 5, 6, 8, 16 and 32 were computed. Then, the volume

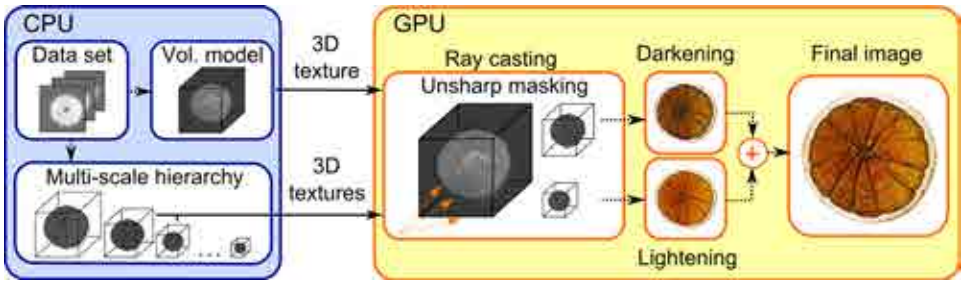


Figure 4.4: *Overview of our rendering algorithm. Once the dataset has been loaded, we build a 3D mipmap from it. Both the model and the mipmap are passed to the GPU, that uses them to generate the enhanced image. Both the lightening and darkening are achieved by unsharp masking the accumulated color of the ray casting process.*

model and one or two levels of the hierarchy are stored as 3D textures and loaded on the GPU. There, the volume is visualized by using the ray casting algorithm, but changing the shading computation of the samples by using unsharp masking. Finally, the enhanced visualization is obtained by compositing the modified shading of each sample along the viewing rays.

In order to increase the local contrast, the shading of the samples can be modified in two different ways: by producing the lightening of salient features or by darkening shadowed regions. Both effects are based on Equation 4.1 and use the same or two different levels of the multi-scale hierarchy to compute the smoothed version of the signal. The main difference between them is that despite the lightening applies the classical unsharp masking to the radiance of the samples, the darkening effect combines two different signals to obtain the desired result. In the next sections, both effects are described in detail.

4.2.1 Lightening salient features

In order to emphasize salient features, we apply a volumetric unsharp masking in a similar way to the method by Tao *et al.* [105], where the radiance of the samples is used as a signal. The main difference between both approaches is how the smoothed signal is obtained. Given a viewpoint, they generate a new volume containing the radiance of each voxel, that is approximated by the Phong shading color. Then, this volume is iteratively smoothed by using a gaussian filter, and stored as another 3D texture. As it has been stated in the previous section, three different volumes of the same size must be stored in memory: the original data set, the radiance volume and its smoothed version.

Instead of computing the radiances of the original volume and smoothing them, we propose to reverse the process by computing a smoother version

(a) Unsharp in CIELab (b) Unsharp in RGB

Figure 4.5: *Comparison between applying unsharp masking in CIELab (a) and RGB color spaces (b). Note that the color hue tends to green in some regions of the surface in (b), whereas it is preserved in (a).*

of the original data set and obtain the smoothed radiances from it. In other words, the average density of a cubic region centred in each sample is firstly computed, and the radiance of this average, also approximated by the Phong shading color, is used as a smoothed signal. In our case, the average density is obtained from a certain level of the multi-scale volume hierarchy by using trilinear interpolation. The main consequence of this modification with respect to [105] is that just the original data set and one level of the hierarchy (of lower resolution) are stored in memory to lighten features.

As it is stated in [84], local contrast enhancement must be performed in the CIELab color space, since it is a perceptually uniform space and, as humans are more sensitive to luminance variations, the luminance can be easily unsharped. On the contrary, the RGB color space is non-linear and directly unsharp the color may lead to hue changes as is shown in Figure 4.5. Therefore, in order to emphasize local details, the Phong shading color of each sample is computed first and transformed to CIELab yielding a triplet $[L, a, b]$. In the same way, the average density is obtained afterwards from a certain level of the hierarchy and its Phong color is transformed to CIELab $[L_{level_i}, a_{level_i}, b_{level_i}]$. Finally, the unsharped color of a sample in CIELab ($U_{CIELab}(C)$) is obtained by using the following equation:

$$U_{CIELab}(C) = [L + \gamma(L - L_{level_i}), k * a, k * b] \tag{4.2}$$

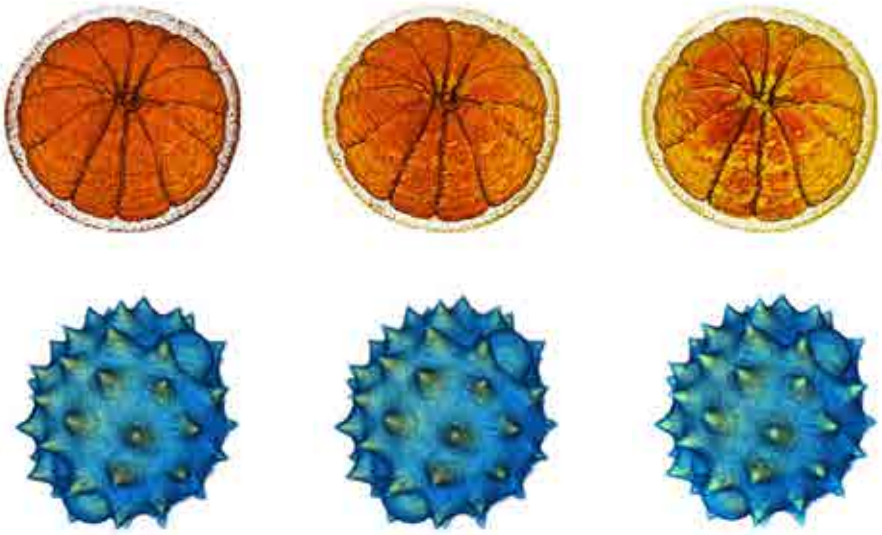
where γ is a user-defined scaling factor that controls the intensity of the effect, and the k factor is needed to guarantee that the saturation is preserved [105]:

$$k = (L + \gamma(L - L_{level_i}))/L \tag{4.3}$$

Once k is applied, Equation 4.2 can be written as:

$$U_{CIELab}(C) = \frac{L + \gamma(L - L_{level_i})}{L} * [L, a, b] \tag{4.4}$$

To finish the shading modification, the unsharped color of a sample is transformed back to RGB and the ray casting algorithm proceeds to the color composition. Note that the opacity channel of the color is not modified during the unsharp masking process, so the opacity of the original sample is used when compositing to preserve the shape of the enhanced features.



(a) Phong shading (b) Lightening level 2 (c) Lightening level 8

Figure 4.6: *Comparison between the Phong shaded visualization of two volume models (a) and the feature-enhanced result obtained using levels 2 (b) and 8 (c) of the multi-scale volume hierarchy. Note how small details on the surface (top (a)) are better perceived using a big resolution levels than a low one (top (b)). On the contrary, relatively big features such the spikes of the pollen grain are effectively enhanced using level 8 (bottom (c)), whereas the enhancement with a big resolution level (bottom (b)) is barely perceivable.*

The final appearance of the lightening effect depends on two parameters: γ and the level of the hierarchy used to obtain the smoother radiance, whose resolution determines which features are enhanced. This is due to the fact that the level used, which is chosen by the user among the levels of the hierarchy, depends on the relative size of the features with respect to the size of the volume. For instance, levels with low scaling factors are better to detect

small features because they provide a finer approximation of the original volume. As a consequence, just the neighbourhoods of the samples belonging to little details, such the small superficial folds of the orange segments shown in Figure 4.6 (top (a)), present average densities different enough to produce significant changes in the shading (top (b)). On the contrary, levels with big scaling factors contain broader approximations of the volume, so not just the samples of the small features, but also other surrounding samples might be enhanced (top (c)). For relatively big features, such the spikes of the pollen model (bottom (a)), levels with bigger scaling factors are needed to obtain significant shading modifications (bottom (c)).

Thanks to the fact that just the luminance is unsharped, the smoothed radiance can be approximated as we propose, by firstly computing an average density and obtaining the radiance from it. Note that the average density may be different from the density value of the current sample and, as a consequence, the Phong color from the average might be completely different from the sample's one, including color hue variations. This fact does not affect the result of the lightening, because just luminance changes are taken into account to emphasize features, independently on the color hues. Something to consider is that the opacity assigned by the transfer function to the average density may be equal to zero. In this case, the Phong color of the average must be replaced by a color with minor luminance than the one of the sample, in order to obtain the lightening (see Equation 4.1).

In the case of salient features, the difference between the density of the sample and the average of its neighbours is bigger, as it is expected to be the difference between the luminances of their computed radiances. For this reason, although unsharp masking modifies the shading of all the samples with a luminance change between their original and smoothed radiances, salient features are perceived more intensely than the other ones.

4.2.2 Darkening shadowed regions

In order to better perceive the features enhanced with the previous effect, local contrast may be increased by darkening shadowed areas, like valleys and deeper regions. To this end, we propose an algorithm equivalent to the one presented in the previous section, but combining two different signals in the process: the radiance of the sample (i.e. the Phong shading color) from the original volume $[L, a, b]$, and the color of the average density assigned by the transfer function $[L_{tf(level_i)}, a_{tf(level_i)}, b_{tf(level_i)}]$, which acts as a smoother version of the signal. Thus, the darkening color ($D_{CIELab}(C)$) is computed as:

$$D_{CIELab}(C) = \frac{L + \gamma(L - L_{tf(level_i)})}{L} * [L, a, b] \tag{4.5}$$

Note that the lightening and darkening are based on applying Equation

(a) Phong shading (b) Lightening (c) Darkening (d) Enhanced result

Figure 4.7: *Feature enhancement is achieved by lightening salient features (b) and darkening shadowed regions (c). The enhanced result is shown in image (d).*

4.1 to luminance values. In the case of salient features, the original signal L is generally a luminance value larger than L^{prime}, so the unsharped luminance $Unsharp(L)$ is larger than L producing the lightening effect. On the contrary, by using the luminance of a shaded color (the Phong color of the sample) as a signal L and the luminance of an unshaded color (the one assigned by the transfer function to the average density of the sample's neighbourhood) as a L^{prime}, the resulting unsharped luminance $Unsharp(L)$ is smaller than L in the shaded regions, producing the darkening effect. In the same way as the lightening, the opacity of the sample remains untouched in the process.

As a final remark, once the lightened and the darkened colors have been computed, both of them are transformed to RGB and combined as follows:

$$Color_{sample} = 0.5 * (RGB(U_{CIELab}(C)) + RGB(D_{CIELab}(C))) \quad (4.6)$$

An example of the combination of both effects is illustrated in Figure 4.7, where the enhanced visualization and the two effects are shown separately. For darkening, levels 2 or 3 of the hierarchy are used, since lower resolution levels produce extra, unpleasant shadowed regions.

4.2.3 Harmonized color change

In order to emphasize local features we have only modified the color by unsharping its luminance while maintaining the saturation. Nevertheless, traditional illustrators often use different colors to stress important details. For this reason, we also propose the use of *harmonic colors* [20] to emphasize local features, what ensures that the overall impression will not change in a sudden or unpleasant way. In order to this, after computing the Phong color of a given sample in the ray casting process, its harmonic is also determined, by selecting the complementary in the *hue color circle*. Then, this harmonic

color is transformed to CIELab and if the density value of the sample is significantly bigger than the average density of its neighbourhood, the original color is replaced with the harmonic one and emphasized using equation 4.2. In Figure 4.8 we can see how harmonic color feature emphasis behaves for the pollen grain and the orange models.

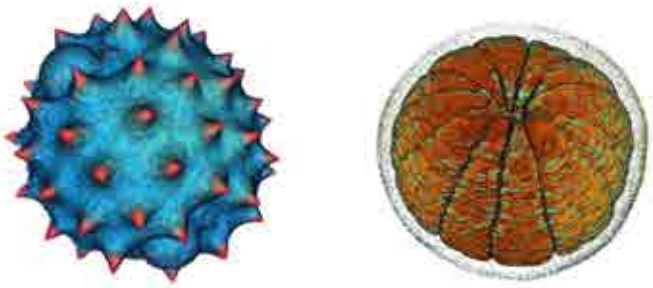


Figure 4.8: *Harmonic color-based feature emphasis for the pollen and the orange data sets.*

4.3 Results

The presented method has been tested with different volume models, whose resolution is up to 512 x 512 x 512 voxels. As it has been mentioned before, the multi-scale volume hierarchy contains levels 2, 3, 4, 5, 6, 8, 16 and 32, from which the user may choose one or two adequate levels to apply the lightening and darkening effects, depending on the size of the features to emphasize. Note that, in the worst case, levels 2 and 3 would be required, increasing memory consumption less than a sixth of the original volume, instead of triple the storage space as in [105].

In the majority of visualizations shown in the figures of this chapter, levels 2 or 3 of the multi-scale hierarchy are used for the darkening effect and levels 2 to 5 for the lightening. As it is shown in Figure 4.9, our approach is also well-suited for semi-transparent structures. In this case, the vessels and other fine details are emphasized in the presence of opaque and translucent materials.

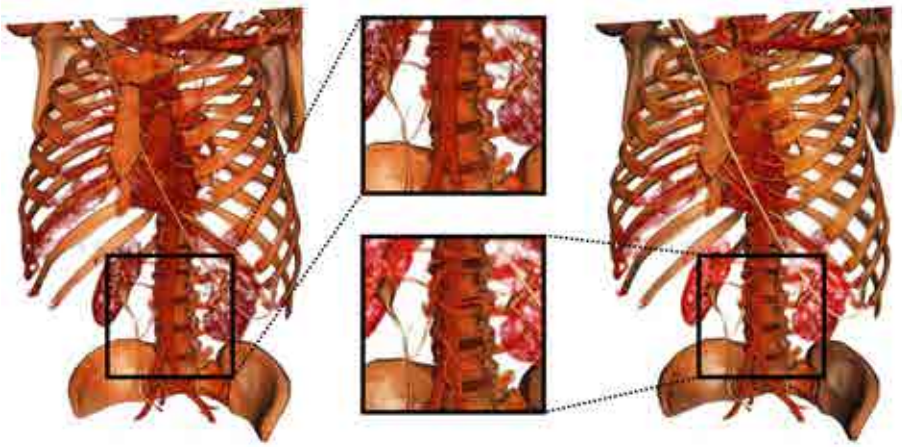
The time required for building the multi-scale hierarchy is shown in Table 4.1. As we can see, compared to the time required to load a volume model, which is usually in the range of several seconds for the larger ones, the time needed for the hierarchy generation is rather low. Concerning rendering performance, frame rates obtained for different data sets are shown in Table 4.2. When using the same level for both the lightening and the darkening, frame rates only decrease up to 20%. On the contrary, when

Volume data set	Volume resolution	Loading time (ms)	Hierarchy building time (ms)
Pollen	$192 \times 180 \times 168$	483	192
Orange	$256 \times 256 \times 64$	260	142
Engine	$256 \times 256 \times 256$	623	562
Abdomen	$512 \times 512 \times 171$	12376	1530
Feet	$512 \times 512 \times 282$	14246	2513
Body	$512 \times 512 \times 512$	20694	4559

Table 4.1: *Comparison between loading time and construction time of the multi-scale hierarchy (levels 2, 3, 4, 8, 16, and 32) expressed in milliseconds. Note how the hierarchy generation is roughly one order of magnitude faster than loading the original model.*

Volume data set	Phong shading	Enhanced 1 level	Impact 1 level	Enhanced 2 levels	Impact 2 levels
Pollen	59.50	59.50	0.08%	42.50	28.6%
Orange	59.47	48.23	18.9%	27.88	53.1%
Engine	58.98	52.43	11.1%	30.83	47.7%
Abdomen	28.38	25.14	11.4%	15.23	46.3%
Feet	22.52	18.01	20.0%	11.84	47.4%
Body	12.87	11.23	12.7%	8.00	37.8%

Table 4.2: *Frame rates obtained with different data sets (fps). When using the same level to apply the lightening and darkening, frame rates decay up to* 20% *compared to the Phong shading visualization, whereas the impact on rendering is up to* 50% *when using two different levels.*



(a) Phong shading (b) Enhanced image

Figure 4.9: *Enhanced image of a model with fine details and semi-transparent structures. Note how the vessels in (a) are more visible in the enhanced image (b). Levels 2 and 3 have been used for the darkening and lightening respectively.*

different levels are used, texture accesses have less coherency, and frame rates decrease up to 50%. However, this result improves the performance of the volumetric unsharp masking by Tao *et al.*, where frame rates decay up to 70% as it is shown in [105]. Timings have been obtained in an Intel Core2 DUO CPU running at 3.0 GHz equipped with a nVidia GeForce GTX 280 GPU with 1GB of RAM memory.

4.4 User study

In order to evaluate our technique, we carried out a user study, where participants had to solve two different tasks. The main goal of the first one was to obtain a subjective evaluation of the images generated with our technique, in comparison to the ones obtained with Phong shading and the method by Tao *et al.* [105]. To this end, we selected a set of six models, and rendered three images with the three approaches. An example of the renderings shown in this task can be seen in Figure 4.10. In order to avoid a learning effect, the three images were randomly ordered for each model and users had to answer which one was better to perceive the features of the model. A set of 42 people took part in this task and their choices show that users preferred the enhanced images produced by our method in the majority of cases (see Figure 4.11). In order to analyze the results, we computed the 95% Confidence Intervals (CI) for the sample proportion of answers Phong (10%),

(a) Phong shading (b) Tao's method (c) Our method

Figure 4.10: *Example of images shown in task 1 of the user study. Users had to choose which of the images was better to perceive the features of the model.*

Tao (18%), and our method (72%) of a survey where $n = 252$. The results obtained, using the usual estimation formulae, namely $\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$ are shown in Table 4.3.

95%	**Phong shading**	**Tao's method**	**Our method**
Lower bound	6.29	13.25	66.45
Upper bound	13.70	22.74	77.54

Table 4.3: 95% *confidence intervals for answers in task 1.*

The second task was designed to test our technique in a semi-transparent medium. The reason to do this was to check if features were emphasized enough to perceive them in a low visibility context. In order to do this, we built a set of synthetic data sets where we randomly placed a number of cylinders over a plane, all inside a semi-transparent cube (see Figure 4.12). Users had to count the number of cylinders, and we measured both the correctness and the time spent to complete the task. A set of 20 images, which were generated with Phong and our method, were displayed randomly, and a group of 16 people took part in the task. In the enhanced images, users correctly counted the number of cylinders, whereas with Phong, they wrongly answered in 9.3% of the cases.

As well as in the first task, the 95% CI for the mean of the times was computed for this second task, where $n = 160$. The results are shown in table 4.4.

In conclusion, the results of the study show that users generally preferred the enhanced visualizations obtained with our method than the ones

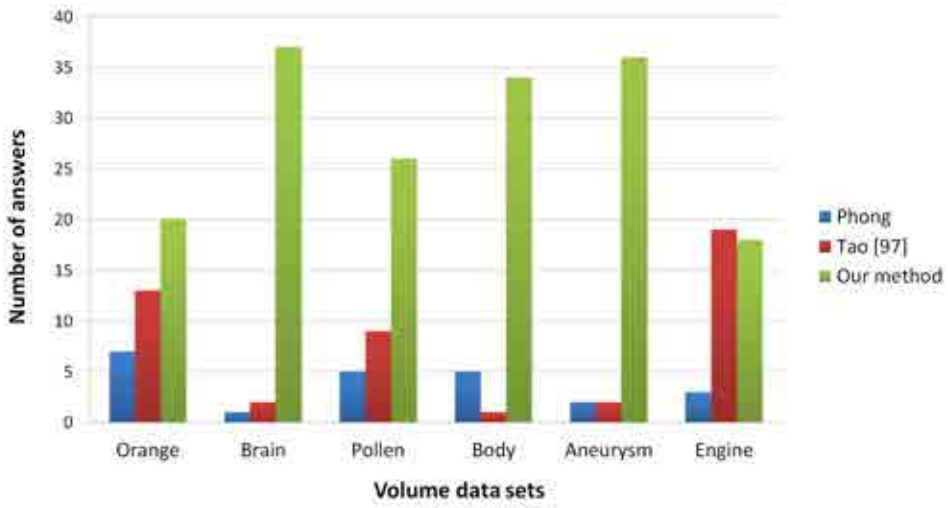


Figure 4.11: *Subjective evaluation of the visualization methods used in task 1. Users considered that our approach was better to perceive local features in the majority of the cases.*

95%	Phong shading	Our method
Lower bound	7.07	6.69
Upper bound	7.96	7.62

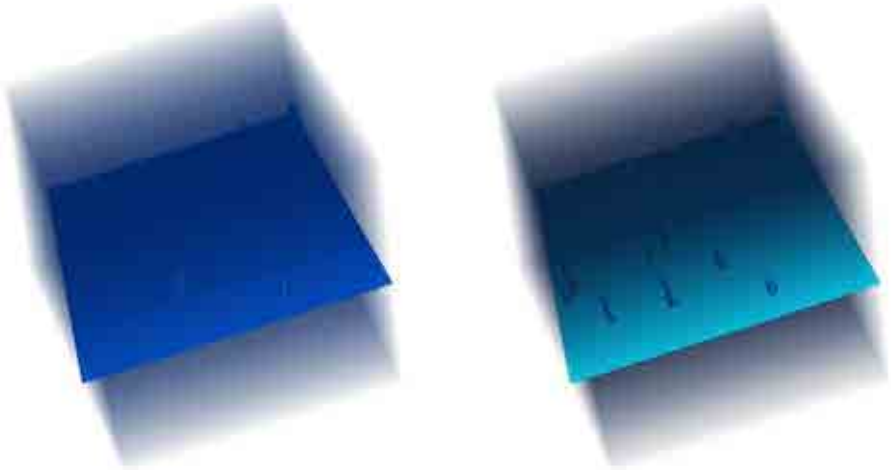
Table 4.4: 95% *confidence intervals for the mean of times (sec) of task 2.*

generated with the method by Tao or the Phong shading. Furthermore, the local contrast of the visualization around the cylinders in a semi-transparent medium was good enough to correctly perceive them and complete the task without errors.

4.5 Discussion

In this Section different approaches that we tested throughout the development of our method are discussed. Concretely, the sampling rate to apply unsharp masking without losing the quality of the results is analyzed.

Due to the fact that the features to enhance belong to certain isosurfaces of the data set, one may think that applying unsharp masking only to the samples of these isosurfaces might be enough to better perceive the local details. This strategy would improve the performance of the volumetric unsharp masking, but as it is shown in Figure 4.13, it does not produce



(a) Phong shading (b) Feature-enhanced image

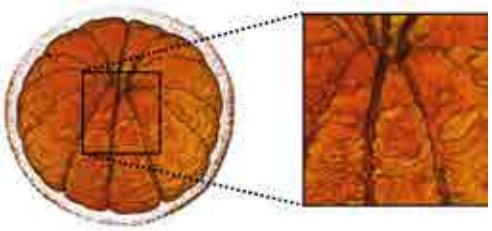
Figure 4.12: *Example of synthetic dataset used in task 2. The users had to count the number of cylinders in the volume.*

good results. In this figure, features are emphasized using the same level of the hierarchy and γ values (see Equations 4.4 and 4.5), which allow the user to modify the intensity of the enhancing effects. Note how local details are clearly perceived when unsharping the shading of all the samples along the viewing rays (Figure 4.13 center), but there is almost no change with respect to Phong shading when it is applied at isosurface level (Figure 4.13 bottom).

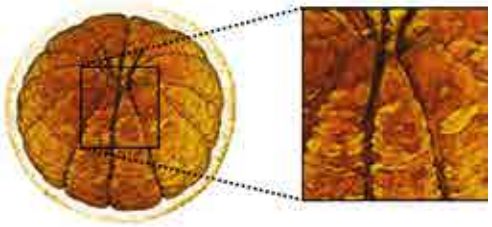
The main reason for obtaining such poor results when unsharp masking is applied at isosurface level, is that the contribution of a single sample to the final color of the ray may be quite low, especially when semi-transparent materials are traversed before reaching the isosurface to be enhanced. Just in the case of having totally opaque structures, the result of applying unsharp masking at a certain isosurface would be good enough to clearly perceive its local features. Nevertheless, the performance gain obtained by sampling a hierarchy level only at a given isosurface is roughly 2% to 10%, when the same level is used for the lightening and darkening effects. Therefore, we consider that is better to take all the samples of the ray into account and not just unsharp at isosurface level.

4.6 Conclusions

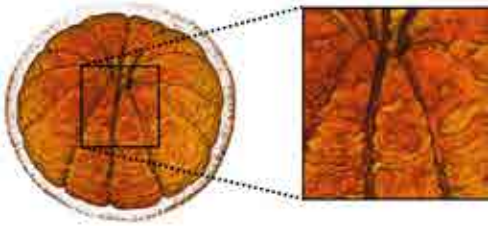
A new feature enhancement technique for volumetric data sets has been proposed in this chapter [26]. The main advantages of our method with respect



(a)Phong shading



(b) Sampling at each point



(c) Sampling at isosurface level

Figure 4.13: *Comparison of different sampling strategies. In our approach (b), the levels of the hierarchy are queried for each sample of the viewing rays. In (c), these levels are only queried when reaching a certain isosurface, in this case, the one that represents the surface of the orange. The Phong shading result (a) is also shown for comparison purposes.*

to previous approaches [105] are the improvement of the local contrast by means of lightening salient features and darkening shadowed regions, as well as the reduction of memory requirements. In order to lighten local details, unsharp masking is applied to the radiance of each sample along the viewing rays. On the contrary, the darkening of shadowed regions is achieved by combining in the process, the radiance and the optical properties assigned by the transfer function. The use of harmonic colors to further emphasize local details has been also proposed. Although it might be not adequate for some kinds of models, such as the medical ones, we believe it may result in nicer-looking illustrative renderings.

Memory requirements are reduced with respect to previous approaches because of the way unsharp masking is applied. The method by Tao *et al.*

[105] also performs unsharp masking in a volumetric basis, but the computation of the smoothed radiances involved in the process require storing in memory two additional volumes with the same resolution as the original one. Instead, our method just needs one or two low resolution versions of the original volume, which are chosen by the user from a multi-scale volume hierarchy, depending on the relative size of the features to enhance.

In order to evaluate the results of our technique we have conducted an informal user study. The results show that users consider that features are generally more perceptible with our method than with Tao's approach and the Phong shading visualization. Furthermore, users also performed better with the proposed method in a task designed to determine if it is suited to enhance features within a semi-transparent media.

5

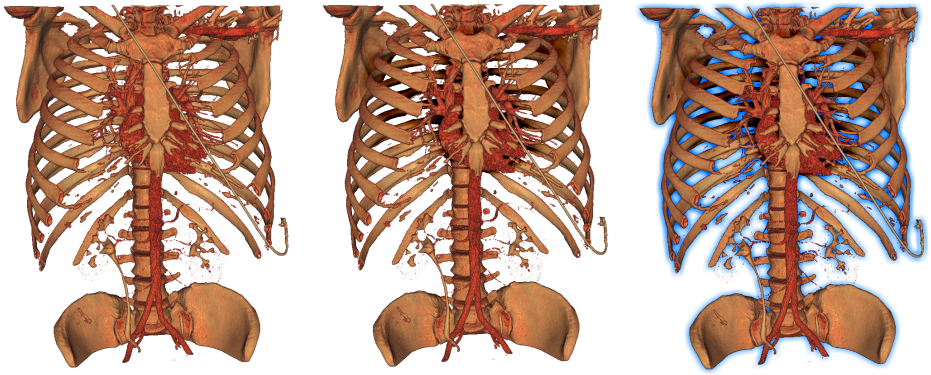
Depth-enhanced Direct Volume Rendering

In *Direct Volume Rendering* (DVR), volume data sets are normally illuminated by one or more light sources, and shading computations are based on the *Phong model* [81]. This is due to the fact that it provides good perceptual cues in the orientation of surfaces, what is generally enough to correctly perceive the spatial relationships among the main structures of the volume. However, due to the complexity of volumetric data, spatial information may be difficult to convey when there are many fine and overlapping structures.

In order to improve depth perception in DVR, research has been focused on modifying the shading model or simulating non-photorealistic techniques. Two examples of the former case are the addition of shadows or the computation of ambient occlusion (see Figure 5.1 (b)), which have proven their effectiveness to enhance the perception of the visualized structures [95]. On the contrary, non-photorealistic approaches reproduce illustrative effects used in anatomical and technical drawings, such enhanced silhouettes or colored halos (see Figure 5.1 (c)), which are used to highlight the features of interest while reducing possible perceptual ambiguities.

This chapter proposes two simple and fast approaches that are based on modifying the shading computations and generating illustrative effects. In both approaches, a novel algorithm and a data structure based on Summed Area Tables [24] are described. The first method, the *Vicinity Occlusion Maps* technique, uses screen-space information for simulating ambient occlusion and halos at a low cost. The second method, the *Density Summed Area Table* approach, considers volumetric information for simulating ambient occlusion with an almost negligible impact on rendering times, but presents more memory requirements than the screen-space method.

The remainder of the chapter is organized as follows: Section 5.1 describes the basic topics related to the proposed techniques. The Vicinity Occlusion Maps method is presented in detail in Section 5.2, including its results and some implementation details. Section 5.3 explains how to simulate ambient occlusion with a Density Summed Area Table, the obtained results with this approach and some implementation details. In order to



(a) Phong shading (b) Ambient occlusion (c) Halos and ambient occlusion

Figure 5.1: *Perceptually enhanced rendering of a volume data set. The visualization using the Phong shading model is shown in (a). The result of simulating ambient occlusion with one of the proposed techniques (the screen-space method) is shown in (b). Note how the back ribs are darkened. In (c) a blueish halo is also rendered in order to emphasize the shape of the structures.*

evaluate the proposed techniques, we carried out an informal user study that is described in Section 5.4. Finally, conclusions are presented in Sections 5.5.

5.1 Preliminaries

The approaches described in this chapter are based on Summed Area Tables (SAT) [24] to simulate ambient occlusion and halos in real time. Although Summed Area Tables may have n dimensions, this section describes 2D SATs, which are used to compute the *Vicinity Occlusion Maps*. The 3D version is presented in Section 5.3.1, before introducing the *Density Summed Area Table* approach. The concept of ambient occlusion is also reviewed in this section.

5.1.1 Summed Area Tables

A Summed Area Table[1], also known as *integral image* [109] in *Computer Vision*, is a 2D grid that stores, for each cell $SAT[x, y]$, the sum of the elements of a given table t, from its origin $t[0, 0]$ until the cell of interest $t[x, y]$. Thus, the value of each cell of the SAT is computed as follows:

[1]Broadly speaking, the term *summed area table refers* to the 2D case.

$$SAT[x, y] = \sum_{i=0,j=0}^{x,y} t[i, j] \tag{5.1}$$

This computation may be performed incrementally by taking the values of the neighbouring cells into account, as it is depicted in Figure 5.2. As a consequence, the cost of building a Summed Area Table is linear with the number of cells of the original table t.

The main advantage of using Summed Area Tables is that they provide a way to filter arbitrarily large rectangular regions of an image in a constant amount of time: to compute the sum of the values of a rectangular region of t, only 4 accesses are required in the worst case (see Figure 5.3), independently of the size of the region. This property allows us to obtain, in a constant time, the information needed to simulate ambient occlusion and halos as it is described in Section 5.2.

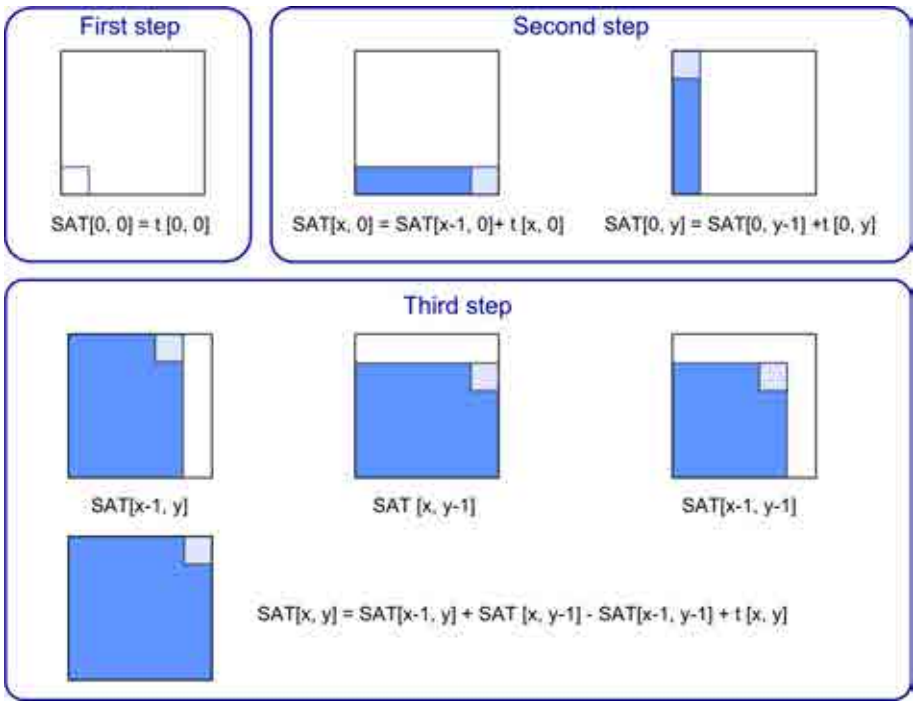


Figure 5.2: *Summed area table computation from a given table t. The algorithm can be implemented incrementally on the CPU by following the three depicted steps.*

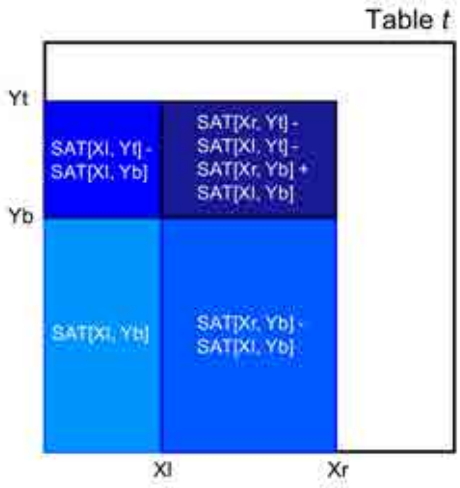


Figure 5.3: *Computing the sum of different rectangular areas of the table t. Note that 4 SAT accesses are needed at most, independently of the size of the rectangular regions.*

5.1.2 Estimation of the ambient occlusion

Ambient occlusion refers to the modulation of the ambient light on a surface point due to the occlusion of its surroundings. Given a point on the surface p with normal $\vec{n}$, the occlusion factor due to the environment of p is computed using the following equation [21]:

$$A(p,\vec{n}) = \frac{1}{\pi}\int_{\Omega} Vis(p,\vec{\omega})(\vec{n}\cdot\vec{\omega})d\omega \tag{5.2}$$

where $\vec{n}$ and $\vec{\omega}$ are normalized, Ω represents the hemisphere over the point (i.e. the set of directions $\vec{\omega}$ for which $\vec{n}\cdot\vec{\omega} > 0$), and $Vis(p,\vec{\omega})$ is a binary function whose value is 1 if p is occluded in the direction $\vec{\omega}$, and 0 otherwise.

In *Computer Graphics*, the ambient occlusion simulation was introduced with the name of *obscurances* by Zhukov *et al.* [122]. Their purpose was to provide an illumination model to account for the ambient light in a more accurate way than the Phong model, but without the complexity of other approaches such as radiosity. Based on the *obscurances*, Stewart was the first to propose an approach to simulate ambient occlusion in *Volume Visusalization*, calling it *vicinity shading* [100]. This method is based on attenuating the illumination that arrives to a given voxel to shade by measuring the occlusions in its local vicinity. In order to do this, the amount of occlusion is estimated by sampling the environment of the voxel with more than 1000 sampling directions. Although the different rays involved in the process are traced in an efficient way, sampling such a high number of directions results in an important impact on rendering time.

Since the *vicinity shading* method was presented, other approaches have been proposed to simulate ambient occlusion for volume data sets (see Chapter 3). The next sections of this chapter describe our proposals to simulate ambient occlusion in real time.

5.2 Screen-space ambient occlusion and halos

This section describes a view-dependent approach to simulate ambient occlusion and halos in DVR. As it has been stated in the previous section, ambient occlusion estimates the occlusion due to the environment of each voxel, and modifies the lighting contribution accordingly. Our proposal is based on approximating the ambient occlusion in image space, by analyzing the depth information of the structures projected in the vicinity of a pixel to shade. Thus, given a rendered image of the data set and considering that the structures that are closer to the observer occlude part of the ambient light, the shading of each pixel is modified according to the average depth of its vicinity.

On the other hand, in order to draw halos around a certain structure, the color of the surrounding pixels has to be changed. To do this, we propose to use the average depth of the vicinity to define an intensity function that decays according to the distance to the structure to be highlighted. By combining a user-defined color with this intensity function, the halos are obtained.

In order to efficiently compute the average depths required for simulating ambient occlusion and halos, a new data structure called *Vicinity Occlusion Map* [30] (*VOM*) is introduced next.

5.2.1 Vicinity Occlusion Maps

In order to simulate ambient occlusion and halos in a fast way, we have designed the *Vicinity Occlusion Map* (VOM), which consists of two Summed Area Tables: SAT_{depth}, which is computed directly from the depth map, and $SATN_{depth}$, which is computed from a bitmap containing 0 for the background pixels of the depth map, and 1 for the rest of the pixels. Thus, the average depth of arbitrarily big vicinity regions and the number of pixels that effectively contribute to the average, can be computed in a fast way.

Two different remarks regarding the depth map have to be taken into account: the first one is that the depth map used to compute the VOM only encodes the depth of the first sample that reaches full opacity along the viewing rays. As a consequence, just opaque structures and the semi-transparent ones that contribute to reach full opacity are considered when simulating ambient occlusion and halos. The second remark is that depths values are codified in the range [0,1], being 0 the furthest point of scene and

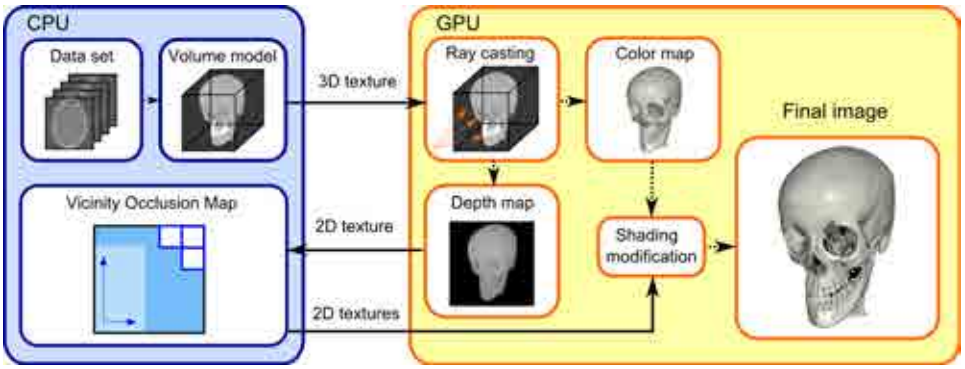


Figure 5.4: *Application architecture: The CPU stores the a volumetric data set in a 3D texture and uploads it to the GPU. There, the ray casting process generates a color and a depth map. Afterwards, the depth map is passed to the CPU and used to compute the Vicinity Occlusion Map (VOM). By using the VOM, the shading of the the color map is modified on the GPU to generate the final visualization.*

1 the closest one [2]. By assigning a depth value of 0 to background pixels instead of 1, less number of bits are required to codify the SAT_{depth} values when there are many background pixels and, therefore, memory consumption is reduced.

Counting how many pixels of the vicinity have different properties than the pixel to shade (depth values in our approach) is a time-consuming task that highly decreases the performance of the rendering process. In order to address this issue, the screen-space ambient occlusion used by Crytek [76] is based on subsampling the neighbouring region of each pixel using a specific pattern. Unfortunately, such a sampling produces visual noise artifacts and filtering the result is needed to obtain a smoothed shading. Instead, our approach adopts a completely different strategy based on computing the average depth of neighbouring pixels using Summed Area Tables. Thus, the average value can be obtained in a fast way (4 SAT queries at most) and there is no need of filtering the result.

5.2.2 Algorithm overview

The algorithm used to simulate ambient occlusion and halos with the VOM data structure comprises three main stages performed alternatively on the CPU and the GPU (see Figure 5.4): initially, the volume model is loaded on the CPU and stored as a 3D texture. Then, this texture is passed to the GPU where a ray casting process is performed. As a result, this first rendering pass generates a color map, containing the visualization of the

[2] Inversely to OpenGL, where 0 corresponds to the zNear plane and 1 to the zFar.

volume using the Phong shading model, and the depth map that is needed to compute the Vicinity Occlusion Map.

Once the depth values have been computed in the first rendering stage, the two Summed Area Tables of the VOM are computed on the CPU, and stored as two 2D textures. After that, the generated VOM is loaded on the GPU where a second rendering step modifies the shading of the color map generated in the first pass, in order to add the ambient occlusion effect or colored halos around the structures of interest. Hence, only one ray casting process is required in the whole pipeline.

Although the VOM could also be computed on the GPU, it would require multiple rendering passes [40]. For this reason, and being that the ray casting process is an intensive algorithm, our initial approach was to download the work to the CPU. An alternative CUDA-based implementation on the GPU, is described in Section 5.2.7.

5.2.3 Ambient occlusion simulation

As it has been stated at the beginning of Section 5.2, or proposal to estimate the ambient of occlusion depends on the depth of a given pixel to shade $[x, y]$ and the average depth of its vicinity $avg_{depth}[x, y]$. Considering the given pixel centred in its vicinity, the average depth is obtained using the following equation:

$$avg_{depth}[x, y] = \frac{\sum_{i=x-size_x, j=y-size_y}^{x+size_x, y+size_y} depth[i, j]}{N_{Elems}|depth_{Elem} > 0}, \tag{5.3}$$

where $size_x$ and $size_y$ are defined by the user and determine the resolution of the vicinity ($(2size_x+1)\times(2size_y+1)$ pixels), and $N_{Elems}|depth_{Elem} > 0$ are the number of pixels that effectively contribute to the average depth. The terms of the numerator and the denominator of the equation are obtained in a fast way from SAT_{depth} and $SATN_{depth}$ of the *Vicinity Occlusion Map* respectively.

For the pixels $[x, y]$ whose vicinity is considered closer to the observer (which is determined by comparing $avg_{depth}[x, y]$ with $depth[x, y]$), the occlusion factor is approximated as follows:

$$occ = (avg_{depth}[x, y] - depth[x, y]) * factor_{vic}{}^3 \tag{5.4}$$

where $factor_{vic}$ is a user-defined scaling factor that controls the intensity of the effect in the final rendering. Although the visualizations obtained with this estimated occlusion are good enough, smoother results may be obtained by filtering this value using the *GLSL* function *smoothstep* [91],

[3] Remember that depth values are codified inversely to OpenGL, so the depth value of closer points is major than the depth of further ones.

(a) Phong shading

(b) Screen-space ambient occlusion

Figure 5.5: *Depth-enhanced volume visualization. The Phong model has been used to obtain (a), whereas the VOM technique has been applied to generate (b). Note how the overall perception of the leaves is improved in the enhanced image.*

which performs a smooth Hermite interpolation between 0 and 1 when the occlusion computed lies between two given values (in this case, 0 and 0.99):

$$dark_{vic} = smoothstep(0., 0.99, occ) \tag{5.5}$$

Finally, the Phong shading color of the pixel $[x, y]$ is modified as follows:

$$Color[x, y] = (1 - dark_{vic}) * Phong[x, y] + dark_{vic} * Color_{vic} \tag{5.6}$$

where $Color_{vic}$ is an RGB triplet that allows the user to simply darken the pixel to shade or change its color in order to emphasize it. All of these computations are performed for the pixels of the image where the volume model is projected, so background pixels are excluded. Figure 5.5 shows a comparison of a volume visualization without (left) and with ambient occlusion (right). Increasing $factor_{vic}$ reduces the overall lighting and improves the depth perception.

5.2.4 Halo rendering

In order to render halos, the same visualization pipeline as the ambient occlusion one is used (see Figure 5.4). In this case, halos have to be rendered around the structure of interest and its intensity has to decay according to the distance to the object. The computations involved in the process

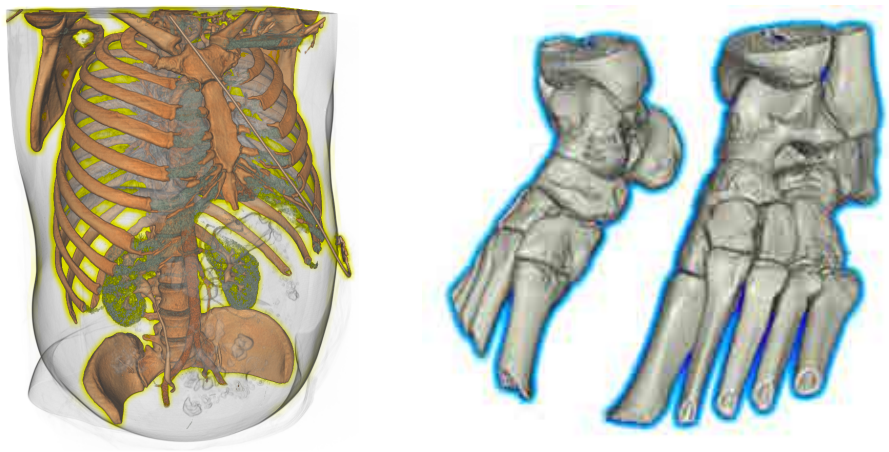


Figure 5.6: *Highlighting different structures with halos. Note that halos are just applied around opaque structures, which are the ones encoded in the depth map.*

are straightforward when halos are rendered over background pixels of the depth map (as it happens in Figure 5.6). Note that the depth of the semi-transparent pixels shown in (*a*) is equal to zero (see Section 5.2.1) and, as consequence, are not emphasized with halos.

Given a certain background pixel $[x, y]$ of the depth map, the average depth of its neighbours is computed as follows:

$$avg_{depth}[x, y] = \frac{\sum_{i=x-size_x, j=y-size_y}^{x+size_x, y+size_y} depth[i, j]}{N_{Elems}}, \tag{5.7}$$

where $size_x$ and $size_y$ are defined by the user and determine the resolution of the vicinity ($N_{Elems} = (2size_x + 1) \times (2size_y + 1)$ pixels). This equation is similar as Equation 5.3, but all the neighbouring pixels have to be taken into account in order to define the intensity function applied to the halos. This is done by using the following equation:

$$int_{halo} = avg_{depth}[x, y] * factor_{halo} \tag{5.8}$$

where $factor_{halo}$ is a user-defined weight that may be used to further increase the intensity of the effect. By taking into account the computed intensity, the color of a pixel $[x, y]$ is modified according to the following formula:

$$Color[x, y] = (1 - int_{halo}) * Phong[x, y] + int_{halo} * Color_{halo} \tag{5.9}$$

where $Phong[x, y]$ is the Phong color of the pixel, and $Color_{halo}$ is the user-defined color applied to the halo.

A more complex case is the generation of internal halos (i. e. the halos are rendered over non-background pixels of the depth map), as it is shown in Figure 5.7. In this case, the intensity function is defined as follows:

$$int_{halo} = (avg_{depth}[x, y] - depth[x, y]) * factor_{halo} \tag{5.10}$$

Note that for background pixels $depth[x, y] = 0$ and therefore, the depth of the pixel does not affect the intensity function. When rendering internal halos, the depth of the pixel has to be taken into account to define a threshold, and consider it to avoid rendering the halo at every pixel where there is a depth variation with its vicinity. Furthermore, the elements above the threshold that contribute to the average depth must be taken also into account in Equation 5.7, in order to decrease the intensity of the halo conveniently.

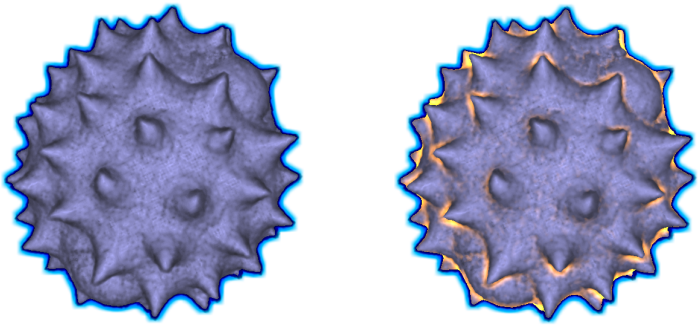
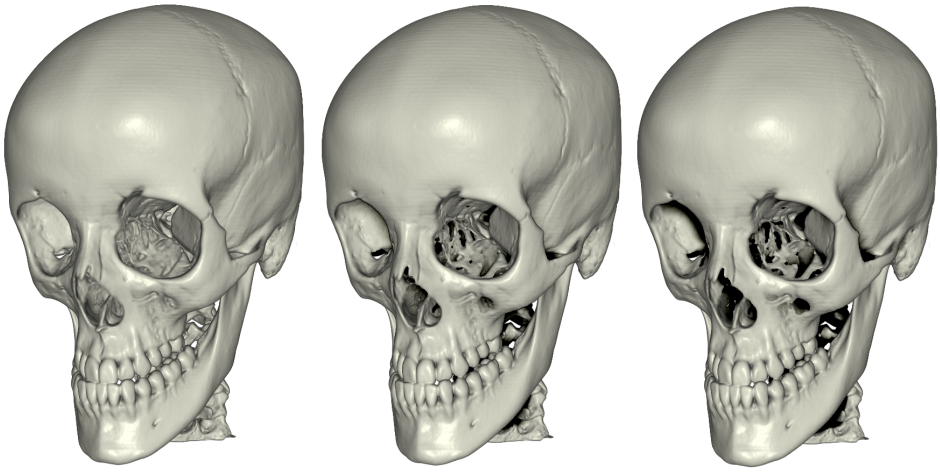


Figure 5.7: *Adding internal halos around the spikes of the pollen grain. Left image shows the halos rendered over background pixels, while the right image shows (in yellow) the internal halos generated by analyzing depth discontinuities.*

5.2.5 Additional features

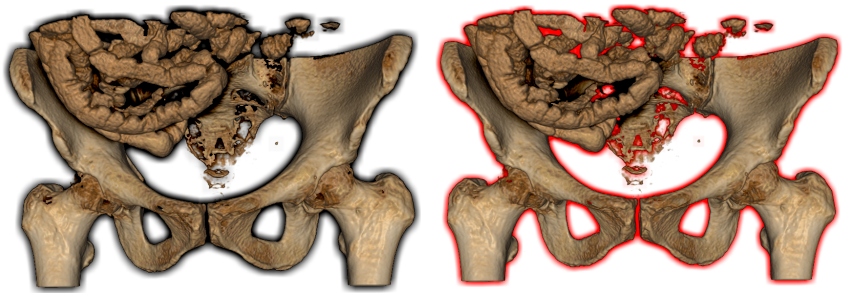
The proposed screen-space method performs a real-time computation of both ambient occlusion and halos (see Section 5.2.6). Since all the calculations are carried out from scratch each frame, the parameters involved in both effects can be changed interactively without cost. This fact allows the user to control the appearance of the effects as follows:

- **Vicinity weight and size:** depth complexity among volume models or different regions of the same model may vary, and therefore, applying a fixed function to control the appearance of the ambient occlusion may excessively darken certain regions while barely modify others. In order to prevent this issue, the user can interactively change



(a) Phong shading (b) Screen-space ambient occlusion (c) Screen-space ambient occlusion

Figure 5.8: *Effect of modifying the* $factor_{vic}$ *parameter. (a) shows the original Phong shading image and (b) and (c) show two different results obtained with a factor of* 0.39 *and* 0.78*, respectively. The higher the value of the factor is, the more perceptible the shading modification is.*



(a) Black halo (b) Red halo

Figure 5.9: *Influence of the halo color in shape perception. The black halo (a) does not provide enough shape cues because it is difficult to judge if some pixels are darkened due to the lighting contribution or the rendered halo. This issue is addressed chosing a different halo color (b).*

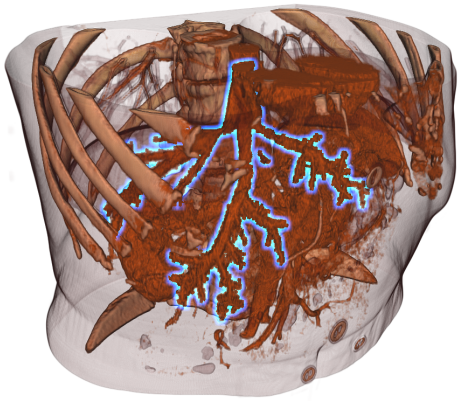


Figure 5.10: *Improving the perception of a vessel tree by rendering a halo around it.*

the default value of the parameter that controls the darkening function ($intens_{vic}$ of Equation 5.2.3). The result of such changes is shown in Figure 5.8. The resolution of the vicinity region used to compute the occlusion factor can also be interactively changed in order to fine tune the results if necessary ($size_x$ and $size_y$ of Equation 5.3). The user can also choose the emphasizing color ($Color_{vic}$ of Equation 5.6), if the depth variations need to be accentuated in a different way.

- **Halo size and color:** thanks to the use of Summed Area Tables to compute the average depths, different sizes of halos require the same number of texture accesses. Therefore, the user may change the halo size with no extra cost ($size_x$ and $size_y$ of Equation 5.3). Furthermore, the color can also be defined interactively ($Color_{halo}$ of Equation 5.9) because different structures or contexts might require a different color to emphasize the object of interest, as it is shown in Figure 5.9.

- **Structure enhancement:** the proposed method also allows the generation of local halos around specific elements of the volume. This process requires the segmentation of the structures to highlight, which are the only ones whose depth is used to compute the VOM. The depth of the other elements is used to know if the pixels of the halo are occluded by closer structures. When it happens, the color of the occluded pixels must remain untouched. Figure 5.10 shows an example where a vascular tree in the liver has been separated from the surrounding structures. Note that some ribs correctly occlude the blueish halo.

5.2.6 Results

The presented screen-space approach was implemented and tested in a computer equipped with an Intel Core 2 Duo CPU running at 3.0 GHz. It also had 4GB of RAM memory and a NVidia GeForce GTX 280 GPU with 1GB of RAM. The resolution of the volume models used was up to 512 x 512 x 512 voxels.

As it is shown in Table 5.1, real time frame rates are obtained for relatively big models. The penalty suffered by the proposed approach when generating both ambient occlusion and halos is low, compared to the cost of the ray casting process. All the examples shown in the figures of this chapter have been rendered in a viewport of 512×512 pixels, taking into account vicinity regions that go from 10×10 to 25×25 pixels. A sampling rate of 3 samples per voxel is used, in order to obtain high-quality visualizations. The ray casting process dominates the overall cost of the algorithm, and this mainly depends on the transfer function used. For this reason, similar timings may be obtained for models of different resolutions.

Concerning efficiency, the main advantage of our screen-space approach with respect to the ones that compute occlusion information in a volumetric way [92, 88, 41], is that the computational cost of estimating the amount of occlusion depends on the size of the viewport and not on the resolution of the data set. Furthermore, thanks to the use of Summed Area Tables, considering large vicinity regions around the pixels to shade does not increase the impact on rendering times because at most 4 accesses to the SAT are needed.

Although VOMs are designed to enhance the depth perception of opaque structures, they may also be used to enhance the perception of single semi-transparent isosurfaces with a determined opacity value. Note that VOMs are computed from a depth map, so the key point is to provide the depth information of the given opacity. However, the proposed technique fails when combining the depth information of different opacity layers.

When dealing with objects with disconnected components and large depth variations between them, the resulting ambient occlusion may excessively darken certain regions of the image. In such a case, a more sophisticated analysis of depth variations would yield better results. Nevertheless, this effect is not always noticeable, such as in Figure 5.1, where there are large depth variations between the closer ribs and the ones at the bottom, but the overall appearance of the enhanced image improves the depth perception in a pleasant way.

In order to improve the shading quality and overcome the aforementioned limitations due to depth variability, the occlusion of the local environment around the voxel to shade should be taken into account. This is the objective of the Density Summed Area Table approach, which is presented in the Section 5.3.

Volume data set	Volume resolution	Viewport resolution	Phong shading	VOM method
Brain	$128 \times 128 \times 58$	256×256	100.19	100.11
		512×512	100.18	58.04
Ear(Baby)	$256 \times 256 \times 98$	256×256	100.22	83.30
		512×512	87.08	43.19
Engine	$256 \times 256 \times 256$	256×256	99.00	63.98
		512×512	59.16	35.41
Intestines	$512 \times 512 \times 127$	256×256	36.12	30.89
		512×512	22.97	18.46
Bonsai	$512 \times 512 \times 182$	256×256	39.70	33.35
		512×512	24.18	19.17
Body	$512 \times 512 \times 512$	256×256	10.70	10.15
		512×512	6.46	5.67

Table 5.1: *Performance comparison with different data sets. Timings are measured in fps. Note how the impact of simulating ambient occlusion and halos is relatively low compared to the complete rendering process.*

5.2.7 Implementation details

A CUDA-based implementation to compute the *Vicinity Occlusion Maps* on the GPU was also considered to improve the performance of the proposed method. In this case, instead of computing the values of the Summed Area Tables incrementally, as it is shown in Figure 5.2, a different strategy was followed. In order to maximize the benefit of the parallel processing capabilities of the GPU, the source depth map was firstly analyzed by columns and the resulting values were then summed by rows. Although the construction times were faster than computing SATs on the CPU (see Table 5.2), the connection between the regular graphics pipeline and the CUDA pipeline presented some limitations. As a consequence, it was required to download the depth map to the CPU before uploading it to CUDA memory. A comparison between the data management of the two versions is shown in Figure 5.11. Due to the downloading process, the frame rates obtained with CUDA were similar to the ones obtained when computing the VOM on the CPU (see Figure 5.3). Newer versions of CUDA are expected to allow sharing texture memory between CUDA and OpenGL, what would save bandwidth and time. Therefore, we believe that the CUDA implementation may improve the performance of the proposed approach.

Viewport resolution	CPU	GPU (CUDA)
256×256	0.608	0.327
512×512	2.427	1.450
768×768	17.951	1.982

Table 5.2: *Comparison of the VOM construction time (milliseconds) on the CPU and the GPU using CUDA.*

Volume data set	Viewport resolution	VOM method (CPU)	VOM method (CUDA)
Brain	256×256	100.11	100.14
	512×512	58.04	60.11
Ear(Baby)	256×256	83.30	98.81
	512×512	43.19	46.45
Engine	256×256	63.98	85.72
	512×512	35.41	37.48
Intestines	256×256	30.89	34.90
	512×512	18.46	18.80
Bonsai	256×256	33.35	38.01
	512×512	19.17	19.68
Body	256×256	10.15	12.09
	512×512	5.67	7.21

Table 5.3: *Comparison between the timings (fps) obtained using the original algorithm (VOM (CPU)) and the CUDA-based version (VOM (CUDA)).*

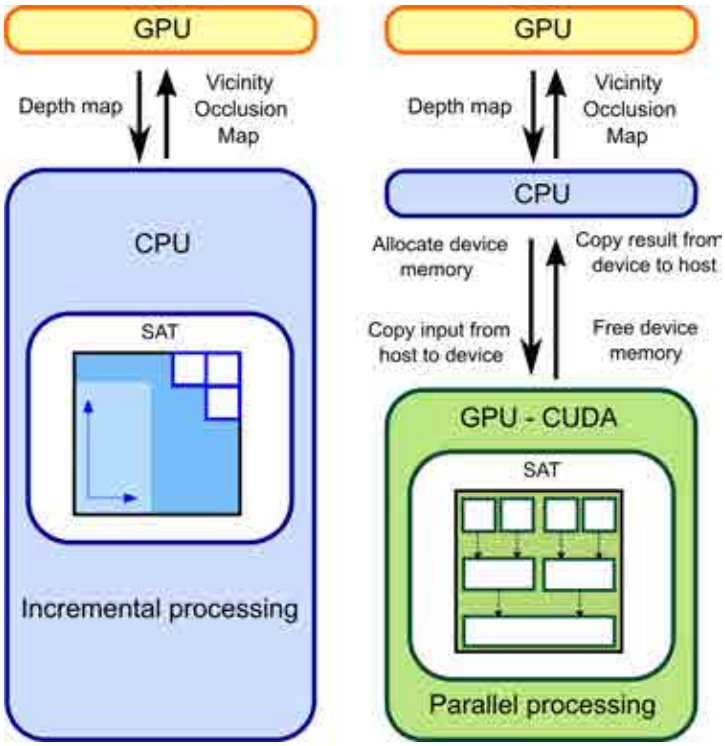


Figure 5.11: *Data management needed when computing the VOM on the CPU and the one required by the CUDA version. The CPU approach requires downloading the depth map to the CPU and uploading the VOM to the GPU. On the contrary, the CUDA version needs to download the depth map to the CPU, upload it again to the GPU to perform the CUDA computation and reverse the process to compute the final shading on the GPU.*

5.3 Volumetric ambient occlusion

The screen-space approach presented previously is based on depth information and does not consider the volume around the point to shade, but only the 2D projection of the data set on the image plane. Due to the fact that depth values are computed when the viewing rays reach full opacity, the occlusion produced by semi-transparent structures does not contribute to the final shading. This section describes a view-independent method that modifies the shading of the opaque structures, but taking into account the ambient occlusion produced by semi-transparent materials.

The basic idea of the new method is to simulate the ambient occlusion at the voxel to shade, by evaluating the opacity of its local environment and reducing the lighting contribution accordingly. In order to do this, we propose to compute the average density of the vicinity, and use the opacity

assigned by the transfer function to this average value as occlusion factor. In the same way as *Vicinity Occlusion Maps* are based on Summed Area Tables to speed up the computation of average depths, the next section describes the concept of *Density Summed Area Table*, which accelerates the computation of the average densities needed to approximate the occlusion due to the neighbouring voxels.

5.3.1 Density Summed Area Tables

Given a volumetric data set V, a *Density Summed Area Table* (DSAT) stores in each cell $DSAT[x, y, z]$, the sum of density values of V from its origin $V[0, 0, 0]$ to the cell of interest $V[x, y, z]$:

$$DSAT[x, y, z] = \sum_{i=0,j=0,k=0}^{x,y,z} V[i, j, k] \tag{5.11}$$

The construction of a 3D Summed Area Table (3DSAT) is similar to the 2D version, and it can also be performed incrementally, as it is shown in Figure 5.12.

In order to compute the sum of the values of a given cuboid region of the volume V, eight queries are needed instead of the four required in the 2D version (see Figure 5.13).

5.3.2 Algorithm overview

The algorithm to simulate ambient occlusion with a DSAT consists of two main stages (see Figure 5.14): firstly, the 3D Summed Area Table of density values is computed on the CPU. Then, the DSAT and the volume model are passed to the GPU, where a ray casting process renders the volume. While sampling the rays, the shading of the opaque samples is modified according to the occlusion caused by their surrounding structures, which is computed by analyzing the average opacity of the sample's vicinity. Thus, the visualization enhanced with ambient occlusion is obtained.

In contrast with the *Vicinity Occlusion Maps*, the DSAT is computed in a preprocessing step. As a consequence, higher frame rates are obtained, since there is no need to further transfer information from the GPU to the CPU and backwards.

One of the advantages of computing a 3D Summed Area Table of density values instead of opacities, is the fact that there is no need of recomputing the DSAT when the transfer function changes. When this happens, the opacity used to estimate the amount of occlusion and, as a consequence, the resulting shading may change, but the average density of the vicinity region remains constant. However, the main drawback of this approach is that depending on the size and the variability of density values within the

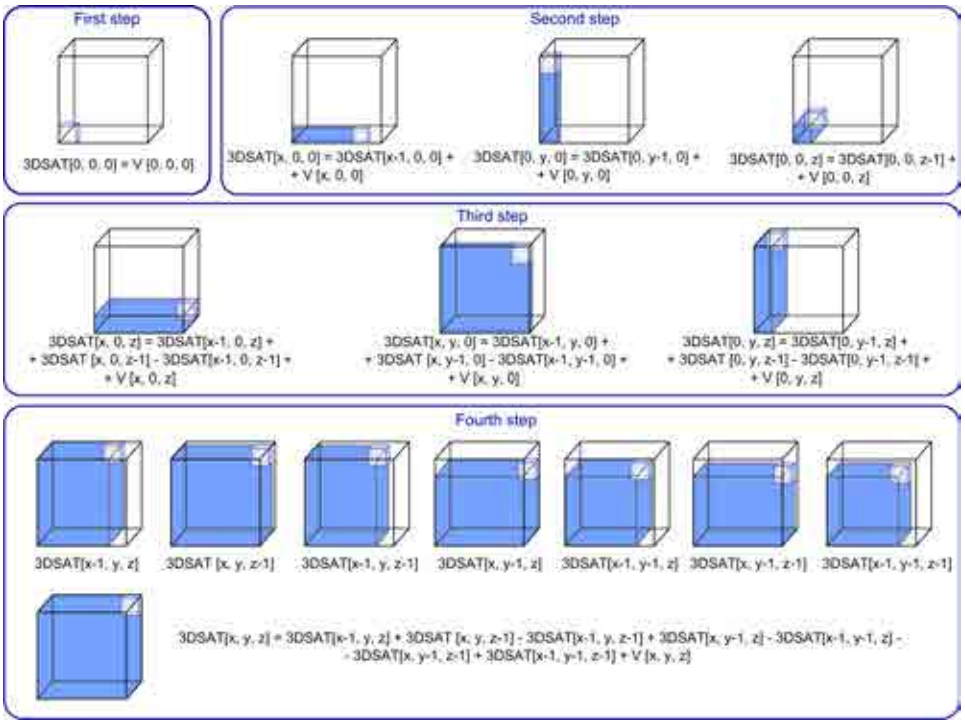


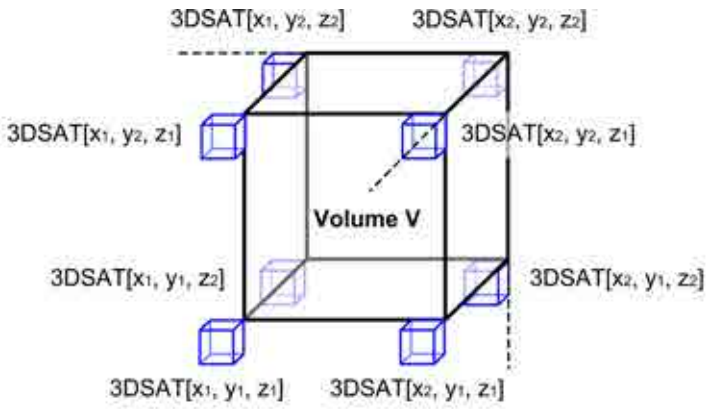
Figure 5.12: *Construction of the 3D Summed Area Table on the CPU. The first three steps depict the construction of three layers of the Summed Area Table, from which the rest of the values are computed incrementally.*

vicinity region, the opacity assigned to the average density may be completely different to the real one, which may produce noticeable artifacts in the final visualization. In order to obtain a more accurate approximation of the amount of occlusion, next section describes a subdivision scheme of the vicinity region.

5.3.3 Ambient occlusion simulation

The purpose of the *Density Summed Area Table* technique is to consider the opacity of the local vicinity of the voxel to shade ($V[x, y, z]$) as an indicator of the amount of occlusion. Thus, the more opaque the surrounding structures are, the more incident light is occluded. A naive approach to approximate the amount of occlusion would consist of using the opacity assigned by the transfer function to the average density of the vicinity region ($avg_{dens}[x, y, z]$). This average is computed with the following equation:

$$avg_{dens}[x, y, z] = \frac{\sum_{i=x-size_x, j=y-size_y, k=z-size_z}^{x+size_x, y+size_y, z+size_z} V[i, j, k]}{N_{Elems}}, \tag{5.12}$$



$$\sum_{\substack{i=x_1,\, j=y_1,\\ k=z_1}}^{x_2,\, y_2,\, k_2} V[i,j,k] = 3DSAT[x_2,y_2,z_2] - 3DSAT[x_1,y_2,z_2] - 3DSAT[x_2,y_2,z_1] + 3DSAT[x_1,y_2,z_1] - 3DSAT[x_2,y_1,z_2] + 3DSAT[x_1,y_1,z_2] + 3DSAT[x_2,y_1,z_1] - 3DSAT[x_1,y_1,z_1]$$

Figure 5.13: *Queries required to obtain the sum of the values of a cuboid region of V using a 3D Summed Area Table.*

where the resolution of the vicinity is $(2size_x+1)\times(2size_y+1)\times(2size_z+1)$ voxels (N_{Elems}), and the term of the numerator is computed from the DSAT. When considering large vicinity regions around the voxel to shade, the error of approximating the real opacity of the surroundings by the opacity of the average density might become excessive, producing visual artifacts in the resulting visualization.

In order to obtain a more accurate estimation of the amount of occlusion, the vicinity region is subdivided into eight subvolumes around the voxel to shade (see Figure 5.15). In this way, the occlusion of the local environment is better approximated because it depends on the opacities assigned to eight average densities, instead of just the one of the whole vicinity. The following equation shows how this value is computed:

$$occ = \sqrt{\sum_{i=1}^{8} Opacity(avg_{dens}(Vic_i))/8}, \tag{5.13}$$

where $avg_{dens}(Vic_i)$ is the average density of one of the subvolumes depicted in Figure 5.15, which is computed using Equation 5.12 for the central voxel of the subvolume. The *Opacity* function returns the opacity assigned by the transfer function to a given density value. Being *Opacity* defined in the range $[0, 1]$, the square root increases the contribution of low average opacity values, but preserves the maximum opacity value to be 1.

In the same way as it is done in the VOM method, the estimated occlusion is filtered using the *GLSL* smoothstep function as follows:

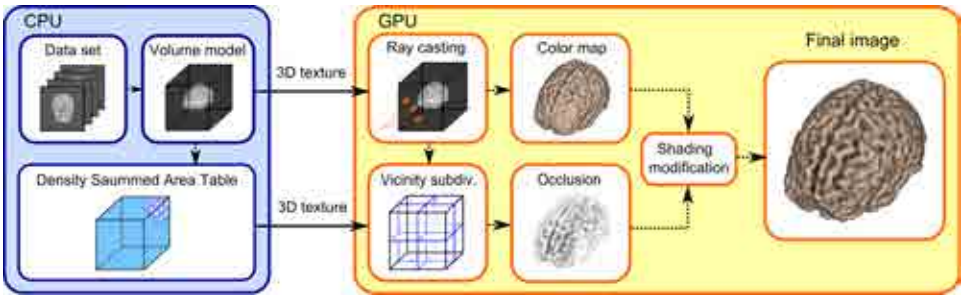


Figure 5.14: *Application architecture: The CPU stores the volumetric data set in a 3D texture and uploads it to the GPU. A 3D Summed Area Table of the original data set (DSAT) is also computed and uploaded to the GPU. There, a ray casting process is performed and the shading of the opaque samples is modified by estimating the amount of occlusion of their vicinity. The occlusion is computed in a fast way by using the information stored in the DSAT.*

$$dark_{vic} = smoothstep(min_{dark}, 0.99, occ) \tag{5.14}$$

where min_{dark} is a user-defined parameter that controls the intensity of the darkening applied.

Finally, the shading of a given sample s that belongs to an opaque isosurface is modified during the ray casting process using the equation:

$$Color(s) = Phong(s) - dark_{vic} * Color_{vic} \tag{5.15}$$

where $Color_{vic}$ allows the user to modify the color of the resulting shading.

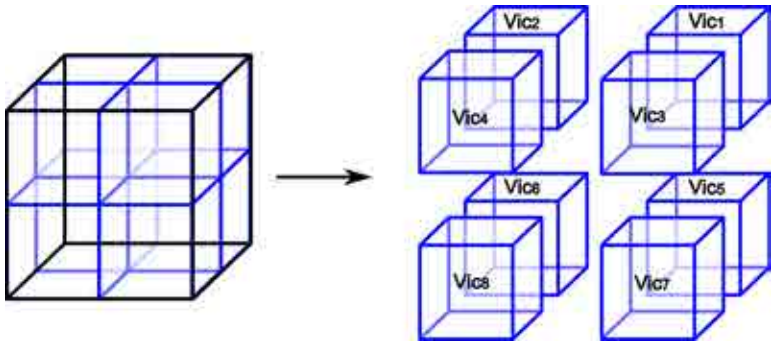
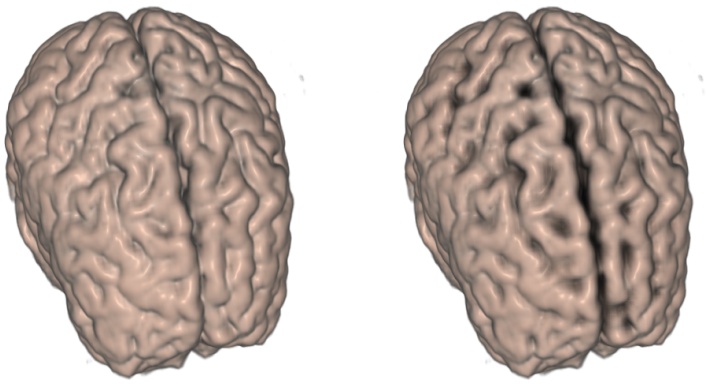


Figure 5.15: *Vicinity subdivision in 8 subvolumes in order to improve the occlusion evaluation.*

Although the subdivision of the vicinity region produces a good approximation of the occlusion for the images shown in this chapter (the size of the vicinity regions used to simulate the ambient occlusion was up to $40 \times 40 \times 40$



(a) Phong shading (b) Volumetric ambient occlusion

Figure 5.16: *Ambient occlusion simulated using the DSAT approach (b). The Phong shading visualization is also provided for comparison purposes (a).*

voxels), subdividing into eight subvolumes would not be precise enough when dealing with larger vicinity regions. A possible way to obtain more accurate approximations of the overall occlusion would be by computing different resolutions of the eight subvolumes, and weighting the occlusion factors according to the resolution of the subvolumes used. However, the performance of the whole process would be affected.

5.3.4 Results

Figure 5.16 shows an example of the volumetric ambient occlusion obtained with the DSAT method. Note how the estimated amount of occlusion due to the vicinity of the voxels on the surface produce the darkening of hollow regions, which improves the overall appearance of the final image.

As for evaluating the performance, the method was tested with the same data sets and equipment used to evaluate the screen-space approach (see Section 5.2.6). The sampling rate in the ray casting process took also 3 samples per voxel to obtain high-quality visualizations. Timings are shown in Table 5.4. Note that the frame rates obtained by simulating the ambient occlusion are similar to the ones obtained with the Phong shading model. This is due to the fact that texture queries to determine the amount of occlusion are only carried out when reaching the isosurfaces to shade, not along the whole ray traversal, which would result in an important performance penalty.

Concerning memory consumption, the DSAT stores a sum of density values. Therefore, the memory required is higher than the memory needed to store the original data set. As it is shown in Table 5.5, an additional 32 bit 3D texture with the same dimensions as the original data set (8 or 16

Volume data set	Viewport resolution	Phong shading	DSAT method
Brain	256×256	100.19	100.18
	512×512	100.18	100.15
Ear(Baby)	256×256	100.22	100.20
	512×512	87.08	80.09
Engine	256×256	99.00	94.63
	512×512	59.16	58.17
Intestines	256×256	36.12	35.82
	512×512	22.97	22.04
Bonsai	256×256	39.70	38.82
	512×512	24.18	22.84
Body	256×256	10.70	10.40
	512×512	6.46	6.25

Table 5.4: *Timings obtained for different data sets (fps). Note that the impact on rendering of simulating the ambient occlusion with the DSAT is almost equal to the timing obtained with the Phong shading. This is due to the fact that the occlusion of the vicinty is only evaluated for the voxels belonging to opaque structures.*

bits) is required to store the DSAT in the worst case. Fortunately, although the amount of extra storage is rather high, it is still acceptable for relatively big volume models, such as the *body*.

When visualizing opaque structures that do not present isolated disconnected components, the resulting shading is similar as the one obtained with the screen-space method (see Figure 5.17). Other visualizations like the one shown in Figure 5.18, may produce quite different results. In this case, the VOM approach modifies the shading in those regions of the image where there are high depth variations, allowing the viewer to identify which parts of the intestines are closer. On the contrary, due to the reduced size of the vicinity regions used in the DSAT approach, the ambient occlusion estimated in a volumetric way improves the perception of local superficial details.

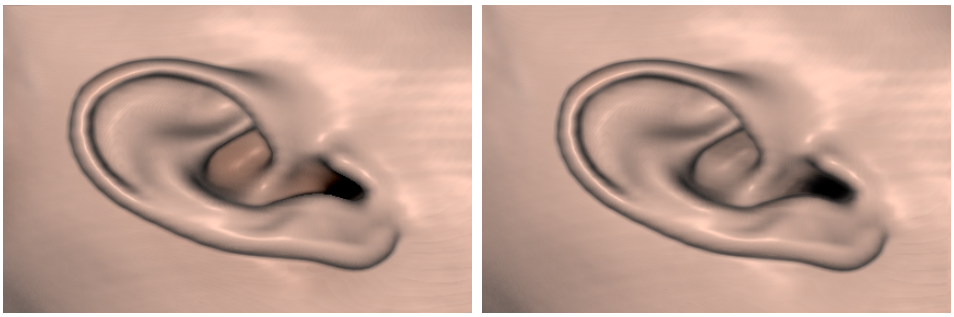
An analysis of the impact on rendering times obtained with the two methods in a viewport of 512×512 pixels is shown in Table 5.6. Note that the impact is lower for the DSAT approach because the 3D Summed Area Table is computed once in a preprocessing step. In contrast to this, the VOM is computed on-the-fly and must be updated when the viewing direction is modified.

Volume data set	DSAT computation	Maximum value	# bits required
Brain	0.04	31238068	25
Ear(Baby)	0.74	198720120	28
Engine	2.07	187475047	28
Intestines	5.75	4294967192	32
Bonsai	6.08	136596356	28
Body	17.1	4294967162	32

Table 5.5: *DSAT features. The second column shows the time required to compute the DSAT (in seconds). The other columns show the maximum value stored for different data sets and the number of bits needed to store the values correctly.*

Volume data set	Phong shading	VOM method	DSAT method
Brain	100.18	58.04 (-42.06%)	100.15 (-0.03%)
Ear(Baby)	87.08	43.19 (-50.39%)	80.09 (-8.02%)
Engine	59.16	35.41 (-40.14%)	58.17 (-1.67%)
Intestines	22.97	18.46 (-19.65%)	22.04 (-4.05%)
Bonsai	24.18	19.17 (-20.71%)	22.84 (-5.55%)
Body	6.46	5.67 (-12.20%)	6.25 (-3.25%)

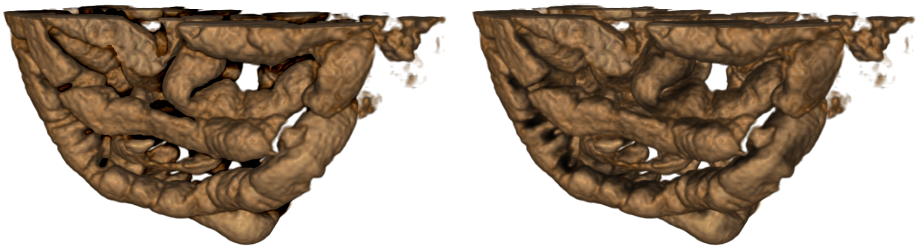
Table 5.6: *Comparison of the impact incurred by using the both methods presented in this chapter in a viewport of* 512×512 *pixels. Note how the penalty of the DSAT technique is really low compared to the visualization obtained in the ray casting using the Phong model (up to an 8% at most). Timings are measured in frames per second (fps).*



(a) Screen-space ambient occlusion

(b) Volumetric ambient occlusion

Figure 5.17: *Comparison between the ambient occlusion generated with the VOM (a) and the DSAT (b) approaches. When there are no large depth variations in the region to shade, both results are quite similar.*



(a) Screen-space ambient occlusion

(b) Volumetric ambient occlusion

Figure 5.18: *Comparison between the ambient occlusion generated with the VOM (a) and the DSAT (b) approaches. The proposed screen-space ambient occlusion enhances the regions where there are large depth variations, whereas the volumetric approach modifies the shading of the voxels at surface level according to the occlusion of their local vicinity.*

5.3.5 Implementation details

A CUDA-based implementation of the algorithm to compute the DSAT using the GPU was also considered to decrease the time spent in the pre-processing step. In order to maximize the performance taking advantage of the parallel processing capabilites of the GPU, the 3D Summed Area Table computation analyzed the elements of the input volume V as follows:

$$\begin{aligned} SX[i,j,k] &= \sum_{x \leq i} V[x,j,k] \\ SY[i,j,k] &= \sum_{y \leq j} SX[i,y,k] \\ 3DSAT[i,j,k] &= SZ[i,j,k] \\ &= \sum_{z \leq k} SY[i,j,z] \\ &= \sum_{z \leq k} \sum_{y \leq j} SX[i,y,z] \\ &= \sum_{z \leq k} \sum_{y \leq j} \sum_{x \leq i} V[x,y,z] \end{aligned}$$

Thus, the CUDA-based algorithm consisted of the three following steps:

1. Computing SX in a first pass, Z-slice after Z-slice, by loading the rows of the slices in shared memory, performing a parallel computation for the rows [37], and writing them back.

2. Computing SY in a single pass, reading memory with a GPU-friendly pattern (coalesced reads).

3. Compute SZ in a single pass, reading memory with a GPU-friendly pattern (coalesced reads).

As it is shown in Table 5.7, the computation time using CUDA is faster than the computation on the CPU, and substantially lower than the time required to load the volume. Loading times are provided to determine if the DSAT computation is noticeable, or implies a relevant penalty with respect to the normal process. As happened with the CUDA-based computation of *Vicinity Occlusion Maps*, the bottleneck was the data communication bus, because the resulting information had to be downloaded to the CPU, as depicted in Figure 5.19. This is something expected to be solved as soon as Frame Buffer Objects are incorporated to CUDA.

Volume data set	**Loading time**	**CPU**	**GPU (CUDA)**
Brain	0.12	0.04	0.11
Ear (baby)	0.20	0.74	0.18
Engine	0.46	2.07	0.31
Intestines	1.42	5.73	0.74
Bonsai	1.87	6.08	0.75
Body	3.58	17.10	1.64

Table 5.7: *Time (in seconds) required to compute the DSAT on the CPU and the GPU using CUDA. Note that the CUDA computation is faster than loading the data set, which is an acceptable result.*

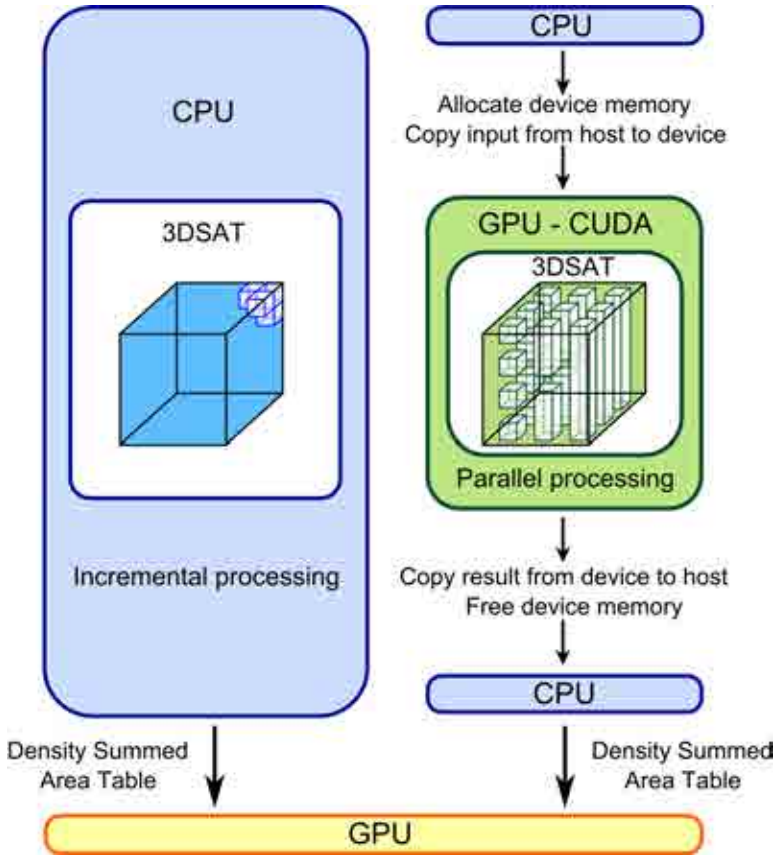


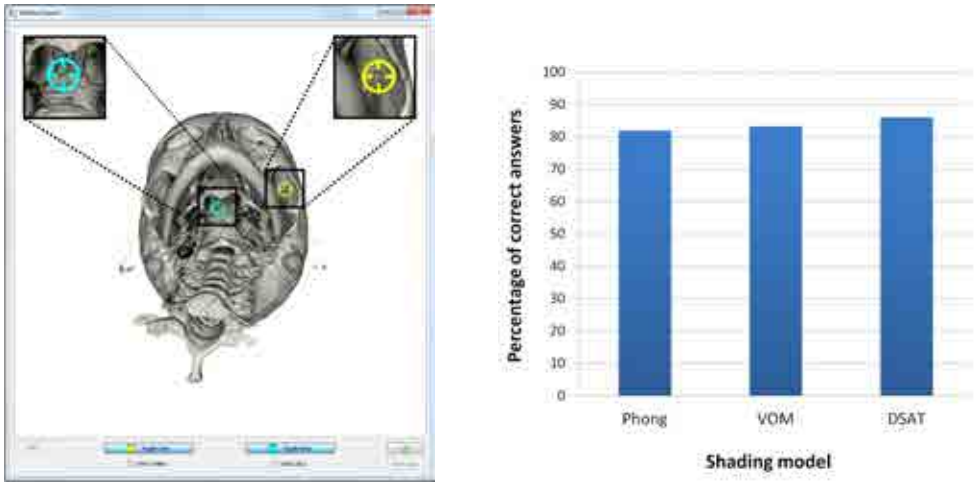
Figure 5.19: *Comparison between the data management needed when computing the DSAT on the CPU and the one required by the CUDA version. Several data transfers between the CPU and the GPU are needed to incorporate the CUDA-based algorithm to the visualization pipeline of the DSAT approach (see Figure 5.14)*

5.4 User study

In order to evaluate the effectiveness for enhancing the depth perception of the ambient occlusion simulated with the two proposed techniques, we carried out a user study based on the one presented by Lindemann and Ropinski in [63]. The study consisted of three different tasks: two of them evaluated the relative and absolute depth perception, whereas the third one was designed to provide a subjective evaluation of our two approaches. A set of 12 people, all of them with computer graphics background, took part in the study.

For the first task, which was the one designed for evaluating the relative depth perception, we compared Phong shading with the ambient occlusion obtained with the VOM and the DSAT techniques. To this end, we gen-

erated a set of 18 images from 6 different volumetric datasets, where each shading approach appeared 6 times. Two circular markers pointing at different elements of the scene were added to each image (see Figure 5.20 (a)), and users had to select the point that they considered closer to the viewer. Images were shown randomly to prevent a learning effect, and the task was evaluated by measuring if the selections were correct.



(a) Applet used for task 1 (b) Percentage of correct answers

Figure 5.20: *Applet and results of the relative depth perception task. This task consisted of selecting the closest element pointed by the markers showed in (a). The percentage of correct answers for each shading model is shown in (b).*

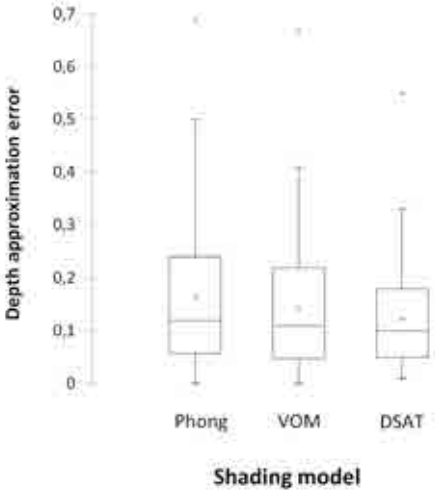
In order to analyze the results of this first task, we used a Cochrane's Q test with a confidence level of $\alpha = 0.05$, since the response variable took just two possible results: 1 if the selected point was the closest one and 0 otherwise. The results of this test showed no significant difference in the percentage of correct answers for the three different techniques (*p-value* $= 0.798 \gg \alpha = 0.05$). Figure 5.20 (b) shows the percentage of correct answers for the Phong shading (81.94%), and the ambient occlusion obtained with VOMs (83.33%) and DSATs (86.11%).

The second task was designed to evaluate the absolute depth perception. As well as it was done in the first task, Phong shading and our two approaches were used to generate a set of 18 images, where each technique appeared 6 times. For this task, a circular marker pointing to an element of the scene was added to each image, as can be seen in Figure 5.21 (a). In this task, users had to estimate the absolute depth in percentage, in relation to the distance between the closest visible element of the scene and the furthest one. Images were shown randomly and the error between the estimated and

the real depth of the marked point was measured.

The results of the second task were analyzed using a one-way analysis of variance (ANOVA) with a confidence level $\alpha = 0.05$, to reject the null hypothesis that all means of user's approximated depths were equal between techniques. The results of this test showed no significant difference between the three compared techniques ($F = 2.04$, $p\text{-}value = 0.13 > \alpha = 0.05$). Figure 5.21 (b) shows the error produced by users when approximating the absolute depth of the marked points.

(a) Applet used for task 2



(b) Results of the absolute depth approximation

Figure 5.21: *Applet and results of the absolute depth perception task. It consisted of estimating the depth in percentage of the marked point, compared to the depth between the closest visible point of the scene and the furthest one. The error produced while approximating the depth of the marked points is shown in* (b).

Finally, we also wanted to know the opinion of the users about the shading techniques used in the study. This was addressed by presenting 6 pairs of images to the users, comparing the Phong shading result with the VOM and the DSAT visualizations respectively. Images were shown randomly and users had to choose which image of each pair they considered better to perceive depth and spatial relations between the elements of the scene. In this task, we just measured the users' choices. The Cochrane's Q test with a confidence level of $\alpha = 0.05$ was used to evaluate this task, and the results showed a significant difference ($p\text{-}value < 0.0001 \ll \alpha = 0.05$) between Phong shading, which was chosen the 5.5% of times and VOM (94.5%). As well, the test showed a significant difference ($p\text{-}value = 0.046 < \alpha = 0.05$)

between Phong (chosen the 33.3% of times) and the DSAT (66.6%).

The results of the third task show that users considered the enhanced visualizations better to perceive the depth perception and spatial information of the visualized data sets. However, despite users solved the relative and absolute depth perception tasks slightly better with the ambient occlusion shading, no significant differences between the three techniques were found analyzing the results.

Apart from the correctness of the answers and the error of the depth approximation, we also considered measuring the time to complete these two tasks, as it is done in [63]. However, we observed that time depended much more on the quantity of information provided by the image, than the shading technique used to generate the visualization. This observation could explain the fact that no clear relations between users' answers and time were found in the study by Lindemann and Ropinski. Based on the observed results, we consider that a more detailed study would be of interest, in order to obtain a more significant evaluation of the proposed techniques. Concretely, we believe that by increasing the number of images, data sets used, and participants, the results of the relative and depth perception tasks may be better for the ambient occlusion approaches, as the subjective evaluation shows.

5.5 Conclusions

This chapter has presented two novel approaches to enhance the depth perception in *Direct Volume Rendering*, which are based on simulating ambient occlusion and generating colored halos. In order to compute the information required to apply both effects, two different data structures have been designed: the *Vicinity Occlusion Map* (VOM) and the *Density Summed Area Table* (DSAT), which take advantage of Summed Area Tables to speed up the computations.

The first method presented [30] is a screen-space approach that uses depth information to estimate the amount of ambient light occluded at each pixel of the image, and modifies the shading accordingly. Thanks to the use of the VOM to compute the amount of occlusion, the impact on rendering is constant per frame, and depends on the size of the viewport. By using the same depth information stored in the VOM, colored halos may also be rendered around the structures of interest in order to highlight them.

The second approach [29] is a view-independent method used to simulate ambient occlusion. In this case, the ambient occlusion is approximated in a volumetric way by evaluating the opacity of the local vicinity of the voxels to shade. In order to obtain the required information in a fast way, a 3D Summed Area Table of the density values (DSAT) of the original data set is computed. This method produces similar images to the ones

obtained with the screen-space approach, except when visualizing discontinuous or separated structures. In this situation, the VOM method enhances the depth discontinuities between structures whereas the DSAT technique applies ambient occlusion locally. The impact on rendering times of the DSAT approach is lower than the VOM method, but memory consumption is higher. de 11

6

Depth-enhanced Maximum Intensity Projection

Maximum Intensity Projection (MIP), originally called *Maximum Activity Projection* (MAP) [111], is a well-known volume rendering technique where, in contrast to *Direct Volume Rendering* (DVR), the highest-intensity values along the viewing rays are projected into a 2D image. Therefore, it is particularly valuable for the interpretation and evaluation of data sets where the features of interest present high intensities, such as in *Positron Emission Tomography* (PET), *CT* or *Magnetic Resonance Angiography* studies. Although the generation of MIP visualizations is computationally fast and does not require a complex transfer function definition process, the main limitations of this approach are the fact that higher intensity values may conceal features of interest with lower intensities, and the loss of depth cues and contextual information, which often produce ambiguous visualizations of the volume model. Figure 6.1 (b) shows one example of high ambiguity, where it is difficult to perceive the correct orientation of the model, whereas it is clear in the direct volume rendering visualization (a).

Different approaches have been proposed in the past in order to address the main drawbacks of MIP. While some of them are based on changing the depth values of the structures for improving its perception [38], other techniques add color cues [89] or modify the shading [14] to convey contextual information. Despite the fact that lighting computations or other shading modifications may improve depth perception and reveal the spatial arrangement of the visualized structures, it has to be taken into account that certain intensity values displayed in the original MIP may be concealed by the new shading. Therefore, in order to preserve the maximum information of the original visualizations, the enhancement of MIP images has to be done appropriately.

This chapter presents a new algorithm to generate *Depth-enhanced Maximum Intensity Projection* images (DeMIP), which has been designed to reduce the loss of contextual information in MIP. To this end, this new approach is based on a subtle modification of the shading, in order to add depth cues (Figure 6.1 (c)) without computing illumination effects. A sphere-based

color mapping extension is also proposed, which slightly tinges MIP results, in order to improve the perception of spatial information in cases of high ambiguity (Figure 6.1 (d)). To evaluate the enhanced visualizations obtained with the proposed method, an informal user study has been carried out. Its results demonstrate that our technique effectively improves context and depth perception in MIP for both mono and stereo vision.

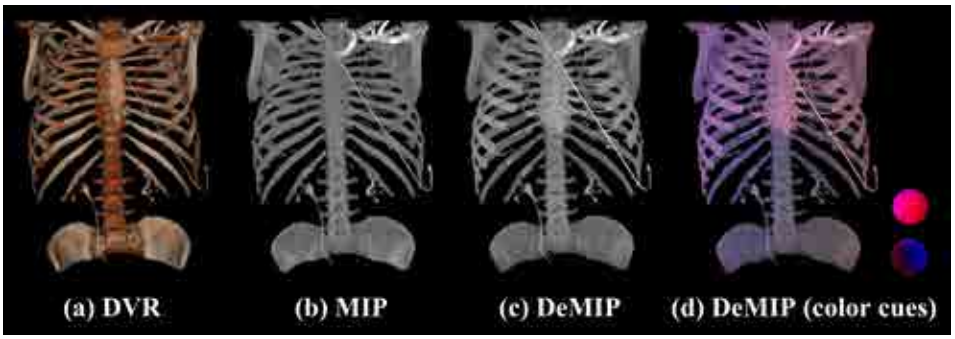


Figure 6.1: *Comparison between MIP (b) and the presented method: DeMIP (c) and DeMIP with color cues (d). The Phong shading visualization (a) is also provided for comparison purposes. The spheres in (d) show the front and back spherical color mapping that is applied to the volume model, in order to aid in the correct interpretation of the spatial distribution of the anatomical structures (see Section 6.1.2).*

The remainder of the chapter is organized as follows: Section 6.1 describes DeMIP in detail and some discussion about different implementation details tested during the development of the approach are reviewed in Section 6.2. The results of DeMIP are shown in Section 6.3 and the user study is described in Section 6.4. Finally, Section 6.5 presents the conclusions.

6.1 Enhancing depth perception

Direct volume rendering provides occlusion cues and illumination effects that generally convey correctly the spatial relationships among the structures of the volume. When the Phong shading model does not obtain quite accurate results, there are more complex techniques that may improve the final visualization, as the ones presented in the previous chapter do.

In order to improve the depth perception in MIP images, occlusion information and shading effects can be computed in a similar way to DVR. Unfortunately, since MIP displays intensity values, the addition of illumination effects or color variations may hide important information provided by the original image. For this reason, instead of mixing DVR shading with the intensity values, DeMIP provides occlusion information by means

of a slightly modification of the displayed intensities taking the depth of the structures of the volume into account. Thus, the shape of the structures is perceivable with a minimum loss of information with respect to MIP.

The use of stereoscopic images might also be a good choice to convey depth and spatial information in MIP. However, the fact that the intensities displayed in nearby pixels of MIP images can be located at different depths or belong to different structures, produces perceptual problems and, as a consequence, the 3D coherence provided by stereo vision is generally lost. On the contrary, as DeMIP enhances the depth perception and preserves the spatial information, stereoscopic rendering can be effectively used with our approach, improving the overall perception of the volume model.

Depending on the viewing direction, adding depth and occlusion cues by a subtle modification of intensity values, might be not enough to disambiguate MIP images. In these cases, DeMIP provides the user with a tool that codifies the 3D location of the displayed structures by means of a spherical color map. Thus, depth perception is clearly enhanced in the most complex cases.

6.1.1 Algorithm overview

The algorithm used to obtain the depth-enhanced visualizations comprises two main stages: the first one is responsible for computing the DeMIP shading, whereas the second one improves, when needed, the perception of spatial information by adding extra color cues.

Given a certain viewing ray, the basic idea behind the enhancement of the first stage is to look for the closest sample to the observer that belongs to the same structure as the one with the maximum intensity along the ray. Then, the shading is modified according to the depth of this closest sample, in order to reveal the structures of the data set. Ideally, we would consider that a given sample belongs to the maximum intensity structure when their materials (i.e. the color assigned by the transfer function) coincide. Unfortunately, depending on how the transfer function is defined, fuzzy boundaries between the structures may produce noticeable artifacts in the final image if the same material is used. In order to address this issue, similar materials are also taken into account, which are determined by means of a user-defined threshold.

The visualization pipeline is based on the ray casting algorithm and consists of a single rendering step performed on the GPU, as it is shown in Figure 6.2. To correctly compute the depth of the closest structure, the depth value of the first sample of the ray is initially stored. While sampling the ray, this value is updated for every sample with bigger intensity and a significantly different material than the sample at the currently stored depth. At the end of the ray casting process, maximum intensity values produce the MIP image and depth values are stored in a depth map. Note

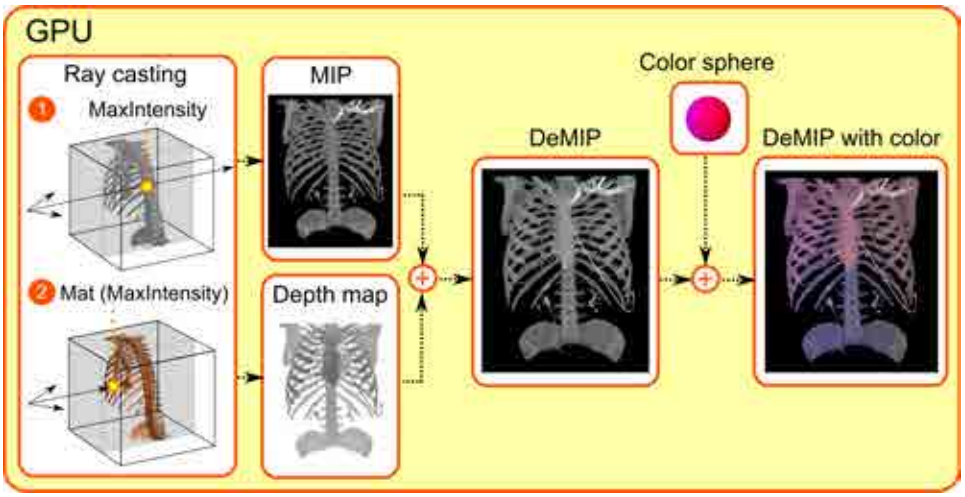


Figure 6.2: *DeMIP visualization pipeline. The maximum intensity value (1) and the first occurrence of a sample with the same material (2) are computed for each viewing ray during the ray casting process. Then, each pixel of the MIP image is shaded taking into account the depth values of the first sample and, eventually, the color provided by a color sphere.*

that when the maximum intensity value belongs to the first sample of a certain material (i.e. there are no previous occlusions), the shading is the same as MIP. Otherwise, the maximum intensity, that is used as a value of the color channels for rendering purposes ($MIP[x, y]$), is weighted with the depth ($depth[x, y]$) using the following equation:

$$DeMIP[x, y] = MIP[x, y] * (1 - d_w) + 2 * d_w * (1 - depth[x, y]) \quad (6.1)$$

where d_w is a scaling factor that controls the intensity of the enhancing effect. Note that if $d_w = 0$, the result is the MIP color. The two addends of the formula have been added in this way to slightly lighten the regions closer to the user and slightly darken the regions far from the observer, in order to increase the contrast. In most cases, assigning a value of around 0.15 to d_w is enough to perceive the occlusions (while keeping the details) of the structures of interest, though users may change this value if it does not fulfill their needs for a certain model. Figure 6.3 shows the effect of changing the d_w factor.

6.1.2 Additional color mapping

The second stage of the algorithm, which is performed in the same rendering step as the first one, provides some color cues for a further improvement of

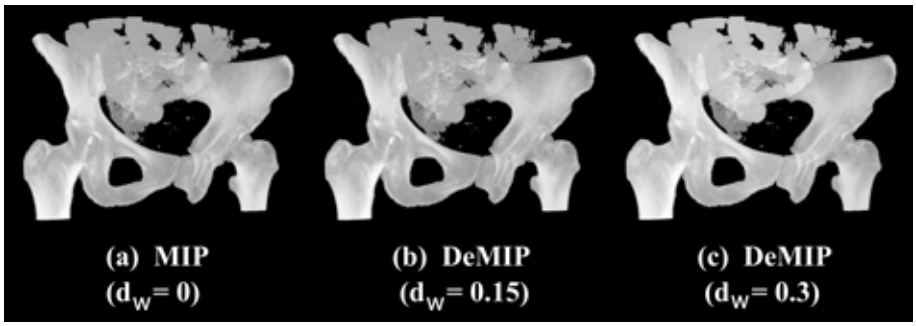


Figure 6.3: *Effect of the depth weighting (d_w) for the abdomen model. Note how the intestines are revealed when using bigger d_w factors. On the contrary, when $d_w = 0$, MIP image is obtained.*

depth perception and the spatial arrangement of the revealed structures. In this case, the 3D position of the sample at the computed depth is used to slightly tinge the MIP color using a spherical texture map. This spherical map is initially textured using a color for the front region and a different color for the back part, which are interpolated across the surface of the sphere.

In order to apply a color from the sphere to the shading of the first stage, we build a vector with initial point at the center of the bounding box of the volume and endpoint at the 3D position of the sample at the computed depth. Then, the intersection of the bounding sphere with a ray whose direction is the one indicated by this vector, determines the color from the texture map ($color_{sph}$). Finally, the color modification for a given pixel $[x, y]$ is computed as follows:

$$Color[x, y] = DeMIP[x, y] * (1.0 - sph_w) + color_{sph} * sph_w, \qquad (6.2)$$

where $DeMIP[x, y]$ is the enhanced shading of the first stage and sph_w is a factor to weigh the color of the sphere map. Figure 6.4 shows the result of the spherical-based color mapping. Note how the DeMIP shading is further improved and the perception of the model orientation is easier to infer (the correct orientation can be appreciated in the Phong shading visualization shown in Figure 6.4 (a)).

The spherical map is represented on screen with a 3D sphere that can be rotated by the user. In this way, possible ambiguities can be resolved without the need of rotating the model, since the color distribution in the rendered image can be inferred from the color sphere, which is always rendered next to the model. A small set of predefined color maps is also provided, as it is shown in Figure 6.5. Thus, users can use the one they consider more adequate.

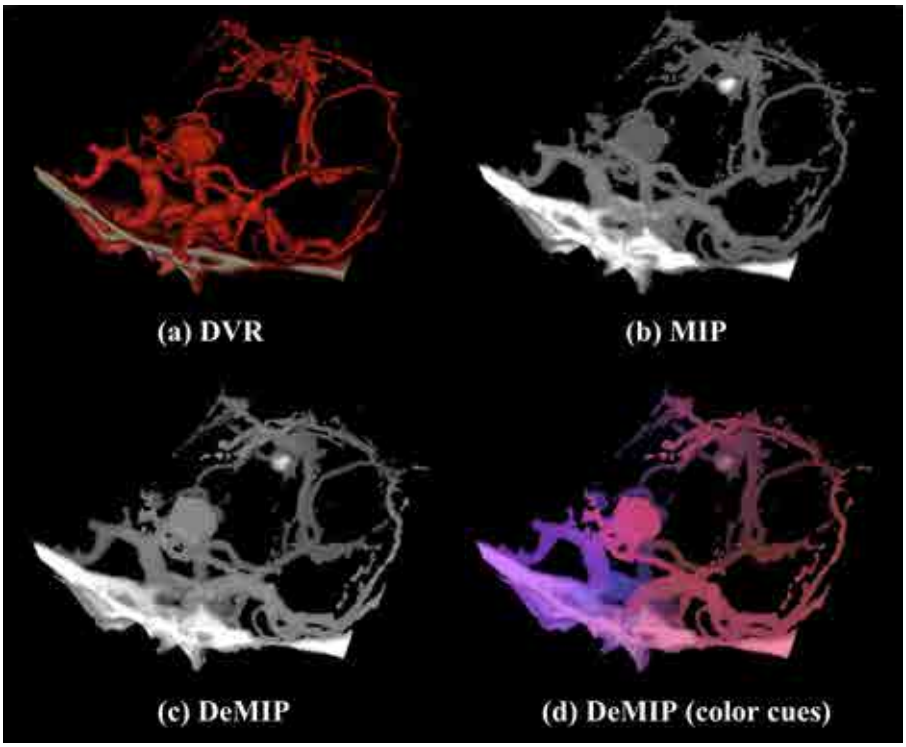


Figure 6.4: *Images of an aneurysm data set generated with MIP (b), DeMIP (c) and DeMIP with color cues (d). Note that the furthest regions of the volume shown in (a), are clearly identified by the blueish color in (c), whereas it is difficult to determine their position in (a) or (b).*

6.2 Implementation details

This section presents some discussion about different implementation details that were considered when developing DeMIP. Concretely, different alternatives to compute the depth of the closest samples were considered, in order to test which method was faster and produced better results. Furthermore, we also evaluated the possibility of splitting the algorithm pipeline into several ray casting and rendering steps, in order to avoid some comparisons in the shader that might affect the overall performance of the algorithm.

6.2.1 Depth computation

A key issue of DeMIP is a good evaluation of depth information, since it is used to modify the shading of the intensities displayed in the original MIP. As we wanted to test the suitability of DeMIP to stereo vision, each final image had to be rendered twice, once for each eye, so we evaluated

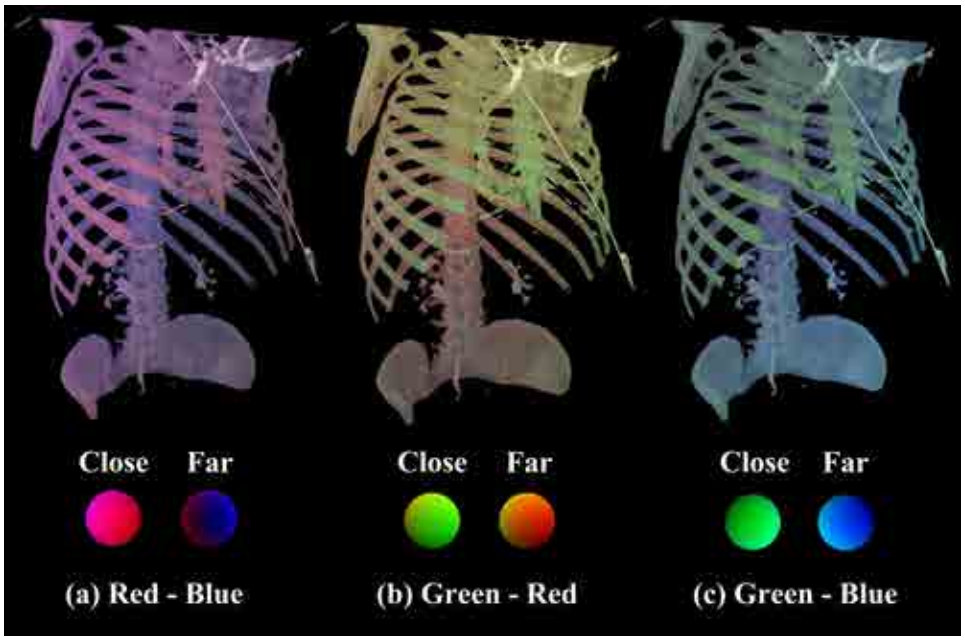


Figure 6.5: *Example of different pre-defined color maps applied to DeMIP in order to improve the spatial distribution comprehension. Although the red-blue combination can be suitable for a lot of users, some may perceive the structures arrangements better with other colors. The sphere mapping determines the color applied to the closer and the further elements of the volume.*

three fast approximations to compute the exact depth of the closest samples along each viewing ray (i.e. distance from the observer to the first sample with the same material as the maximum intensity along the viewing ray). One possible approximation would be considering the distance of the closest sample with respect to the *zNear* plane (see Figure 6.6), as it is done in OpenGL. However, since the perspective projection is used in DeMIP, these depth values might not be accurate enough to approximate the real depth for certain rays.

Another possibility would be computing the number of ray steps to reach the closest sample (see Figure 6.6). Unfortunately, the classical GPU ray casting algorithm [56] traces rays from the boundary of the bounding box of the volume, so the number of steps covered is not proportional to the distance to the viewer. Although this approach obtains good results for static views (see Figure 6.7 (b)), some artifacts are noticeable when rotating the model.

The last approximation would combine the two previous approaches: the distance to the boundary of the bounding box computed with respect to the zNear plane and then adding the number of steps along the ray.

This method yielded a value that was correctly mapped to zero at the near plane, and almost equal to the exact distance. In fact, the images obtained using this approximation were visually indistinguishable from the ones with the exact depth, and the shader computations were simpler. However, we found that this methods, though computationally cheaper, did not yield clear performance gains. Therefore, instead of using this third approach, the exact distance from the observer to the closest samples were finally computed. The results of approximating the depth with the three different methods are shown in Figure 6.7.

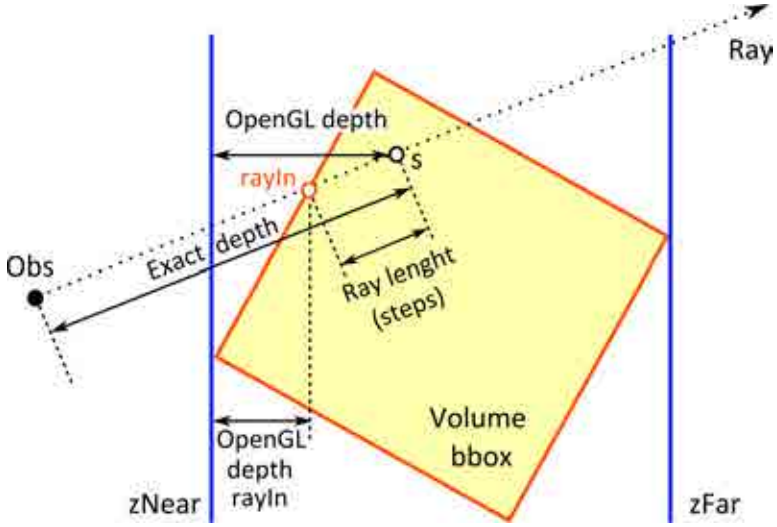


Figure 6.6: *Depth computation methods. The first approach uses the OpenGL depth. The second one computes the number of steps along the ray from the bounding box of the volume. The third method, the one that yielded the best results, combines the two previous approaches.*

6.2.2 Single vs double pass ray casting

In order to obtain the DeMIP result, a single ray casting pass computes the MIP image and the depth map used to modify the shading. This process can be performed in this way because transfer functions used in MIP are quite simple, normally defined using standard window/level information that assigns a single material with different opacities to the density values of the data set. When working with more complex transfer functions (i.e. different materials assigned to different ranges of density values), apart from considering similar materials to compute the closest sample, the first occurrence of each material must be stored while sampling the ray and, when the ray casting finishes, compute the depth of the closest sample with the same material as the maximum intensity value. Another possibility would be to

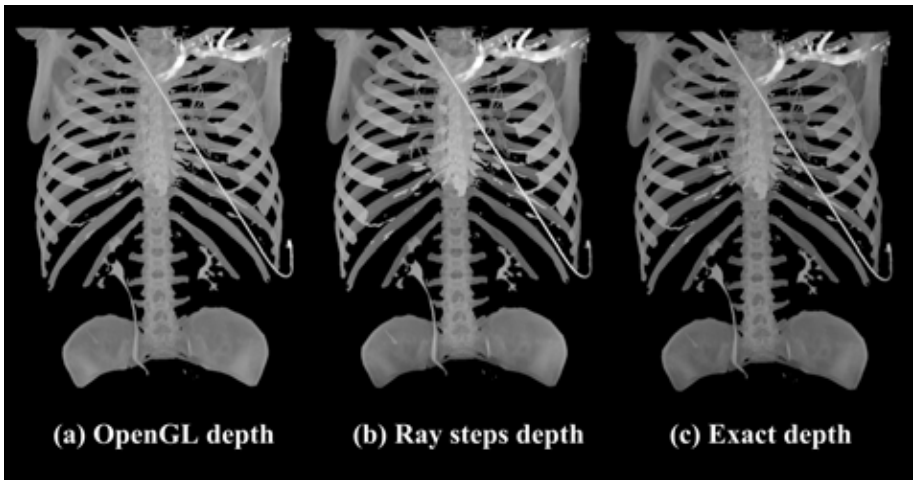


Figure 6.7: *DeMIP results computing depth information in three different ways: with respect to de zNear plane as in OpenGL (a), computing the number of ray steps inside the bounding box of the volume (b) and the exact distance from the observer to the closest sample of each ray (c).*

trace a second ray once the MIP image has been computed, and sample the volume until the first occurrence of the same material is found. This version of the algorithm was the one developed initially [28].

6.3 Results

The proposed approach was tested with several volume models of up to 512 x 512 x 512 voxels on a computer equipped with an Intel Core 2 Duo CPU E8400 @ 3.0GHz, 4 GB of RAM and a NVidia GeForce GTX 280 GPU with 1GB of RAM memory. Although there are several ways to accelerate MIP [77] and, as a consequence, DeMIP, the classical GPU ray casting-based version was used, because we believe this might be an adequate frame rate reference for most of the readers.

A comparison of the timings obtained with different visualization methods and volume data sets is shown in Table 6.1. In all the cases, timings were taken with a sampling rate of 1 sample per voxel on a viewport of 512 x 512 pixels. As it is shown in the table, MIP is faster than DeMIP because there is neither depth nor color cue computations in the original MIP algorithm. Furthermore, both techniques are generally faster than DVR (GPU ray casting and Phong shading) because there is no need of computing illumination effects at each sample of the rays. Maximum Intensity Difference Accumulation (MIDA) [14] timings are also provided for comparison purposes. This is due to the fact that the visualizations obtained with MIDA

Volume data set	Volume resolution	MIP	DeMIP	MIDA	DVR
Jaw	$512^2 \times 40$	64.7	53.8	48.1	49.8
Aneurysm	$512^2 \times 120$	76.2	54.5	43.9	45.8
Abdomen	$512^2 \times 171$	59.4	51.8	50.1	53.4
Head	$512^2 \times 485$	28.9	25.8	17.3	17.6
Body	$512^2 \times 512$	28.1	24.1	26.6	27.1

Table 6.1: *Comparison of frame rates between MIP and DeMIP. Timings for MIDA [14] and DVR are also provided for reference purposes. Timings are measured in frames per second and the sampling rate is one sample per voxel, as more samples do not visually affect image quality in MIP or DeMIP.*

lie in between DVR and MIP, and DeMIP images lie in between MIDA and MIP. Therefore, by providing the frame rates of the four approaches, the reader may appreciate the impact on the performance of different DVR and MIP-based methods.

DeMIP with color cues presents similarities with the pseudo-chromadepth approach proposed in [89], but the motivation is different. In pseudo-chromadepth, the objective is to add a color that helps to perceive distances. The DeMIP purpose is to reinforce overall spatial distribution perception, and provide a visual reference that may be manipulated by the user. Hence, if the user moves the sphere, the color applied to the visualization changes accordingly. In this way, by rotating the auxiliary sphere, users can get further cues to infer the spatial relationships of the visible structures. User can also change the colors mapped in the sphere in order to select the ones that they find more comfortable. This may overcome possible perceptual issues such as color blindness. We believe this is a powerful tool thanks to its flexibility and because there is no need of rotating the volume model. In Figure 6.8 we compare our sphere-based color mapping, pseudo-chromadepth applied to DeMIP, and the original MIP dyed with the pseudo-chromadepth color scale. Note how the occlusions revealed by the depth-enhancement are not visible in this last case.

6.4 User study

In order to evaluate the accuracy of DeMIP for enhancing depth perception in MIP images and stereoscopic visualizations, we carried out an informal user study among a set of 12 people, where most of them were computer scientists or students with computer graphics background. The study con-

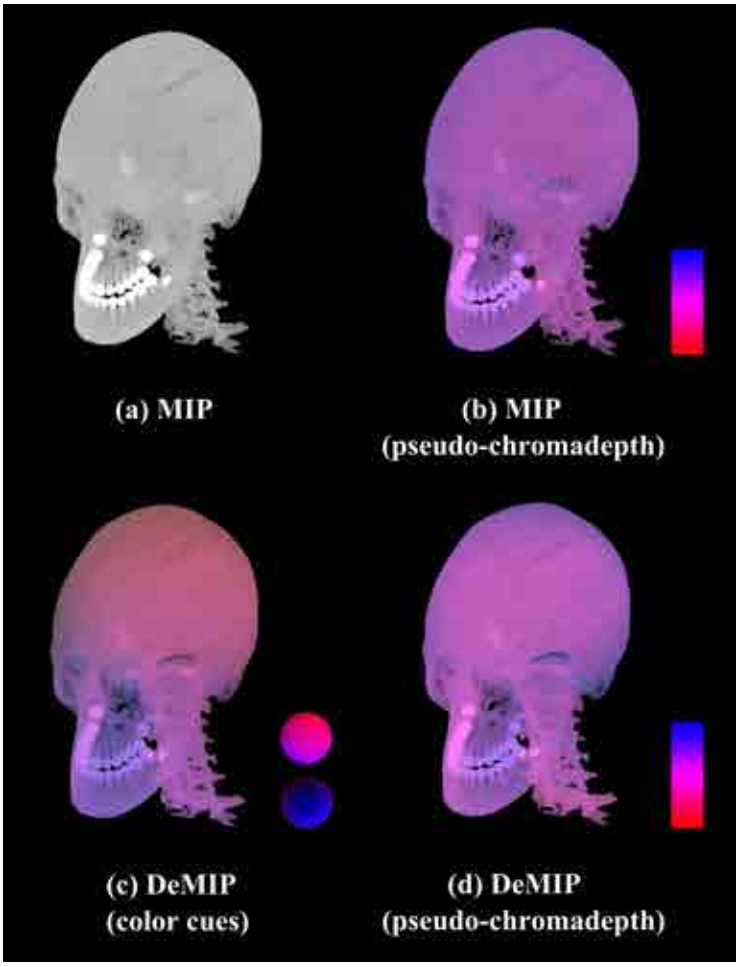


Figure 6.8: *Color mapping strategies: DeMIP with color cues (c) and DeMIP with pseudo-chromadepth (d) show little differences, but the occlusion emphasis obtained by the depth-enhancement becomes notorious if we compare (c) or (d) against a MIP rendering enhanced by applying the pseudo-chromadepth color scale, that is shown in (b).*

sisted on 6 different perceptual scenarios: 3 of them with mono images and the other 3 with stereo renderings.

The main goal of the 3 mono scenarios was testing the effectiveness of DeMIP in those cases where the chosen point of view generates ambiguous MIP renderings. To this end, users were asked to determine the orientation of the body model, the head model and the location of a missing tooth in a jaw model (see Figure 6.9). For these three cases, a set of images was generated using MIP, different versions of DeMIP, varying the way to estimate depth information (see Section 6.2.1), and DeMIP with color cues (see Figure 6.10). Images were shown randomly for each scenario in order to avoid a learning effect.

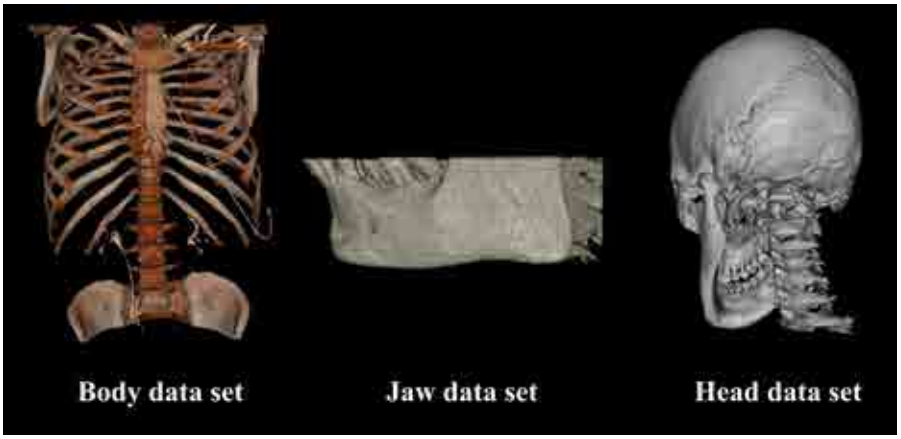


Figure 6.9: *Volume models used in the user study generated with Direct Volume Rendering.*

In the case of the body model, results were clearly favorable to the DeMIP images, as it is shown in Figure 6.11. With the MIP version, the majority of the users answered wrongly, while all the DeMIP images correctly disambiguated the orientation of the model. In the case of the head (see Figure 6.12), MIP images did not provide any clue to determine if the head was facing forwards or backwards. Although DeMIP images without color cues improved the visualization, only DeMIP with color completely disambiguates the orientation of the model. The missing tooth problem was an example where users did not know if there was a hole in the MIP image (Figure 6.10 (a) bottom). On the contrary, all DeMIP images reduce ambiguity and the best results were obtained again by DeMIP with color cues. The answers of the users for this case are shown in Figure 6.13.

The other 3 perceptual scenarios were designed to evaluate the suitability of DeMIP to stereo vision. In this case, a set of stereoscopic renderings was generated using DVR, MIP, DeMIP and DeMIP with color cues, for the body, the head and the jaw datasets. Actually, the displayed images were

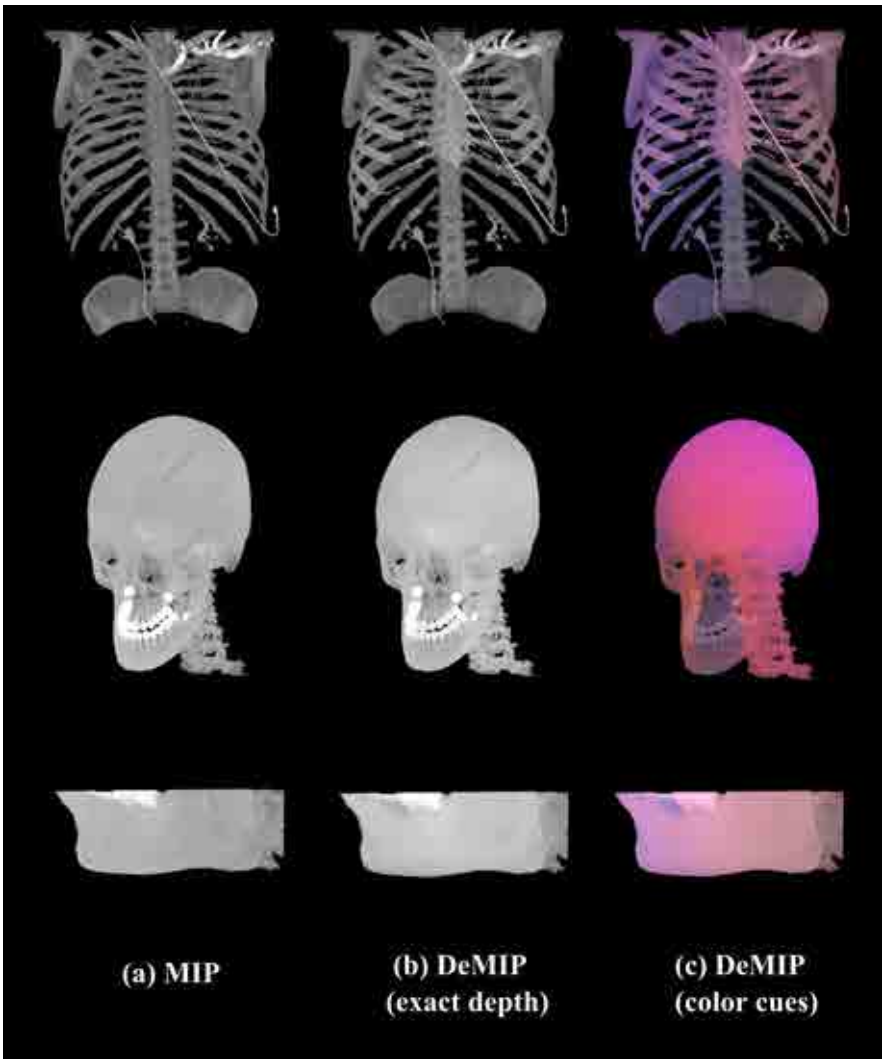


Figure 6.10: *Subset of images shown in the user study. Besides DeMIP computing the exact depth, DeMIP images using the OpenGL depth and the ray steps depth were also shown. The stereo scenarios were analized with the stereo versions of these images.*

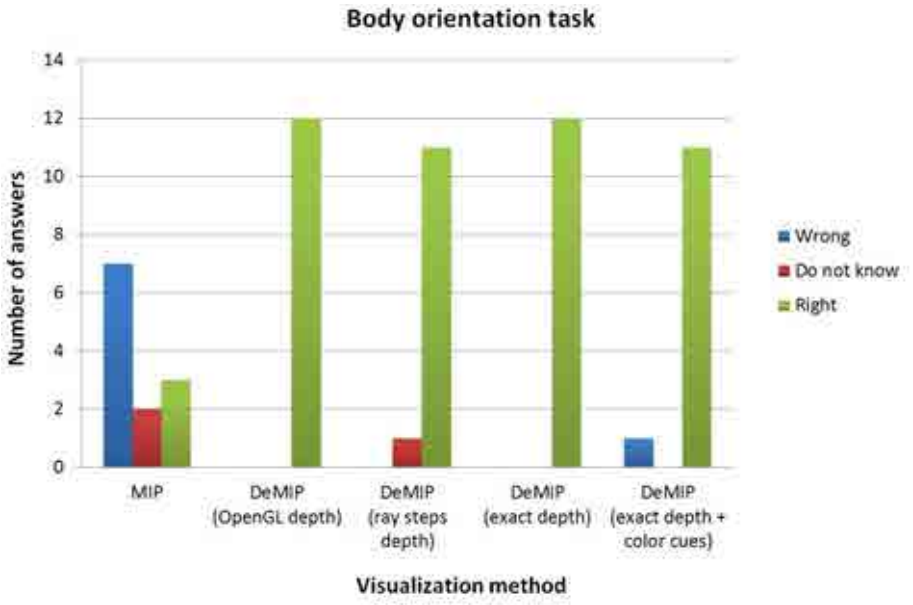


Figure 6.11: *Evaluation of the body orientation. As we can see, users answer correctly in almost all the cases with the DeMIP versions, whereas the orientation was ambiguous in MIP.*

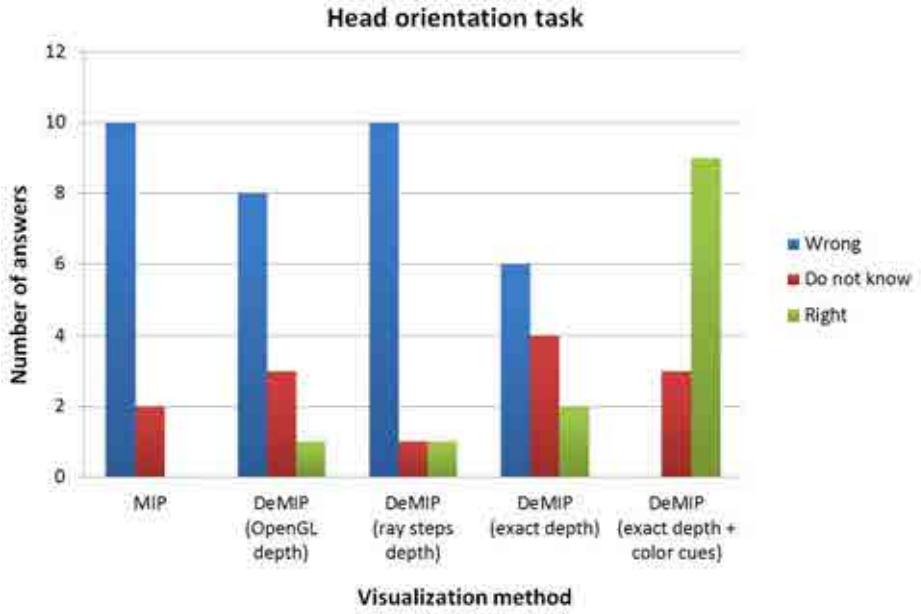


Figure 6.12: *Evaluation of the head orientation. While the DeMIP method with exact depth produces better results than MIP rendering, still most of the users do not correctly interpret the orientation of the skull. The combination of DeMIP with Color Sphere clearly disambiguates the view.*

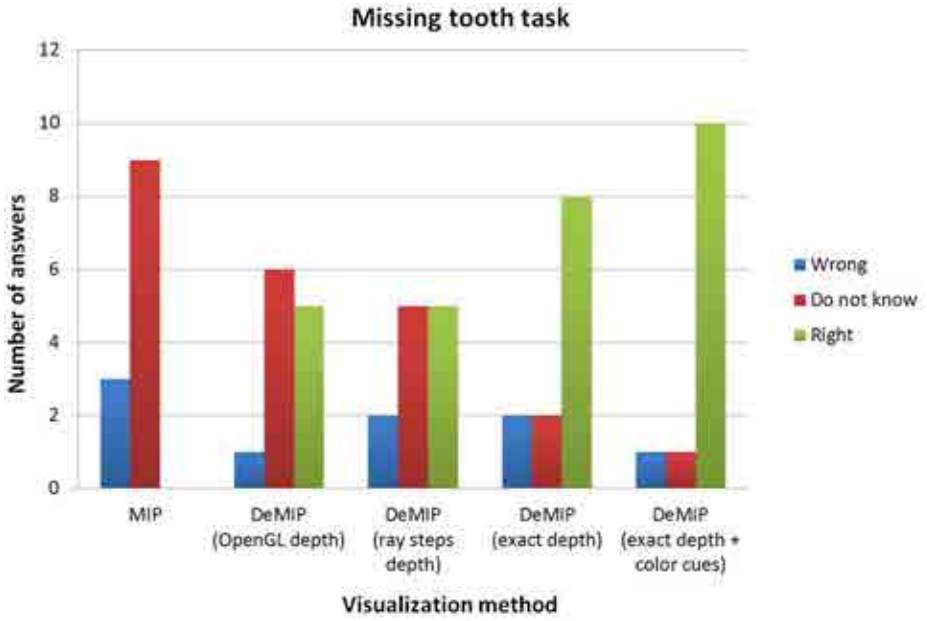


Figure 6.13: *Finding the missing tooth. In this example, the users clearly see that the tooth missing belongs to the left part of the jaw when using DeMIP or Color Sphere methods, while MIP rendering is completely ambiguous.*

the stereoscopic version of the ones shown in Figure 6.10. For these scenarios, DVR visualizations were always shown in the first place as reference images and the other ones were shown randomly. Images were displayed in an active stereoscopic monitor with shutter glasses and users were asked to evaluate in a seven point Likert scale (1 strongly disagree, 7 strongly agree) whether the different stereo images helped to perceive the same spatial distribution of the structures as in DVR.

We analyzed the results of the 3 stereoscopic cases estimating 95% confidence intervals for the means of the answers with the ANOVA analysis, and we may state with our test set that with a confidence level of 95%, one may expect that the answer will be between the limits shown in Table 6.2. Thus, we can conclude that DeMIP and specially DeMIP with color cues are better suited than MIP to stereo vision.

At the end of the study, users were asked to give their opinion about which techniques were better to perceive the overall distribution of the structures of the data set. Furthermore, we asked them to choose between mono and stereo vision and if there was any loss of information in the enhanced images with respect to MIP. Answers showed that more than 90% of users preferred DeMIP with color cues and 60% also found useful DeMIP without color. In the case of the stereo vision, there was unanimity in preferring

95%	MIP	DeMIP	DeMIP (color cues)
Lower bound	2.33	3.66	4.99
Upper bound	3.50	4.83	6.16

Table 6.2: 95% *confidence intervals for the stereo vision tasks.*

it instead of mono images. Finally, 60% of users considered that DeMIP preserves the MIP information while a 40% thought that there is a little loss. Although the results state that DeMIP clearly enhances depth perception in Maximum Intensity Projection, a more detailed user study with the participation of radiologists would be necessary to validate the proposed technique for the medical practice.

6.5 Conclusions

This chapter has proposed a new technique to improve the depth perception [28] in Maximum Intensity Projection. To this end, two different visual cues have been added to the visualization: a depth-based modification of the displayed intensity values and a spatial-based coloring of the volume. In the former case, the maximum intensity projected on each pixel is slightly shaded according to the depth of the closest sample belonging to the structure with higher intensity along the viewing ray. In the latter, the 3D coordinates of the closest sample are used to change the color by using a supporting spherical map. In both cases, samples with the same material as the maximum intensity values are considered to belong to the same structure.

The results of an informal user study demonstrate the effectiveness of the proposed technique to improve MIP images. Furthermore, the enhancements applied allow the user to use stereoscopic devices to visualize MIP renderings. The depth-based intensity change obtains visually similar results than previous enhancement approaches [38] without the need of extracting isosurfaces, and using directly the same transfer function used in MIP. A comparison of the certain observations on DeMIP and other MIP-based methods is shown in Table 6.3. Among all of the compared techniques, DeMIP is the one that preserves most of the information provided by the original MIP, with no need of defining complex transfer functions such as the ones used in *Direct Volume Rendering*.

Vis. method	Most suitable for	Observations
MIP	Contrasted data sets	Contextual information is lost
DSMIP [38]	Contrasted data sets	Preserves contextual information Requires isosurface extraction Polygonal rendering
LMIP [94]	Contrasted data sets	Preserves contextual information May occlude MIP information Simple transfer function definition
DeMIP [28]	Contrasted data sets	Preserves contextual information Most of MIP information preserved Simple transfer function definition Slower than MIP Faster than DVR
SEMIP [121]	Contrasted data sets	Preserves contextual information May occlude MIP information Simple transfer function definition
MIDA [14]	General data sets	Preserves contextual information Occludes MIP information Simple transfer function definition
DVR	General data sets	Provides contextual information Intensities are not displayed Complex transfer function definition

Table 6.3: *Observations on different methods ranging from MIP to DVR. The main advantage of DeMIP with respect to the others is that there is a minor loss of the information provided by the original MIP.*

7

Adaptive cross-sections of anatomical models

The visualization of inner structures is an important topic of research in the volume visualization field. For instance, when working with medical datasets, it is important to clearly perceive the 3D relationships of the internal structures and to be able to analyze certain elements without occlusion, since the lack of accuracy or a misinterpretation of the data might lead to serious mistakes while diagnosing or planning surgery.

Several methods have been proposed to address this issue in the past. Some of them are based on clipping the volume using planes or more complex geometries [114], and are commonly included in the majority of volume renderers because they are simple and easy to use. Unfortunately, contextual information, which is useful to better perceive the shape, position and orientation of the clipped structures, is often removed, as it is shown in Figure 7.1 (a). Note that the use of a clipping plane allows the visualization of the inner parts, but internal elements are equally clipped. As a consequence, it is difficult to imagine how the clipped structures are placed inside the body. In order to address the loss of contextual information, more sophisticated approaches visualize simultaneously external and internal structures using traditional illustration effects, such as cutaways [108] or exploded views [11]. Unluckily, most of these methods do not allow a simple and intuitive interaction with the data, so a certain level of expertise is frequently required.

This chapter presents a new method that extends the basic clipping techniques, visualizing the inner structures of volumetric data sets without removing contextual information and preserving the ease of use. Based on traditional hand-drawn illustrations, our approach provides a direct manipulation tool that allows the user to cut the volume and extrude anatomical structures from the cross-section of the cut. This improves the perception of their shape, position and orientation, as it is shown in Figure 7.1 (b). In this example, the ribs, the heart and other elements emerge from the cross-section, which add valuable information of the anatomical structures around the cut. In order to guarantee its usability, the proposed method is based on mouse dragging movements instead of relying on setting several parameters,

(a) Plain cross-section (b) Adaptive cross-section

Figure 7.1: *Comparison between the classical clipping plane technique and the result of our new approach. Note how different structures are equally clipped in (a), whereas some contextual information is preserved in (b) by extruding certain structures of the volume.*

that often makes it difficult to obtain the desired result. Furthermore, a set of illustrative motifs is also provided in order to give a traditional illustrative look to the final rendering. Besides the fact that the generated images show the effectiveness of the proposed technique, the results of an informal user study demonstrate that the manipulation tool is well-suited for creating such visualizations easily even for inexperienced users.

The rest of the chapter is organized as follows: the next section introduces our approach and describes the interaction process. Section 7.2 describes the application architecture and the results are presented in Section 7.3. The evaluation of the proposed technique by means of the results obtained with an informal user study is described in Section 7.4. At the end of the chapter, conclusions are presented in Section 7.5.

7.1 Illustrative editable cuts

Our main goal when designing the presented technique was the development of a simple manipulation tool to explore the internal parts of a volume model in an easy way. As a result, our method allows the user to cut the volume with a clipping plane and extrude the visible structures out from the cross-section of the cut. In this way, users may interactively specify what portion of each visible structure needs to be extruded, and therefore, they decide which contextual information is worth preserving to get a better

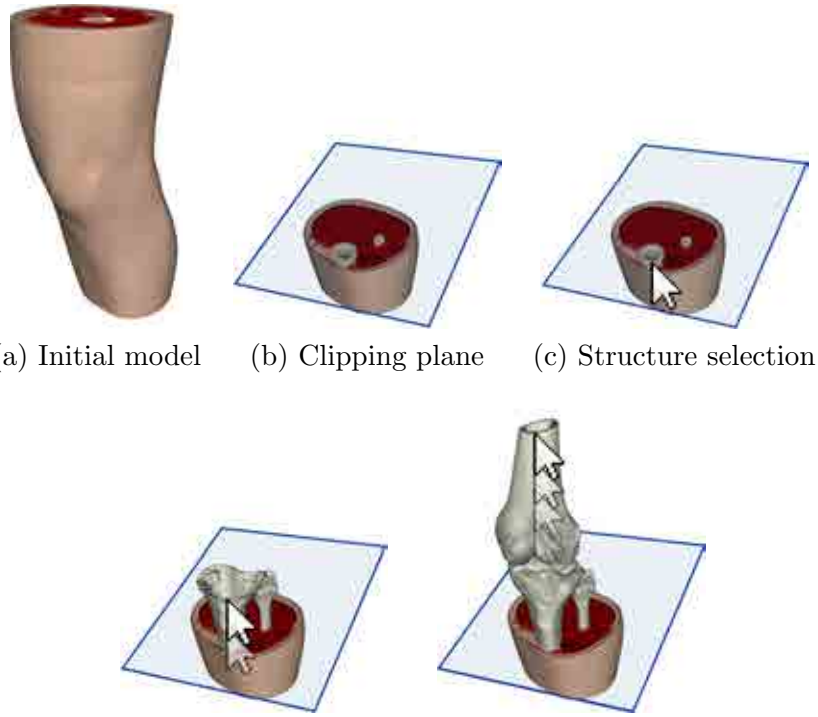
understanding of the clipped structures.

In order to preserve the ease of use of the basic clipping techniques, the interaction with the model is based on the direct manipulation of the structures of interest via mouse movements, instead of interacting with several supporting planes or more complex bounding geometries for each structure. Direct manipulation has other advantages, like the direct coupling between mouse movements and actual model modification, which provides a more controllable way of editing the volumes. Furthermore, we also wanted to work with raw datasets, and not just with pre-segmented volumes, in order to deal with a wider range of models. The only condition was to make the process transparent to the user, that is achieved by making the application able to deal with segmented and non-segmented structures in the same way and maintaining the interactivity. To this end, an automatic segmentation process based on the region growing algorithm is triggered when needed. This process is described in detail in Section 7.2.2.

7.1.1 Interacting with the volume

The manipulation process to generate an adaptive cross-section of a volume model is depicted in Figure 7.2, and it consists of the following steps: firstly, **the model is visualized** and displayed onto the screen (Figure 7.2 (a)). Then, the user **cuts the volume** by defining a clipping plane whose position and orientation can be modified. This plane divides the volume into two regions: the editing one, that is visually removed and corresponds to the subvolume in the positive half-space of the plane (Figure 7.2 (b)), and the non-clipped region, where the original visualization is preserved. Once the clipping plane has been set, the user may start the editing process. Editing is carried out with a number of clicks and mouse dragging, whose use is familiar and predictable, producing an easy-to-learn process. The structure edition follows this pattern: once **the user clicks on certain pixel** p_c (Figure 7.2 (c)), the closest non-transparent structure (S) projected onto it is selected. Then, mouse dragging **extrudes** S at **a distance proportional to the mouse movement** (Figure 7.2 (d)). Once the editing ends, users may also add, if necessary, illustrative effects tailored to giving the image a traditional illustrated look.

In order to appropriately adapt the extrusion shape to the selected structure, we must identify the conditions that a given sample of the volume s, must satisfy to be considered part of the extrusion (E). Let $V \subset \mathbb{R}^3$ be a volumetric dataset, let v_c be the voxel that has been implicitly selected by the user (the voxel visible in pixel p_c) and let Π be the defined clipping plane. Function $pos()$ returns the 3D coordinates of a sample. Then, s will be part of the extrusion, if and only if:



(a) Initial model (b) Clipping plane (c) Structure selection

(d) Structure expansion (e) Final position

Figure 7.2: *Steps of the editing process: the inital model is shown in* (a)*. Then, the user defines a clipping plane* (b)*. Afterwards, the structure to expand can be selected* (c) *and extended* (d) *up to the desired position* (e)*.*

Prop. 1 $0 < \Pi(pos(s)) \leq dist,$

and

Prop. 2 $Mat(v_c) = Mat(v_s),$

where v_s is the voxel that binds s, $dist$ depends on $\| \overrightarrow{m} \|$, where $\overrightarrow{m}$ is the mouse movement vector on the screen, and $Mat()$ is a function that returns the identifier (id) of the material (structure) of the voxel. Note that, for segmented volumes, $Mat()$ only requires querying a 3D structure that stores the identifiers of the structure per voxel.

For the sake of completeness, as it has been specified previously, we want to deal with non-segmented volume models. To this end, the system incorporates an on-the-fly segmentation process, that is achieved by starting a region growing algorithm with a single seed and detecting a single region with an expansion criterion determined by the homogeneity of materials. Although this process can be implemented in different ways, we simply check

that the density value of the candidate voxel lies inside the range of densities previously determined for each material or structure. Formally, this implies a third condition required to consider s as a sample that belongs to the extrusion E:

Prop. 3 *There exists a path σ of face-connected voxels v_i between v_s and v_c* $\mid \{ \forall v_i, Mat(v_i) = Mat(v_c) \wedge 0 < \Pi(pos(v_i)) \leq dist \}$.

It is important to notice that the connectivity constraint imposed by **Prop. 3** is not necessarily satisfied for initially segmented volumes because they may use unique identifiers for the same material (i.e. *bone* might encompass all metacarps in the hand). In this case, non-connected structures would be extruded together if they have the same *id*. Obviously, this can be overcome by ensuring connectivity between voxels on the positive side of the cross-section, although this will be more costly.

7.2 Application architecture

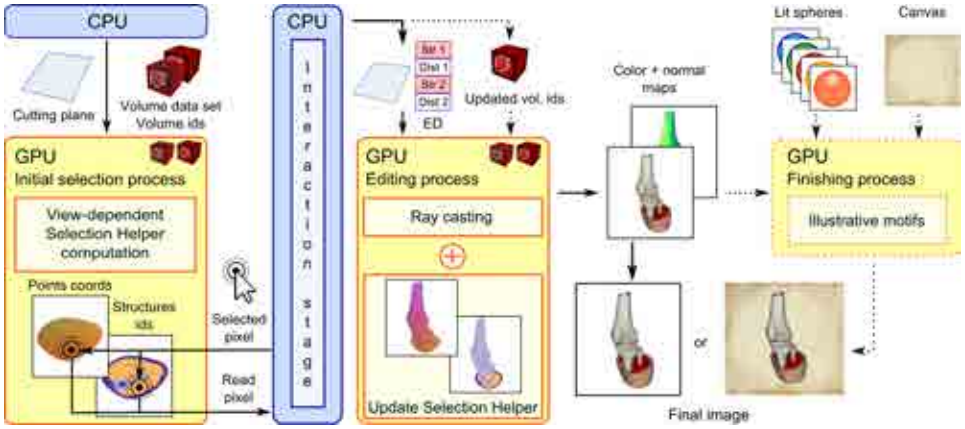


Figure 7.3: *Basic steps performed by a simple structure extrusion. Once the clipping plane has been set, users can proceed and select a pixel on it (left). The CPU identifies the structure projected on it by reading the information from the Selection Texture. Then, the mouse dragging operation modifies the set of visible samples that belong to the selected structure (center). When the editing has been finished, the user may give it the desired finishing by adding some illustration motifs (right).*

As it has been described in the previous section, the selection and extrusion of structures is carried out once the clipping plane has been set. This section presents the architecture of the application by exemplifying a

concrete interaction: a simple extrusion that is performed defining a clipping plane, selecting and extruding one single structure, and giving the final rendering a certain illustrative finish. The data and interaction flows are depicted in Figure 7.3.

7.2.1 Structure manipulation overview

Without loss of generality, we first describe the different processes assuming that the input is a segmented model. Therefore, when interaction starts, the volume (V), the transfer function (TF), and the volume segmentation data structure (VS), which is a 3D texture of *ids* that maps each voxel to its corresponding anatomical structure, are already loaded on the GPU.

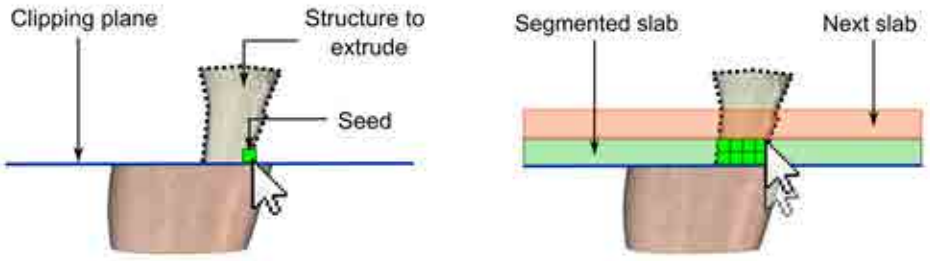
Our application has two major blocks: interaction and rendering. The interaction control is managed by the CPU, while the rendering is implemented as a multi-pass algorithm on the GPU. Although there are several possibilities for implementing each step, we have designed our algorithm in order to guarantee transparency and a smooth interaction through the whole editing process. Taking this objective into account, we use two data structures that contain auxiliary information: the *Selection Helper* and the *Extrusion Distances*.

The *Selection Helper* (SH) is a table whose resolution matches the one of the viewport, that makes the selection of structures very efficient. It stores, for each pixel of the viewport, the 3D coordinates of the closest non-transparent sample projected on it, and the identifier of the structure to whom the sample belongs. The SH is implemented as a couple of 2D textures that reside in the GPU.

After the plane definition, the **selection** stage begins. First of all, the SH is initialized with the information of the cross-section using ray casting. Hence, the SH is view dependent. This process is called *Initial Selection* in Figure 7.3. When the user selects a structure (clicks with the mouse), the **interaction** process reads the clicked pixel and retrieves the information of the selected structure from SH. Then, the user extrudes the structure using a simple *drag-and-drop* movement controlled by a CPU thread.

The *Extrusion Distances* (ED) is a 256-element table that stores, for each selected structure, the distance it has been extruded to from the clipping plane. For non-extruded structures this value is zero. ED is stored as a 1D texture that is incrementally updated according to the *drag-and-drop* movements.

The **editing** process performs a GPU-based ray casting that uses the Phong model to compute the shading of the samples lying on the non-clipped region of the volume. For the samples on the editing region, Phong shading is also used, but it must be checked if they belong to an extruded structure, in order to visualize them. This can be simply detected by querying the volume segmentation texture (VS) and checking in ED if the identifier has



(a) (b)

Figure 7.4: *The incremental segmentation process. In this case, when the user selects the bone, a seed is placed in the selected voxel (a) and an automatic region growing algorithm segments the structure in the first slab of the editing region. Then, the next slab (b) will be segmented if the user continues extruding the bone.*

been extruded. Samples that do not belong to an extruded structure are rendered transparent. In the case that a sample belongs to an extruded structure, it must be determined if the sample is closer to the plane than its corresponding extrusion distance. In this way, the extrusion process detects and properly shades samples that fulfill properties 1 and 2 defined in Section 7.1. When the extrusion ends, users can proceed to another structure. Modified structures can also be edited again if desired.

Once the extrusion of the selected structure ends, some **post-processing** effects can be added. In order to do this, the editing process generates a color image and a normal map that are used as an input information to the Finishing process (see Figure 7.3), which provides a set of effects that include silhouette rendering, *lit sphere* shading [99] and background texturing. Although this post-processing stage can also be applied during the editing process, it seems of little use for the interactive editing step.

7.2.2 On-demand segmentation process

We considered the possibility of working with non-segmented volumes while keeping the same user interaction metaphor useful. In this context, we decided to incorporate an interactive segmentation process based on the region growing algorithm [90], that is triggered when the original data set has not been previously segmented. The seed propagation of the region growing fulfills properties 1, 2 and 3 described in Section 7.1.

The segmentation process involves different steps. Due to the fact that the volume is non-segmented, the *Selection Helper* (SH) initially contains the 3D coordinates of the samples at the cross-section. The *ids* of their associated structures are set to zero. When clicking on a certain pixel of the cross-section, the voxel that contains the sample obtained from the SH is

selected. Then, the first free structure *id* is associated to the new structure to segment, and a seed is placed in the selected voxel. After that, the region growing starts by propagating the seed through all the face-connected voxels that belong to the same density range (i. e. structure) than the selected voxel. In our implementation, the range of densities for a certain structure depends on the ranges defined by the transfer function. Thus, we guarantee that structures with different optical properties are correctly segmented. As the process continues, the face-connected voxels are added to the new segmented structure if and only if they are in the same density range as the selected voxel and they have not been previously segmented. The process iterates until there is no change in the two successive steps.

In order to guarantee interactivity, the segmentation is carried out on demand in an incremental way. Instead of segmenting the whole structure at once, the positive half space region defined by the clipping plane (i. e. the editing region of the volume) is divided into slabs (see Figure 7.4), and the segmentation is carried out slab by slab until the extrusion distance is reached. Because segmenting a slab is faster than segmenting a whole structure, the segmentation can be incrementally done while the user manipulates the volume, preserving the interactivity in most of the cases (see Section 7.3). These slabs are generated by taking the clipping plane and moving it a certain offset (twice the voxel diagonal by default) in the direction of the plane's normal. The only information needed for the next step is the active front of the current segmentation, in order to know which slab has to be segmented afterwards. During the process, the volume segmentation texture (VS) is continuously updated and uploaded to the GPU each time it is modified. The main advantage of the proposed segmentation process is that the rendering algorithm is the same for raw and segmented data sets. Therefore, it concurrently supports a combination of segmented and non-segmented regions.

Complex structures may require additional constraints for an accurate seed propagation, such as gradient computations. In these cases, when the user drags the mouse fast, the interaction may suffer and might become non-interactive. On the contrary, for structures with fuzzy boundaries or similar density values, a previous accurate full segmentation is desirable [18].

From the users' point of view, nothing special has to be done for non-segmented models, since the segmentation is triggered automatically and transparently if required.

7.2.3 Interaction design

When designing the application interaction, the following requirements were considered: firstly, the user had to be able to define a plane which could be easily rotated and translated. Secondly, we had to provide a way to select a structure from the cross-section of the cut, which is done by clicking on a

pixel where the structure is projected. Thirdly, the extrusion of the selected structure had to be carried out in an easy way, so we decided to base the process on mouse movements to adjust the portion of the structure to be shown. Finally, in order to deal with cuts with more complex shapes, we decided to include a second clipping plane.

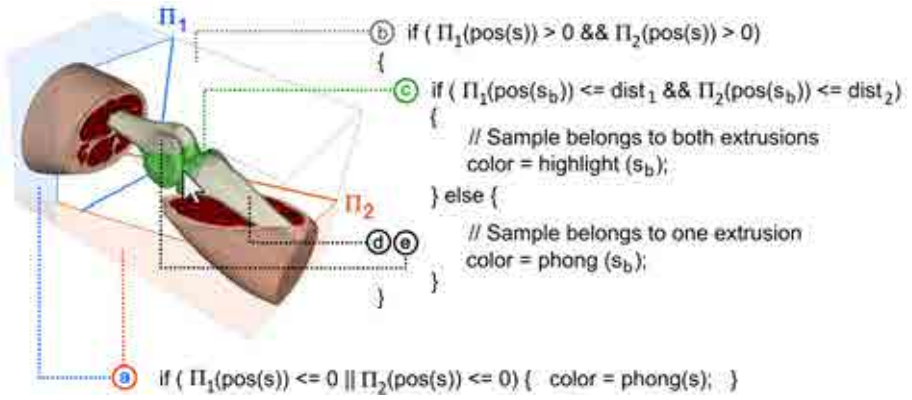


Figure 7.5: *Structure extrusion from two different planes (Π_1 and Π_2) and the basic algorithm to provide a helpful visual feedback while manipulating the volume. The non-clipped region (a) is the union of the negative half-spaces of the clipping planes. The editing region (b) contains all the samples (s) of the volume within the intersection of the positive half-spaces of the planes. Depending on the extrusion distances of the bone ($dist_1$ and $dist_2$) from both of the planes, the samples (s_b) of the extruded structure belong to a certain region (c, d, or e) and are coloured accordingly. Extrusions intersection (c) is highlighted in green. $\Pi_i(pos(s))$ is a function to determine the signed distance from a sample (s) to a plane (Π_i).*

Taking the previous requirements into account, we designed the interaction process with two main objectives: flexibility and simplicity. In order to ensure simplicity, all processes are supported by a visual feedback, as it is illustrated in Figure 7.5. For example, when editing the model, the bounding box of the volume is drawn in wireframe and the cutting planes are also shown. Furthermore, the editing and the non-clipped regions of the volume are illustrated by giving a pale transparent look to the negative parts. When a structure is selected, its supporting plane is highlighted by doubling its line size. Thus, a visual cue to indicate the selection of a certain element is provided, and it also shows which plane a structure has been extruded from. If the extrusions from two different planes overlap, the intersection is rendered with a different color. When selecting a point in this intersecting region, both planes can be potentially selected, and the user decides the proper one by toggling between them using the space bar. Finally, the sup-

(a) (b) (c)

Figure 7.6: *Illustrative motifs provided by our application. View-dependent contours computed with a Sobel filter are shown in (a). The result of applying the lit sphere shading and the spheres used are shown in (b). The combination of the two previous effects over a canvas texture is depicted in (c). The lit spheres were taken from [13].*

porting planes can be modified at any moment, and the extruded structures change accordingly, thanks to the fact that the *Extrusion Distances* table (ED) stores information that is independent from the plane position. Figure 7.5 also shows the different cases that the shader must deal with. For instance, all the samples belonging to the non-clipped region must be rendered according to the optical properties provided by the transfer function and the Phong shading computation. On the contrary, only the samples belonging to extruded structures must be rendered in the editing region. As it is described in Section 7.4, the results of the user study show that the proposed technique requires little training time, in part thanks to the visual clues provided, as it was stated by most of the users involved in the study.

7.2.4 Illustrative finish

Besides the flexible geometry editing, the presented technique also provides a set of traditional illustration motifs to easily tune the final look of the resulting images. Concretely, view-dependent contours, the lit sphere shading and the addition of a canvas (i.e a background texture), can be applied taking as a basis the color and the normal maps generated in the editing process (see Figure 7.3).

Professional illustrators often emphasize the contours of the objects to improve the perception of shapes. In computer graphics, different image processing operators can be used in order to extract the edges of a source image, such as the *Sobel filter*. When working with complex datasets with many fine and overlapping structures, extracting all the contours of the generated visualization may produce visual clutter. In order to avoid this and

taking into account that humans are more sensitive to luminance changes [7], we have decided to apply the filter just to the luminance channel of the image, extracting thus the most perceptible contours. Therefore, the post-processing step of the algorithm transforms the RGB color to CIELab before applying the Sobel kernel. Figure 7.6 (a) shows the contours extracted from the hand model.

Another illustrative motif available is the *lit sphere shading* [99]. This method is designed to capture custom artistic shading models from sampled artwork. Its main idea is to use a reference sphere that contains the shading of a certain material and retrieve the color from the sphere using the normal vector or the gradient of the point to shade. Figure 7.6 (b) shows an example of using different lit spheres for shading the skin, the bones and the muscles of the hand model. In this case, the color of the sphere is blended with the color obtained from the ray casting process, assigning the same weight to each contribution. As a result, the color of a given pixel $[x, y]$ is computed as follows:

$$Color[x, y] = (0.5 * Phong[x, y] + 0.5 * LS(Normal[x, y])) * int_{contour} \quad (7.1)$$

where $Phong_[x, y]$ is obtained from the color map generated in the ray casting process, $LS(Normal[x, y])$ is the color of the lit sphere texture, which is obtained by using $Normal[x, y]$ from the normal map of the visualization, and $int_{contour}$ is a weight assigned to the edges as detected by the Sobel filter.

Finally, the presented method offers the possibility of adding a canvas by means of a background texture. The main reason for this effect is that solid backgrounds often produce an artificial look that must be avoided in order to simulate a professional illustration. The result of combining the previous effects is shown in Figure 7.6 (c).

7.2.5 Implementation details

The commitment to guarantee ease of use and interactivity has lead us to take some implementation decisions. First, as shown in Figure 7.3, the *selection helper* (SH) is written in two different stages. This is due to the fact that the first process (initial selection) uses a simple shader that does not account for extruded surfaces. The editing process may update SH at a negligible cost and therefore it is ready for the next selection process. This also avoids an extra rendering pass when selecting another structure.

We have also implemented a basic GPU version of the region growing segmentation process. It iteratively segments a slice of the volume at a time by rendering a quad and checking, for each voxel of the slice, if it belongs to the material of the initial seed (the one placed at the voxel selected by the user), and if it is connected with the current segmented region (at the first

step of the process, the segmented region just contains the voxel with the seed). The segmentation of a slice finishes when no new voxels are added to the segmented region. The limitations that the OpenGL pipeline impose require a *ping-pong* strategy to compute the 3D segmentation texture, as well as wasting quite a lot of resources, since the whole slice is treated each step (restricting the region to analyze would require an extra auxiliary structure to be updated each step). As a result, this approach is less efficient than the CPU algorithm in most cases, since it only processes potentially selectable voxels. Another possibility would be to use the newer Fermi GPUs, that would permit CUDA computation, and making the result available for the OpenGL pipeline. We leave this for future work optimizations.

Instead of the GPU version, an incremental strategy was chosen to include in the system, since we want to prevent the user from waiting for the whole segmentation to finish. To this end, slabs are segmented at once, trying to segment the data in advance of users' moves. If users had to wait a long time, this would probably break their flow of thought and interaction would become uncomfortable. However, the segmentation process may compromise the interactivity if the region to expand is very big. In these cases, the user may notice a slowing in the interaction refresh, as compared to a previously segmented region. In a typical on-the-fly segmentation the frame rate may drop to 4-8 fps. Nevertheless, these frame rates are high enough for the user to perceive continuous response from the system.

7.3 Results

The proposed method was tested with different volume models of up to 512 x 512 x 512 voxels. Timings were taken in a computer equipped with an Intel Core2 DUO CPU running at 3.0 GHz, and a NVidia GeForce GTX 280 GPU with 1GB of RAM memory. The sampling rate was 1 sample per voxel and the size of the viewport was 800 x 600 pixels. Two different scenarios with segmented models are shown in Table 7.1: in the first one (*Edition*) the user continuously extrudes the structures with one or two clipping planes (1Π and 2Π in column 4, respectively). The rightmost column (*Final*) shows the frame rates obtained for an edited volume with contours, a canvas, and the lit sphere shading. As we can appreciate, interactive frame rates are obtained for all the models in the different scenarios, and there is almost no difference between using one or two clipping planes while editing the volume. Furthermore, note that the impact on rendering of adding the additional illustrative motifs is almost negligible.

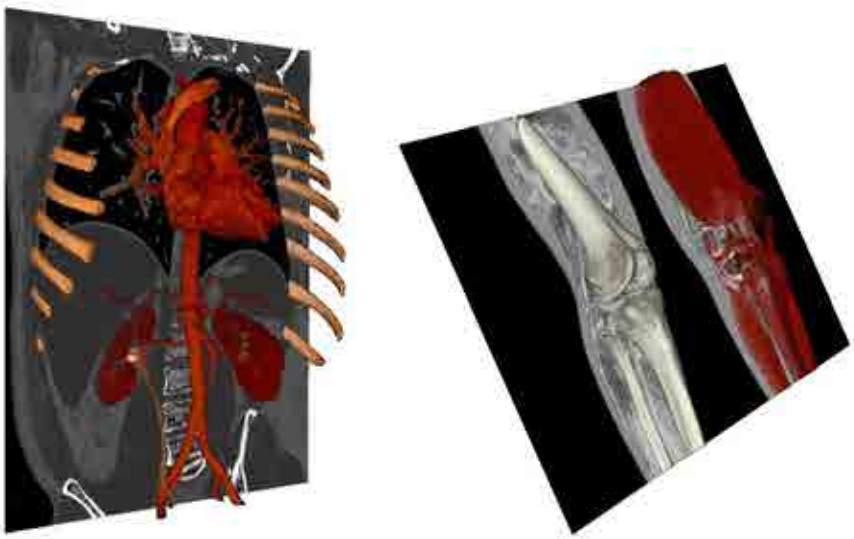
When working with non-segmented data sets, the frame rate obtained is slightly slower. For instance, when segmenting on-the-fly the structures of the *Foot* data set shown in Table 7.1, frame rates of the Edition column may decay to 6-7 fps, depending on the size of the structure to segment. However,

Volume data set	Volume resolution	Phong shading	Adaptive cross-sections Edition 1Π	Edition 2Π	Final 1Π	Final 2Π
Tooth	$256^2 \times 162$	100	69.4	69.4	69.2	66.1
Engine	$256^2 \times 256$	93.6	61.4	61.4	61.4	60.9
Hand	$512^2 \times 170$	48.2	33.6	31.2	32.9	31.0
Foot	$512^2 \times 256$	41.6	30.2	28.5	30.6	29.1
Legs	$512^2 \times 256$	45.0	31.6	28.6	31.3	29.1
Head	$512^2 \times 460$	33.2	23.7	21.9	23.5	21.9
Body	$512^2 \times 512$	16.3	14.2	12.2	14.1	12.2

Table 7.1: *Performance of our system with different segmented models in fps. Third column shows the rendering times of the Phong shading visualization clipped by the plane* 1Π*. The Edition column shows the average frame rates obtained in several minutes of edition (i. e. extruding structures) using one (*1Π*) and two (*2Π*) planes, respectively. The Final column shows average frame rates combining the extruded elements with the finish effects (extrusions, lit sphere, background texture, and contours).*

the penalty on the rendering time may be hardly noticeable with other volume models, since the time required for the segmentation of one slab, in most of the tested cases, is in the range of 0.008 to 0.012 seconds, which does not affect the interactivity of the editing. For brisk moves (that might involve a large region to segment), the frame rate decays, but we still get continuous interaction. In order to test a worst-case scenario, we simulated the selection of air in an empty region: for a large slab (covering 4 slices of 512×512), the region growing detected 958K voxels and it took 0.19 seconds. Normal extrusions usually generate 12K to 13K voxels, thus, the overhead is generally affordable. On the other hand, we may adapt the size of the slab to sudden moves if required, but in most cases a distance of 2 voxel diagonals from the clipping plane is a good compromise and achieves good frame rates. The proposed application also may perform the whole segmentation at once, but in this case, informing the user during the underlying process. After that, the interaction would be in real-time again. However, we did not find the necessity to swap to this mode during our experiments.

For non-segmented models, the accuracy of the extrusion may be limited by the simple region growing algorithm which only considers density information based on the ranges defined by the transfer function. The proposed application architecture can be extended by including more sophisticated methods, such as multi-dimensional transfer functions [50], but to correctly identify the structures with similar density values or fuzzy boundaries, more



(a) (b)

Figure 7.7: *Enhancing contextual information of CT images. The ribs, the heart, the kidneys and the aorta have been extruded from a coronal slice of a human torso in (a). Additional contextual information has been also added to a coronal CT slice of the legs in (b).*

complex segmentation algorithms would be desirable. These algorithms are usually neither fully automatic nor interactive [18, 43, 104], and they cannot easily be adapted to our editing proposal. So, they should be applied as a pre-process and use our interaction tool to manipulate the segmented data set. Nevertheless, thanks to the data structures design, the proposed system is also able to seamlessly work with partially segmented volumes.

Another interesting feature is that with no need of significant modifications, additional contextual information, such as the density values displayed while inspecting medical images, can be incorporated, as it is shown in Figure 7.7. In these examples, a plane intersects the volume and the density values of the intersection are rendered on it. Then, by selecting different voxels on the plane, the user extrudes the structures to correctly identify them and perceive their spatial arrangement.

7.4 User study

Among the techniques reviewed in Chapter 3 to visualize the inner structures of volume models, there are only a small number of methods that allow the direct manipulation of the structures. These systems do not implement structure-aware volume modification, apart from the recent elastic membrane clipping [5], which works in a similar way to our approach, but

User	Learning time	Hand 1Π	Hand 2Π	Foot 1Π	Foot 2Π
Subject 1	7' 30"	34"	40"	50"	1' 33"
Subject 2	7' 09"	21"	17"	36"	1' 22"
Subject 3	10' 05"	57"	38"	4' 10"	3' 26"
Subject 4	9' 10"	18"	14"	30"	1' 09"
Subject 5	8' 02"	1' 37"	1' 29"	1' 05"	2' 41"
Subject 6	8' 07"	36"	40"	1' 14"	3' 15"
Subject 7	7' 15"	20"	18"	16"	54"
Subject 8	9' 45"	45"	32"	42"	1' 26"
Subject 9	7' 59"	35"	46"	26"	1' 12"
Subject 10	7' 17"	18"	22"	15"	1' 19"
Subject 11	9' 48"	16"	40"	35"	1' 15"
Subject 12	10' 28"	36"	2' 53"	36"	3' 21"

Table 7.2: *Time spent by users in the two stages of the user study. Note that learning time was less than 10 minutes in the majority of cases, and the time to replicate the target image was less than 1' and 30" for the most complex target image (the cut of the foot model using 2 clipping planes) in most of the cases.*

requires partial indirect manipulation that may be a time-consuming task. Since the method presented in this chapter works directly on the extruded structures, comparing it with such techniques would not be fair. Thus, in order to validate its usability, we carried out an informal user study among a set of 12 people, who were either computer scientists or computer science students.

The study had two stages: the first one was designed to show the users how the application works, whereas in the second one, they had to replicate some target images. In the first stage, a tutorial video explaining the main functionalities of the system was used. After describing each functionality, users might stop the video and put what they learned into practice. In the same way, when the video finished, users could spend as much time as they wished to get familiar with our tool. The length of the tutorial video was almost 3 minutes and as it can be seen in Table 7.2, the whole learning process took less than 10 minutes for all the users.

The goal of the second stage was to test the usability of the manipulation tool. To this end, we generated 4 images obtained with 1 and 2 clipping planes that users had to replicate (see Figure 7.8). Images were shown randomly in order to avoid a learning effect, and we measured the time to obtain the same visualization as the target images. As it is shown in Table 7.2, the average time to obtain the images of the hand model was 41 seconds,

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

Figure 7.8: *Examples of the test images given to the users. (a) and (c) can be obtained with a single clipping plane, while (b) and (d) require extruding from two planes.*

56 seconds for the foot model with one plane and 114 with two planes.

Although target images were shown randomly, there was a tendency to reduce the times during the experiment. For instance, in the case where the foot model was clipped using two planes, users took much less time to finish the task when it was displayed at the end. On the contrary, subjects 5 and 6 had to solve this case first and the time spent was longer than with the other cases (see Table 7.2). For this reason, we think that a larger sample size should be analyzed in order to raise more significant conclusions. However, we consider that users obtained quite promising results, taking into account that none of them had previous knowledge of the system. In addition to the user study, we also showed our application to medical experts from different fields of internal medicine. They consider that our approach may be successfully used to create images for teaching media, and one of them also said that it might be useful for surgery preparation by using the real data sets of the patients.

7.5 Conclusions

This chapter describes a novel approach [27] to explore the internal parts of volume models, which allows the easy and intuitive creation of anatomical illustrations. The proposed method has been designed to preserve the ease of use of the most basic clipping techniques without removing contextual information. To this end, instead of manipulating several clipping planes or complex bounding geometries to cut the volume, we present a direct manipulation tool based on mouse dragging interaction, that directly extrudes structures from the cross-section of the cut. Furthermore, different illustrative motifs may be applied to the visualization for obtaining a more traditional anatomical illustration look.

In order to deal with raw data sets, the system incorporates an on-the-fly segmentation process based on the region-growing algorithm [90]. The novel feature of the process is the slab-based strategy undertaken to segment the selected structures, which allows the preservation of the interactivity while manipulating the volume. Thanks to the presented application architecture, the user is unaware of when the segmentation process is triggered.

An informal user study demonstrates that the learning process to use the technique is quite low (up to 10 minutes). With this little training time, most of the users were able to reproduce a set of selected target images in times that ranged from several seconds up to a couple of minutes. The whole interaction process is performed in real time and, even with non-segmented data, interactive frame-rates are obtained in most of the tested cases. it is also quite flexible, since it provides the user with the possibility to add extra illustrative effects.

Figure 7.9: *Anatomical illustration created with the proposed technique. Two clipping planes have been used to make visible the brain and the ventricles while preserving part of the skull. Enhanced contours and a background canvas have been added to provide an illustrative look to the visualization.*

8 Conclusions

Due to the nature of volumetric data sets, the generation of convenient visualizations is crucial to correctly convey the relevant information of the data. In order to enhance the perception of different features, the spatial relations among structures and the overall appearance of volume models, this PhD dissertation has presented several approaches, which are mainly based on the modification of the shading model and the simulation of traditional illustration effects.

In Chapter 4, we have described a new method to emphasize local features in real-time, which is based on volumetric unsharp masking and a multi-scale volume hierarchy of the original data set. The main advantages with respect to previous approaches are the low memory consumption, which makes this technique well-suited for relatively big volume models, and the improvement of the perception, not just by brightening salient features, but also by darkening deeper regions or changing the color in a pleasant way. Although the presented technique effectively enhances the perception of local details, there is room for further improvements. Depending on the relative size of the features to emphasize, users manually choose which levels of the multi-scale hierarchy must be used in the enhancing process. Future research may consider the automatic selection of the adequate hierarchy levels and a selective or local application depending on the size of the features of a target region.

In order to enhance the depth perception in *Direct Volume Rendering* (DVR), we have proposed two different approaches in Chapter 5. The first one is a view-dependent method based on 2D summed area tables that modifies the shading by simulating the ambient occlusion. Furthermore, this technique allows the generation of colored halos around the structures of interest, in order to highlight them. The second approach is a view-independent method based on a 3D summed area table of density values, that is also used to simulate ambient occlusion. In this case, occlusion computations are performed in a volumetric way, instead of using image-space information as in the view-dependent approach. The main advantage of both techniques is that, thanks to the use of summed area tables, ambient occlusion can be simulated in a fast way and, as a consequence, the final images

are generated in real time. Despite the fact that both techniques obtain similar quality results, the view-independent method yields better frame rates because of the preprocessing computations, while the view-dependent one has less memory requirements. The main limitation of the view-dependent method is its dependency on the depth buffer. As a consequence, ambient occlusion and halos are just applied to the structures represented in it (in this case, the most opaque ones). Future research must study how to deal with several opacity layers, in order to apply both effects to more general cases. Concerning the view-independent method, ambient occlusion is also applied to rather opaque structures to obtain a good performance. Furthermore, memory consumption may be excessive for relatively big data sets. An interesting line of research is the evaluation of different compression schemes to reduce memory requirements and the global application of ambient occlusion while preserving interactivity.

Whereas the previous techniques improve the depth perception in DVR, Chapter 6 presented a novel approach to enhance the depth perception of *Maximum Intensity Projection* (MIP) images. This new method called *Depth-enhanced Maximum Intensity Projection* (DeMIP), modifies the intensities displayed in the original MIP by using depth information of the structures contained in the volume. Thus, the proposed approach reveals the occlusion and the spatial relationships between the inner elements in real-time, with a minimum loss of information with respect to MIP. Furthermore, spatial information can be also enhanced by adding extra color cues. Although the results of an informal user study show that DeMIP effectively adds contextual information to MIP, the next step would be an accurate validation by medical experts in order to know if DeMIP could be used in routine clinical practice.

Finally, Chapter 7 describes a new interaction method that provides a direct manipulation tool to explore inner structures and create anatomical illustrations easily. The main idea behind this approach is cutting the volume with clipping planes and directly extruding the clipped structures from the cross-section using mouse movements. This results in an easy to use tool that is well-suited for inexperienced users, since the manipulation is not controlled by several indirect parameters and there is no need of dealing with complex bounding geometries. Furthermore, the architecture application makes the interaction fast and the method supports on-demand data segmentation when required. Although the segmentation is carried out using a slab-based strategy to preserve the interactivity, it highly depends on the size of the structures to segment. Therefore, future work may focus on improving the segmentation process, not just for preserving the interactivity while segmenting big structures, but also to obtain more accurate segmentations.

In conclusion, the different techniques presented in this thesis have provided new ways to enhance the visualization of volumetric data sets, not just

by proposing different solutions to overcome the limitations of previous approaches, but also by providing new ideas to develop more recent techniques, such as the *Shape-enhanced Maximum Intensity Projection* by Zhou *et al.* [121], which is based on DeMIP, or the *Extinction-based Shading* by Schlegel *et al.* [96], which uses 3D summed area tables to simulate shadowing effects.

8.1 Future research

Besides effectively improving the perception in the volume visualization field, we consider that the presented methods can be the starting point of future research. The work presented in this thesis has been focused on modifying the shading model to better perceive local features and depth information, as well as the simulation of traditional illustration effects to properly convey the information provided by the data sets.

From the shading perspective, there is room for further research, as it has been shown in Chapter 3. For instance, the simulation of shadows or other global illumination effects is an interesting topic that is evolving rapidly thanks to the new computing capabilities of the GPUs. Another interesting line of research is the evaluation of the existing methods by means of user studies, in order to get better insights on their ability to improve perception and provide new ideas for the development of future approaches. In a similar way as the perceptual study presented by Lindemann and Ropinski in [63], classical illumination models and other existing visualization methods could be tested to know their suitability to other scenarios, such as virtual reality environments or mobile devices.

Illustrative visualization is another important topic of research because it provides the levels of abstraction needed for a better understanding of the data. Focusing on the work presented in this thesis, we believe that summed area tables could be further exploited to generate other illustrative effects. For instance, the combination of different depths of field while visualizing the volume could improve the exploration of the data by guiding the attention of the observer to the features of interest.

The direct manipulation of volumetric information is another interesting topic of research. The final appearance of the resulting visualizations generally depends on different parameters that must be manually adjusted. As a consequence, a certain level of expertise is often required to obtain the desired views of the data. In this thesis, we have proposed a direct manipulation tool that replaces the parametrization by drag-and-drop mouse movements, an interaction metaphor that is well-known by the layman. Further research on the automatic definition of parameters and easy manipulation tools would be interesting, in order to improve the exploration of volumetric information.

8.2 Publications

The contributions presented in this thesis have been published in the following papers:

- Jose Díaz, Héctor Yela and Pere Pau Vázquez. Vicinity Occlusion Maps: Enhanced Depth Perception of Volumetric Models. *Computer Graphics International*, pages 56-63, 2008.
- Jose Díaz, Pere Pau Vázquez, Isabel Navazo and Florent Duguet. Real-time Ambient Occlusion and Halos with Summed Area Tables. *Computers & Graphics*, 34(4):337-350, 2010.
- Jose Díaz and Pere Pau Vázquez. Depth-enhanced Maximum Intensity Projection, *IEEE/EG International Symposium on Volume Graphics*, pages 93-100, 2010.
- Jose Díaz, Jordi Marco and Pere Pau Vázquez. Cost-effective Feature Enhancement for Volume Datasets. *International Workshop on Vision, Modeling and Visualization*, pages 187-194, 2010.
- Jose Díaz, Eva Monclús, Isabel Navazo and Pere Pau Vázquez. Adaptative Cross-sections of Anatomical Models. *Computer Graphics Forum (Proceedings of Pacific Graphics 2012)*, 31(7): 2155-2164, 2012.

Bibliography

[1] L. Bavoil and M. Sainz. Multi-layer dual-resolution screen-space ambient occlusion. In *ACM SIGGRAPH 2009: Talks*, page 45:1, 2009.

[2] L. Bavoil, M. Sainz, and R. Dimitrov. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008: Talks*, pages 22:1–22:1, 2008.

[3] K.M. Beason, J. Grant, D.C. Banks, B. Futch, and Hussaini M.Y. Precomputed illumination for isosurfaces. In *Conference on Visualization and Data Analysis*, pages 1–11, 2006.

[4] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. In *VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 39–46, 1998.

[5] Å. Birkeland, S. Bruckner, A. Brambilla, and I. Viola. Illustrative membrane clipping. *Computer Graphics Forum*, 31(3):905–914, 2012.

[6] Å. Birkeland and I. Viola. View-dependent peel-away visualization for volumetric data. In *Proc. of Spring Conference on Computer Graphics*, pages 121–128, 2009.

[7] H.R. Blackwell. Contrast thresholds of the human eye. *Journal of the Optical Society of America*, 36(11):624–632, 1946.

[8] J.F. Blinn. Models of light reflection for computer synthesized pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, 1977.

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

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

[12] 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.

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

[14] S. Bruckner and M.E. Gröller. Instant volume visualization using maximum intensity difference accumulation. *Computer Graphics Forum*, 28(3):775–782, 2009.

[15] M. Burns and A. Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Trans. Graph.*, 27:1541–1547, 2008.

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

[17] H.J. Chen, F.F. Samavati, and M. Costa Sousa. Gpu-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer*, 24:689–698, 2008.

[18] A. Chica, E. Monclús, P. Brunet, I. Navazo, and A. Vinacua. Example-guided segmentation. *Graphical Models*, pages –, 2012.

[19] P. Cignoni, R. Scopigno, and M. Tarini. A simple normal enhancement technique for interactive non-photorealistic renderings. *Computer & Graphics*, 29(1):125–133, 2005.

[20] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y-Q. Xu. Color harmonization. *ACM Trans. Graph.*, 25(3):624–630, 2006.

[21] R.L. Cook and K.E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, 1982.

[22] C.D. 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.

[23] C.D. Correa, D. Silver, and M. Chen. Technical section: Constrained illustrative volume deformation. *Computers & Graphics*, 34:370–377, 2010.

[24] F.C. Crow. Summed-area tables for texture mapping. In *ACM SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, 1984.

[25] 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.

[26] J. Díaz, J. Marco, and P. Vázquez. Cost-effective feature enhancement for volume datasets. In *15th International Workshop on Vision, Modeling and Visualization*, pages 187–194, 2010.

[27] J. Díaz, E. Monclús, I. Navazo, and P. Vázquez. Adaptive cross-sections of anatomical models. *Computer Graphics Forum*, 31(7):2155–2164, 2012.

[28] J. Díaz and P. Vázquez. Depth-enhanced maximum intensity projection. In *IEEE/EG International Conference on Volume Graphics*, pages 93–100, 2010.

[29] J. Díaz, P. Vázquez, I. Navazo, and F. Duguet. Real-time ambient occlusion and halos with summed area tables. *Computers & Graphics*, 34(4):337–350, 2010.

[30] J. Díaz, H. Yela, and P. Vázquez. Vicinity occlusion maps: Enhanced depth perception of volumetric models. In *Computer Graphics International 2008*, pages 56–63, 2008.

[31] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. In *Computer Graphics Forum*, pages 523–532, 2003.

[32] F. Drago, K. Myszkowski, T. Annen, and N. Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. *Computer Graphics Forum*, 22(3):419–426, 2003.

[33] D.S. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *VIS'00: Proceedings of IEEE Visualization 2000*, pages 195–202, 2000.

[34] K. Engel, M. Hadwiger, J.M. Kniss, C. Rezk-salama, and D. Weiskopf. *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[35] R. Haber and D. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In *IEEE Visualization in Scientific Computing*, pages 74–93. 1990.

[36] C. Hansen and C. Johnson. *The Visualization Handbook*. Academic Press, Inc., Orlando, FL, USA, 2004.

[37] M. Harris, S. Sengupta, and J.D. Owens. Parallel Prefix Sum (Scan) with CUDA. In Hubert Nguyen, editor, *GPU Gems 3*, pages 851–876. Addison Wesley, August 2007.

[38] W. Heidrich, M. McCool, and J. Stevens. Interactive maximum projection volume rendering. In *Proceedings of the 6th conference on Visualization '95*, pages 11–18, 1995.

[39] T. Heimann and H. Delingette. Model-based segmentation. In *Biomedical Image Processing*, Biological and Medical Physics, Biomedical Engineering, pages 279–303. 2011.

[40] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. Fast summed-area table generation and its applications. *Computer Graphics Forum*, 24(3):547–555, 2005.

[41] F. Hernell, P. Ljung, and A. Ynnerman. Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):548–559, 2010.

[42] K. Höhne, M. Bomans, M. Riemer, R. Schubert, U. Tiede, and W. Lierse. A volume-based anatomical atlas. *IEEE Comput. Graph. Appl.*, 12:73–78, 1992.

[43] Y.C. Hu, M. Grossberg, and G.S. Mageras. *Survey of Recent Volumetric Medical Image Segmentation Techniques*. 2009.

[44] V. Interrante and C. Grosch. Strategies for effectively visualizing 3d flow with volume lic. In *Proceedings of the 8th conference on Visualization '97*, pages 421–425, 1997.

[45] S. Islam, D. Silver, and M. Chen. Volume splitting and its applications. *IEEE Transactions on Visualization and Computer Graphics*, 13:193–203, 2007.

[46] H. Jänicke and M. Chen. A salience-based quality metric for visualization. *Compututer Graphics Forum*, 29(3):1183–1192, 2010.

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

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

[49] 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 03*, pages 513–520, 2003.

[50] 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.

[51] J. Kniss, S. Premoze, C. Hansen, and D. Ebert. Interactive translucent volume rendering and procedural modeling. In *Proceedings of the conference on Visualization '02*, pages 109–116, 2002.

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

[53] J. Kontkanen and S. Laine. Ambient occlusion fields. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 41–48, 2005.

[54] J. Kronander, D. Jönsson, J. Löw, P. Ljung, A. Ynnerman, and J. Unger. Efficient visibility encoding for dynamic illumination in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):447–462, 2011.

[55] 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.

[56] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003*, pages 38–, 2003.

[57] H. Landis. Production-ready global illumination. In *ACM SIGGRAPH 2002: Course Notes*, Washington, DC, USA, 2002.

[58] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3), 1988.

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

[60] M. Levoy and R. Whitaker. Gaze-directed volume rendering. In *Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 217–223, 1990.

[61] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph.*, 27:101:1–101:7, 2008.

[62] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph.*, 26, 2007.

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

[64] T. Lokovic and E. Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2000)*, pages 385–392, 2000.

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

[66] T. Luft, C. Colditz, and O. Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics*, 25(3):1206–1213, 2006.

[67] R. Machiraju, J.E. Fowler, D. Thompson, and B. Soni. Evita: Efficient visualization and interrogation of tera-scale data. In *Data Mining for Scientific and Eng. Applications*, pages 257–279, 2001.

[68] M. Malmer, F. Malmer, U. Assarsson, and N. Holzschuch. Fast precomputed ambient occlusion for proximity shadows. *Journal of Graphics Tools*, 12(2):59–71, 2007.

[69] S. Marchesin, J. Dischler, and C. Mongenet. Feature enhancement using locally adaptive volume rendering. In *Proceedings of the Sixth Eurographics / Ieee VGTC conference on Volume Graphics*, pages 41–48, 2007.

[70] S. Marchesin, J. Dischler, and C. Mongenet. Per-pixel opacity modulation for feature enhancement in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):560–570, 2010.

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

[72] M.J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization*, pages 53–60, 2003.

[73] T. McInerney and S. Broughton. Hingeslicer: interactive exploration of volume images using extended 3d slice plane widgets. In *Proceedings of Graphics Interface 2006*, pages 171–178, 2006.

[74] T. McInerney and P. Crawford. Ribbonview: interactive context-preserving cutaways of anatomical surface meshes. In *Proceedings of the 6th international conference on Advances in visual computing - Volume Part II*, pages 533–544, 2010.

[75] J. Mensmann, T. Ropinski, and K. Hinrichs. Interactive cutting operations for generating anatomical illustrations from volumetric data sets. *Journal of WSCG – 16th International Conference in Central*

Europe on Computer Graphics, Visualization and Computer Vision, 16(1-3):89–96, 2008.

[76] M. Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007: Courses*, pages 97–121, 2007.

[77] B. Mora and D.S. Ebert. Low-complexity maximum intensity projection. *ACM Trans. Graph.*, 24(4):1392–1416, 2005.

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

[79] E. Penner and R. Mitchell. Isosurface ambient occlusion and soft shadows with filterable occlusion maps. In *Volume and Point-Based Graphics*, pages 57–64, 2008.

[80] J.D. Pfautz. Depth perception in computer graphics. Technical Report UCAM-CL-TR-546, University of Cambridge, Computer Laboratory, September 2002.

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

[82] B. Preim and D. Bartz. *Visualization in Medicine: Theory, Algorithms, and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[83] T. Ritschel. Fast gpu-based visibility computation for natural illumination of volume data sets. In *Short Paper Eurographics 2007*, pages 17–20, 2007.

[84] T. Ritschel, K. Smith, M. Ihrke, T. Grosch, K. Myszkowski, and H-P. Seidel. 3d unsharp masking for scene coherent enhancement. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3):90:1–90:8, 2008.

[85] T. Ropinski, C. Döring, and C. Rezk Salama. Interactive volumetric lighting simulating scattering and shadowing. In *IEEE Pacific Visualization*, pages 169–176, 2010.

[86] T. Ropinski, Christian Döring, and Christof Rezk Salama. Advanced Volume Illumination with Unconstrained Light Source Positioning. *IEEE Computer Graphics and Applications*, 2010.

[87] T. Ropinski, J. Kasten, and K.H. Hinrichs. Efficient shadows for GPU-based volume raycasting. In *Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2008)*, pages 17–24, 2008.

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

[89] T. Ropinski, F. Steinicke, and K. Hinrichs. Visually supporting depth perception in angiography imaging. In *Proceedings of the 6th International Symposium on Smart Graphics*, pages 93–104, 2006.

[90] A. Rosenfeld and A.C. Kak. *Digital Picture Processing.* Academic Press, Inc., Orlando, USA, 2nd edition, 1982.

[91] R.J. Rost. *OpenGL(R) Shading Language (2nd Edition).* Addison-Wesley Professional, 2005.

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

[93] S. Rusinkiewicz, M. Burns, and D. DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):1199–1205, 2006.

[94] Y. Sato, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Local maximum intensity projection (lmip): A new rendering method for vascular visualization. *Journal of Computer Assisted Tomography*, 22(6):912–917, 1998.

[95] M. Sattler, R. Sarlette, T. Mücken, and R. Klein. Exploitation of human shadow perception for fast shadow rendering. In *Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, APGV '05, pages 131–134, 2005.

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

[97] M. Schott, V. Pegoraro, C.D. Hansen, K. Boulanger, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum*, 28(3):855–862, 2009.

[98] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *SIGGRAPH Comput. Graph.*, 24(5):63–70, 1990.

[99] P-P.J. Sloan, W. Martin, A. Gooch, and B. Gooch. The lit sphere: a model for capturing npr shading from art. In *No description on Graphics interface 2001*, GRIN'01, pages 143–150, 2001.

[100] A.J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proceedings of the 14th IEEE Visualization 2003*, page 47, 2003.

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

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

[103] N. Svakhine, D.S. Ebert, and D. Stredney. Illustration motifs for effective medical volume illustration. *IEEE Comput. Graph. Appl.*, 25:31–39, 2005.

[104] S. Takahashi, Y. Takeshima, I. Fujishiro, and G.M. Nielson. Emphasizing isosurface embeddings in direct volume rendering. In Georges-Pierre Bonneau, Thomas Ertl, and GregoryM. Nielson, editors, *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Mathematics and Visualization, pages 185–206. Springer Berlin Heidelberg, 2006.

[105] Y. Tao, H. Lin, H. Bao, F. Dong, and G. Clapworthy. Feature enhancement by volumetric unsharp masking. *The Visual Computer*, 25(5):581–588, 2009.

[106] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing to enhance real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–1244, 2006.

[107] M. Tatzgern, D. Kalkofen, and D. Schmalstieg. Compact explosion diagrams. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 17–26, 2010.

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

[109] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:511, 2001.

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

[111] J.W. Wallis, T.R. Miller, C.A. Lerner, and E.C. Kleerup. Three-dimensional display in nuclear medicine. *IEEE Transactions on Medical Imaging*, 8(4):297–230, 1989.

[112] Y. Wan and C. Hansen. Fast volumetric data exploration with importance-based accumulated transparency modulation. In *IEEE/EG International Conference on Volume Graphics*, pages 61–68, 2010.

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

[114] 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*, 9:298–312, 2003.

[115] L. Westover. Footprint evaluation for volume rendering. *SIGGRAPH Comput. Graph.*, 24(4):367–376, 1990.

[116] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[117] R. Yagel, A. Kaufman, and Q. Zhang. Realistic volume imaging. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 226–231, 1991.

[118] X. Yuan and B. Chen. Illustrating surfaces in volume. In *Proceedings of Joint IEEE/EG Symposium on Visualization*, pages 9–16, 337, 2004.

[119] C. Zhang and R. Crawfis. Shadows and soft shadows with participating media using splatting. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):139–149, 2003.

[120] K. Zhou, Y. Hu, S. Lin, B. Guo, and H-Y. Shum. Precomputed shadow fields for dynamic scenes. In *ACM SIGGRAPH 2005*, pages 1196–1201, 2005.

[121] Z. Zhou, Y. Tao, H. Lin, F. Dong, and G. Clapworthy. Shape-enhanced maximum intensity projection. *The Visual Computer*, 27(6-8):677–686, 2011.

[122] S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. In *Rendering Techniques*, pages 45–56, 1998.