Università degli Studi di Cagliari

DOTTORATO DI RICERCA

IN INGEGNERIA ELETTRONICA ED INFORMATICA

Ciclo XXIII

Real-time GPU-accelerated Out-of-core Rendering and Light Field Visualization for Improved Massive Volumes Understanding

Settore/i scientifico disciplinari di afferenza

ING-INF/05 SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

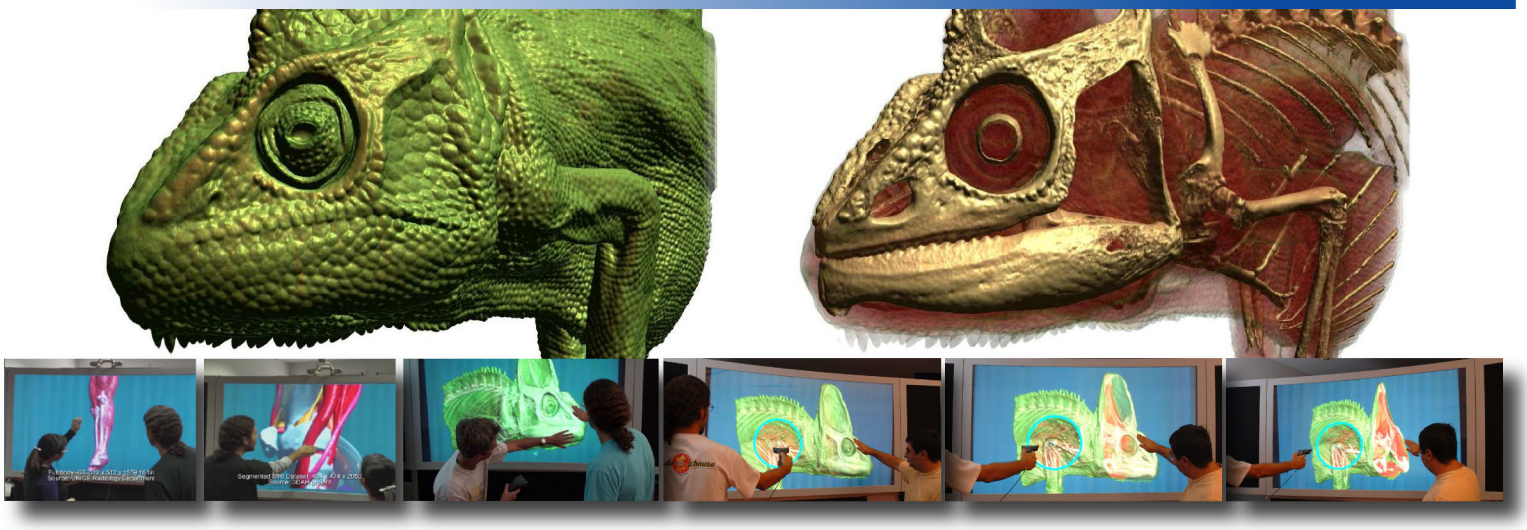
Presentata da:	JOSÉ ANTONIO IGLESIAS GUITIÁN
Coordinatore Dottorato	Prof. ALESSANDRO GIUA
Relatori	Prof. MASSIMO VANZI Dr. ENRICO GOBBETTI

Esame finale anno accademico 2009 - 2010

José A. Iglesias Guitián

Real-time GPU-accelerated Out-of-core Rendering and Light-field Visualization for Improved Massive Volumes Understanding

Ph.D. Thesis



University of Cagliari

Dedicado a Pepe

Dedicated to Pepe

Sommario

Negli ultimi anni si sta verificando una proliferazione sempre più consistente di modelli digitali di notevoli dimensioni in campi applicativi che variano dal CAD e la progettazione industriale alla medicina e le scienze naturali. In modo particolare, nel settore della medicina, le apparecchiature di acquisizione dei dati come RM o TAC producono comunemente dei dataset volumetrici di grosse dimensioni. Questi dataset possono facilmente raggiungere taglie dell'ordine di 1024^3 voxels e dataset di dimensioni maggiori possono essere frequenti.

Questa tesi si focalizza su metodi efficienti per l'esplorazione di tali grossi volumi utilizzando tecniche di visualizzazione diretta su piattaforme HW di diffusione di massa. Per raggiungere tale obiettivo si introducono strutture specializzate multi-risoluzione e algoritmi in grado di visualizzare volumi di dimensioni potenzialmente infinite. Le tecniche sviluppate sono "ouput sensitive" e la loro complessità di rendering dipende soltanto dalle dimensioni delle immagini generate e non dalle dimensioni dei dataset di input. Le caratteristiche avanzate delle architetture moderne GPGPU vengono inoltre sfruttate e combinate con un framework "out-of-core" in modo da offrire una implementazione di questi algoritmi e strutture dati più flessibile, scalabile ed efficiente su singole GPU o cluster di GPU.

Per migliorare la percezione visiva e la comprensione dei dati, viene introdotto inoltre l'uso di tecnologie di display 3D di nuova generazione basate su un approccio di tipo light-field. Questi tipi di dispositivi consentono a diversi utenti di percepire ad occhio nudo oggetti che galleggiano all'interno dello spazio di lavoro del display, sfruttando lo stereo e la parallasse orizzontale. Si descrivono infine un insieme di tecniche illustrative interattive in grado di fornire diverse informazioni contestuali in diverse zone del display, così come un motore di "ray-casting out-of-core" basato su CUDA e contenente una serie di miglioramenti rispetto agli attuali metodi GPU di "ray-casting" di volumi. Le possibilità del sistema sono dimostrate attraverso l'esplorazione interattiva di dataset di 64-GVoxel su un display di tipo light-field da 35-MPixel pilotato da un cluster di PC.

Keywords: Computer Graphics, Visualizzazione Scientifica, Medical Imaging, Volume Rendering, Ray-casting, Illustrative Rendering, Level-of-detail, Light-field Display.

Abstract

Nowadays huge digital models are becoming increasingly available for a number of different applications ranging from CAD, industrial design to medicine and natural sciences. Particularly, in the field of medicine, data acquisition devices such as MRI or CT scanners routinely produce huge volumetric datasets. Currently, these datasets can easily reach dimensions of 1024^3 voxels and datasets larger than that are not uncommon.

This thesis focuses on efficient methods for the interactive exploration of such large volumes using direct volume visualization techniques on commodity platforms. To reach this goal specialized multi-resolution structures and algorithms, which are able to directly render volumes of potentially unlimited size are introduced. The developed techniques are output sensitive and their rendering costs depend only on the complexity of the generated images and not on the complexity of the input datasets. The advanced characteristics of modern GPGPU architectures are exploited and combined with an out-of-core framework in order to provide a more flexible, scalable and efficient implementation of these algorithms and data structures on single GPUs and GPU clusters.

To improve visual perception and understanding, the use of novel 3D display technology based on a light-field approach is introduced. This kind of device allows multiple naked-eye users to perceive virtual objects floating inside the display workspace, exploiting the stereo and horizontal parallax. A set of specialized and interactive illustrative techniques capable of providing different contextual information in different areas of the display, as well as an out-of-core CUDA based ray-casting engine with a number of improvements over current GPU volume ray-casters are both reported. The possibilities of the system are demonstrated by the multi-user interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display driven by a cluster of PCs.

Keywords: Computer Graphics, Scientific Visualization, Medical Imaging, Volume Rendering, Ray-casting, Illustrative Rendering, Level-of-detail, Light-field Displays.

"Roads? Where we're going we don't need roads."

Dr. Emmett Brown

Contents

List of Figures

Preface

This thesis represents a summary of work done from 2008 to 2010 at the Visual Computing group of CRS4 (Center for Advanced Studies, Research and Development in Sardinia) under the supervision of Enrico Gobbetti, whom I want to thank for trusting me and offer me the unique opportunity to work and be part of his research group. During this time I also attented the Ph.D. Program in Electronic and Computer Engineering under the kind tutoring of Massimo Vanzi, who I would like to thank as well. Without both their dedication and guidance this work would not have been possible. In particular I would like to thank my past and present colleagues. Special thanks go to Fabio Bettio, Andrea Giachetti, Fabio Marton, Marco Agus, Gianni Pintore, Ruggero Pintus, Antonio Zorcolo, Yanlin Luo, Jonas Nilsson, Katia Brigaglia and Luca Pireddu. I would also like to express my appreciation for the people with whom I had the pleasure of collaborating these years: Nadia Magnenat-Thalmann and Jérôme Schmid from University of Geneva, Hervé Delingette and Francois Chung from INRIA Sophia-Antipolis, Massimiliano Baleani and Caroline Öhman from the Rizzoli Orthopaedic Institute, Ioana Ciuciu from STARLab, Renato Pajarola and Susanne Suter from University of Zurich and Alex Bronstein from the Technion Institute of Technology. Finally, I specially would like to express my infinite gratitude to my closest family and friends, with special mention to my parents which have always been the main and most reliable support.

The work presented in this thesis has been partially supported by the EU Marie Curie Program, under the 3DANATOMICALHUMAN project (MRTNCT-2006-035763).

Pula, Italy, February 2011 *José A. Iglesias Guitián*

Chapter 1

Introduction

Volumetric datasets are growing at incredible rates in terms of number and size resulting in two visualization challenges: maintaining performance and extracting meaningful information. These two challenges are closely related, since user interaction, which imposes real-time constraints, is a key to volumetric understanding. In this thesis, we introduce scalable methods for rendering volumes of potentially unlimited size on modern GPU architectures. Furthermore we present methods to improve their understanding through illustrative techniques and presentation on advanced 3D displays. This chapter outlines the motivation behind our research, summarizes research achievements, and describes the organization of the thesis.

1.1 Background and Motivations

Scientific visualization is the formal name given in computer science to the field that encompasses data representation and processing algorithms, user interfaces, visual and other sensory representations [McCo 88, Schr 96]. The goal of scientific visualization is to transform data into sensory information in order to glean insight from raw simulation data or data analysis.

Rectilinear scalar volumes, i.e., scalar functions sampled on a 3D grid, are among the most important and challenging datasets in scientific visualization. Volumetric information are generated by simulations as well as by acquisition devices like for example computerized tomography (CT), magnetic resonance imaging (MRI), and ultrasounds. Consequently, volumetric datasets arise in many engineering and scientific areas, such as medicine, non-destructive testing, astronomy, or seismology.

New advances in computer simulation and in data acquisition devices are leading to a steady increase in the resolution of produced datasets. Applied to medical imaging, these advances create new interesting opportunities in diagnostic medicine, surgical simulation or radiation treatment planning. In each case, these opportunities have been brought about by visualizations of portions of the body previously inaccessible to view [DeFa 89].

In this context of emerging data-intensive knowledge discovery and data analysis, the visualization challenge is to create new methods that allow the domain analyst to visually examine this massive amount of data, understand it, and take decisions in a time critical manner. This requires improving both the performance of current visualization systems, to let them deal, interactively, with potentially unlimited amounts of data, and the quality of their representation, to make the data easier to understand. These two aspects are closely related, since user interaction, which imposes real-time constraints, is a key to volumetric understanding.

1.1.1 Volume Rendering of Large Data

Many sophisticated techniques for real-time volume rendering have been proposed in the past, taking advantage of CPU acceleration techniques, GPU acceleration using texture mapping, or special purpose hardware. In the last few years, improvements in the programmable and performance capabilities of GPU processors have made GPU-based solutions the main option for real-time rendering on desktop platforms [Enge 06]. Current high quality solutions, based on ray-casters fully executed in GPU, have demonstrated the ability to deliver real-time frame rates for moderate-size data, but they typically require the entire dataset to be contained in GPU memory. Rendering of large datasets can be achieved through compression, multi-resolution schemes, and out-of-core techniques. Current solutions, however, are not fully adaptive and, with the exception of flat blocking schemes [Ljun 06], are not typically implemented within a single-pass ray casting framework, with increased frame buffer bandwidth demands and/or decreased precision and flexibility in the computation of volume integrals. Scalable single-pass solutions, capable of working out-of-core, are of extreme importance for achieving high performance while supporting a variety of effects.

1.1.2 Improving Understanding through Illustrative Visualization

Resolving the spatial arrangement of complex three-dimensional structures in images produced by direct volume rendering techniques is often a difficult task. In particular, data produced by medical acquisitions often contain many overlapping structures, leading to cluttered images which are difficult to understand. Therefore, enhancing shape perception in volumetric rendering is a very active research area, which is tackled under different angles. Recent contributions include methods for improving photorealistic rendering quality, as well as non-photorealistic approaches to improve image readability with illustrative techniques [Viol 06, Bruc 08]. Illustrative visualization and non-photorealistic rendering (NPR) techniques extend scientific visualization incorporating traditional arts and illustration concepts [Eber 00, Enge 06]. The power of traditional illustration is supported by illustrators long experience in depicting complex shapes in a comprehensible way and their subtle understanding on devising techniques to emphasize or de-emphasize information to effectively communicate various messages to the viewer. Rendering flexibility is thus paramount for a volumetric visualization pipeline.

1.1.3 Improving Understanding through Advanced 3D Displays

Using illustrative visualization methods is not the only way to improve volumetric understanding. An orthogonal research direction consists of presenting results on displays capable of eliciting more depth cues than the conventional 2D monitors or providing improved color reproduction. For instance, Ghosh et al [Ghos 05] have shown how a high dynamic range display can substantially improve volume understanding through perceptually optimized transfer functions. Another possibility consists of enhancing spatial comprehension of 3D data through perceptual cues for accommodation, stereo and motion parallax delivered by a light-field display, i.e., a display supporting high resolution direction selective light emission. This direction looks very promising, since there is evidence that ego- and/or model-motion as well as stereopsis are essential cues to achieve rapid direct perception of

volumetric data [Bouc 09, Mora 04]. Recent advances in 3D display design demonstrate that high resolution display technology capable of reproducing natural light-fields is practically achievable [Balo 05, Jone 07]. Rendering for such displays requires generating a large number of light beams of appropriate origin, direction, and color, which is a complex and computationally intensive task. Moreover, the displays optical characteristics impose specialized rendering methods. For best results, the potential of such displays should also be exploited by specialized illustrative techniques.

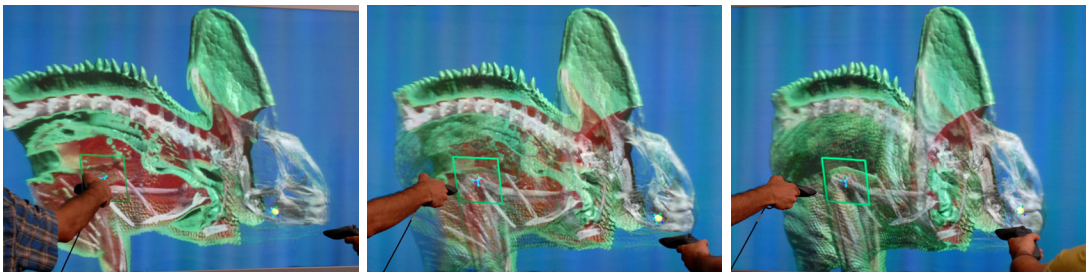


Figure 1.1: **Multi-user interactive exploration of a chameleon 1-GVoxel dataset on a 35-MPixel light-field display.** Users freely mix and match 3D tools creating view-dependent illustrative visualizations. Objects appear floating in the display workspace, providing correct parallax cues while delivering direction-dependent information.

1.2 Objectives

The principal research objective of this thesis is to enable the interactive exploration and better understanding of the information contained in large volumetric datasets using direct volume visualization techniques on commodity computing platforms delivering images using light-field displays. Advancing the state-of-the-art in this area requires solving the following problems:

Improvement in scalability of state-of-the-art rendering methods. Nowadays large datasets are becoming increasingly available and there exists a clear need of specialized and efficient rendering methods. Therefore, we need to study and develop specialized multi-resolution data structures and algorithms able to directly render volumes of potentially unlimited size. Ideally, the developed techniques should be output sensitive, i.e., with a cost depending only on the complexity of the generated images and not on the complexity of the input dataset.

Adapting algorithms to GPGPU parallel architectures. We need to study and develop an efficient implementation of the proposed algorithms and data structures on GPUs or GPU clusters in an out-of-core framework. The solution should keep the scalability, and work both in a desktop setting and in a large scale parallel graphics setting.

Light-field reconstruction and inspection of volumetric data. Since light-field displays provide better depth and shape perception being by eliciting more depth cues, we would like to improve visual perception and understanding using novel 3D display technologies based on the light-field approach. This kind of device allows multiple users to perceive virtual objects floating inside the display workspace, exploiting stereo and motion parallax. In order to make this approach practical, we plan to study methods that exploit perspective coherence in order to improve the rendering performance.

Implementing suitable illustrative techniques. The need to enhance important features present in volumetric datasets invites us to develop and evaluate appropriate illustrative techniques integrated within the previously mentioned out-of-core framework. Specifically, it would be worth to exploit the view-dependent characteristics of the light-field displays.

Validation of the system. Potential perceptual advantages of the approach should be validated through user testing, e.g., by evaluating the layout discrimination capabilities of the approach in comparison to standard techniques.

While some partial solutions for some of these problems have been proposed in the recent years, there is currently no existing single approach able to fulfill all these requirements.

1.3 Achievements

The research work carried out during this thesis has led to the following achievements:

- The introduction of a novel single-pass ray casting framework for interactive out-of-core rendering of massive volumetric models. The method is GPU-accelerated and has demonstrated the capability of managing multi-gigavoxel datasets [Gobb 08]. The key insight of the method is to use specialized multi-resolution structures, separating visibility-aware level-of-detail selection from the actual rendering using co-operative algorithms.

- The generalization of the previous ray casting framework using the possibilitues offered by modern GPGPU architectures [Agus 08c]. The method supports a more flexible ray traversal (e.g., changes in the direction of the ray propagation or different accumulation strategies) and proposes an improved solution for incorporating visibility feedback.

- The development of a volume representation technique [Agus 10b] suitable for cases where the volumes represent physical objects with well defined boundaries separating different materials, giving rise to models with quasi-impulsive gradient fields. In this representation, we replace blocks of N^3 voxels by one single voxel that is split by a feature plane into two regions with constant values. We also show how to convert a standard mono-resolution representation into an out-of-core multi-resolution structure, both for labeled and continuous scalar volumes.

- The introduction of an adaptive technique for the interactive rendering of volumetric models on projector-based multi-user light-field displays. The method achieves interactive performance and provides rapid visual understanding of complex volumetric datasets even when using depth-oblivious compositing techniques [Agus 08c, Agus 08a, JAIg 08].

- The development of a new interactive visualization framework which enables multiple naked-eye users to perceive detailed multi-gigavoxel volumetric models as floating in space, responsive to their actions, and delivering different information in different areas of the workspace [Agus 09, Igle 10]. The main contributions include a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of the display, as well as an out-of-core CUDA-based ray casting engine with

a number of improvements over current GPU-accelerated volume ray-casters. The possibilities of the system have been demonstrated by the multi-user interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display driven by a cluster of PCs.

- The evaluation of volume rendering on light-field displays and its relevance for medical training and virtual examinations. Initial results demonstrate increased efficiency in tasks requiring spatial understanding. The development of preliminary psycho-physical tests which demonstrate that light-field displays and virtual reality improve understanding in common medical tasks [Agus 08b]. The continuation of such perceptual experiments to evaluate the depth discrimination capabilities of the light-field display technology with respect to two-view (stereo) and discrete multi-view designs [Agus 10a]. The evaluation employ a large scale multi-projector 3D display offering continuous horizontal parallax in a room size workspace. Two tests are considered in the context of depth oblivious rendering techniques: a layout discrimination task, and a path tracing task.

1.4 Organization

This thesis is organized in order to show in a natural and coherent order all the results obtained. Many readers would prefer to skip parts of the text and go back and forth through the different chapters. In this section the reader can find a brief overview of what can be found in each chapter.

Chapter 2: GPU-Accelerated Out-of-core DVR. It presents an adaptive out-of-core technique for rendering massive scalar volumes employing single-pass GPU ray casting. The method is based on the decomposition of a volumetric dataset into small cubical bricks, which are then organized into an octree structure maintained out-of-core. Co-operation between CPU and GPU make it possible to interactively explore extremely massive volumes using an output-sensitive method. Results obtained demonstrate how is possible to interactively explore multi-gigavoxel datasets on a desktop PC by using the proposed method.

Chapter 3: Taking Advantage of GPGPU Architectures. This chapter describes a series of improvements on presented multi-resolution volume rendering method in chapter 2. The new rendering system is constructed around a configurable GPU ray casting kernel, exploiting the advanced characteristics of modern GPGPU architectures to achieve both flexibility and performance.

Chapter 4: Handling Discontinuous Datasets. This chapter introduces a new volumetric primitive, named *split-voxel*, to handle datasets containing sharp boundaries and discontinuities. This chapter shows how to convert a standard mono-resolution representation into an out-of-core multi-resolution structure, both for labeled and continuous scalar volumes using the *split-voxel* primitive. It also shows how it is possible to interactively explore the resulting models using a multi-resolution GPU ray casting framework.

Chapter 5: Rendering on Light-field Displays. This chapter presents a GPU-accelerated volume ray casting system interactively driving a multi-user light-field display. The display is based on a specially arranged array of projectors and a holographic screen that provides full horizontal parallax. The characteristics of the display are exploited to develop

a specialized volume rendering technique able to provide multiple freely moving naked-eye viewers the illusion of seeing and manipulating virtual volumetric objects floating in the display workspace. The method achieves interactive performance and provides rapid visual understanding of complex volumetric datasets even when using depth oblivious compositing techniques. Perceptual experiments are also presented to evaluate the depth discrimination capabilities of this technology.

Chapter 6: Illustrative Techniques. In this chapter, we report on a set of illustrative and non-photorealistic rendering (NPR) techniques that complement the work presented in chapters 2, 3 and 5. This chapter proposes an illustrative technique in the field of context-preserving volume rendering, named *context-preserving focal probes*, to focus the users' attention in a region of interest while preserving the information in context. The focus and context information are separated by the assignment of different rendering styles that can be smoothly blended to provide a more continuous effect. This chapter also introduces a new accumulation scheme for importance-driven volume rendering and a set of specialized interactive illustrative techniques capable of providing different contextual information in different areas of a light-field display.

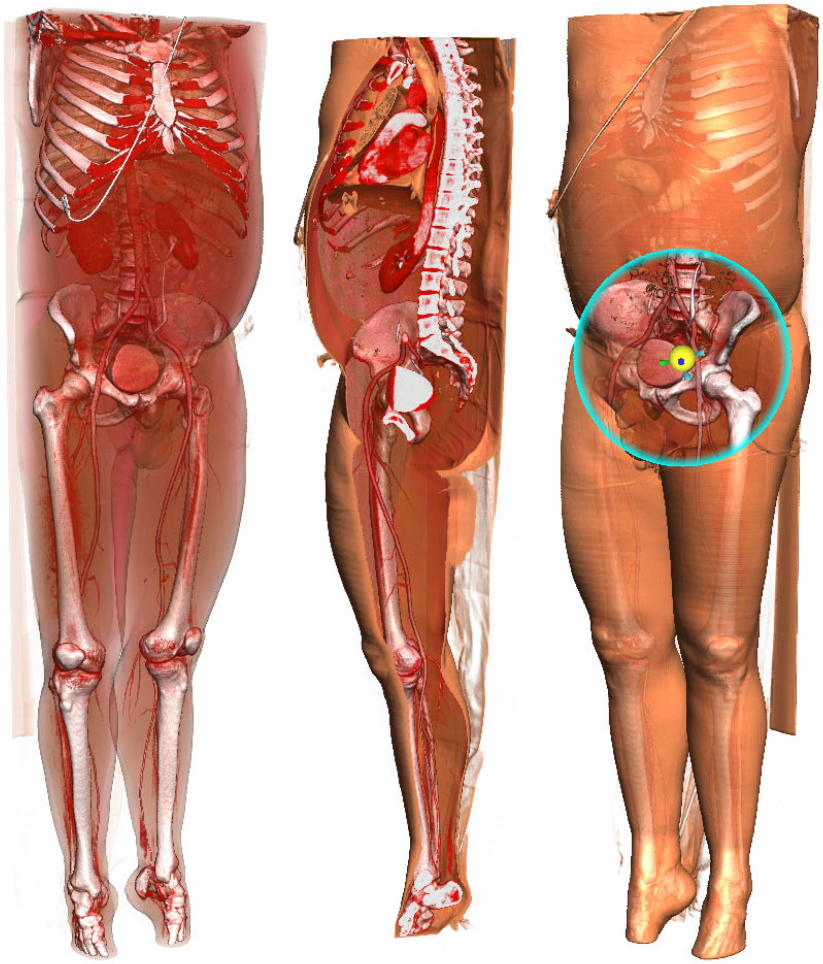


Figure 1.2: **GPU-accelerated ray caster working with medical data.** GPU rendering images of a medical dataset demonstrating some advanced exploration capabilities.

Chapter 2

GPU-Accelerated Out-of-core DVR

The first research objective of this thesis is to make it possible to visualize massive scalar volumes at interactive rates on commodity graphics platforms. This chapter presents a novel adaptive out-of-core technique for rendering massive scalar volumes employing single-pass GPU ray casting. The method is based on the decomposition of a volumetric dataset into small cubical bricks, which are then organized into an octree structure maintained out-of-core. The key insight of the method is to use specialized multi-resolution structures, separating visibility-aware level-of-detail selection from the actual rendering using co-operative algorithms. Results obtained with the proposed method demonstrate how multi-gigavoxel datasets can be interactively explored on a desktop PC.

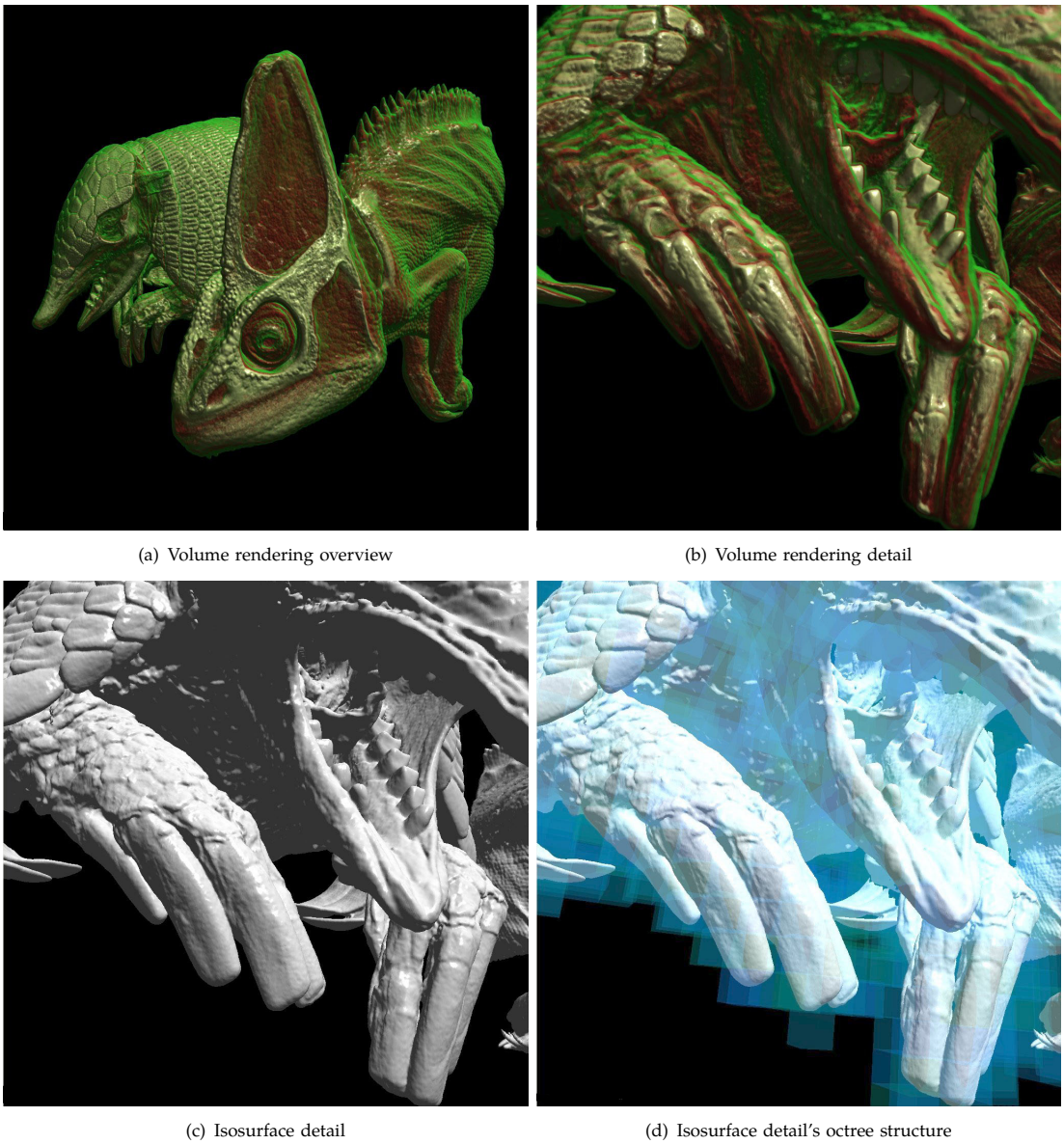
2.1 Introduction

The ability to interactively render rectilinear scalar volumes containing billions of samples on desktop PCs is of primary importance for a number of applications, which include medical visualization, industrial engineering and numerical simulation results analysis.

Many sophisticated techniques for real-time volume rendering have been proposed in the past, taking advantage of CPU acceleration techniques, GPU acceleration using texture mapping, or special purpose hardware. In the last few years, improvements in the programmable and performance capabilities of GPUs have made GPU solutions the main option for real-time rendering on desktop platforms [Enge 06]. Current high quality solutions, based on ray-casters fully executed in GPU, have demonstrated the ability to deliver real-time frame rates for moderate-size data, but they typically require the entire dataset to be contained in GPU memory. Rendering of large datasets can be achieved through compression, multi-resolution schemes, and out-of-core techniques.

Current solutions, however, are not fully adaptive and, with the exception of flat blocking schemes [Ljun 06], are not typically implemented within a single-pass ray casting framework, with increased frame buffer bandwidth demands and/or decreased precision and flexibility in the computation of volume integrals (see section 2.2).

In order to remove such limitations, we presented an adaptive out-of-core technique for rendering massive scalar datasets within a single-pass GPU ray casting framework. The method exploits an adaptive loader executing on the CPU for updating a working set of bricks maintained on GPU memory by asynchronously fetching data from an out-of-core volume octree representation. At each frame, a compact indexing structure, which spatially organizes the



(a) Volume rendering overview

(b) Volume rendering detail

(c) Isosurface detail

(d) Isosurface detail's octree structure

Figure 2.1: **Interactive exploration of multi-gigabyte CT datasets.** This 2-GVoxel 16bit dataset is interactively explored on a desktop PC with a NVIDIA 8800 Ultra graphics board using a 1024×1024 window size. Transfer functions and isovalues can be interactively changed during navigation. The volume rendered images have a full Phong model with specular reflections and view-dependent transparency.

current working set into an octree hierarchy, is encoded in a small texture. This data structure is then exploited by an efficient ray casting algorithm, which computes the volume rendering integral by enumerating non-empty bricks in front-to-back order and adapting sampling density to brick resolution. The algorithm is a streamlined octree extension of an efficient stackless ray traversal method for kd-trees [Havr 98, Popo 07], which reduces costly texture memory accesses by computing neighbor information on-the-fly. In order to further optimize memory and bandwidth efficiency, the method also exploits feedback from the renderer to avoid refinement and data loading of occluded zones.

Although not all the techniques presented here are novel in themselves, their elaboration and combination in a single system is not trivial and represents a substantial enhancement to

the state of the art. The resulting method is extensible, fully adaptive, and able to interactively explore multi-gigavoxel datasets on a desktop PC (see Fig. 2.1).

2.2 Related Work

In the context of this thesis, the discussion of the state-of-the-art will be limited to the approaches most closely related to massive volume visualization. The out-of-core organization of massive volumetric data into a volume octree is a classic one. Lamar et al. [LaMa 99] proposed a multi-resolution sampling of octree tile blocks according to view-dependent criteria. Boada et al. [Boad 01] proposed a coarse octree built upon uniform sub-blocks of the volume, and used, instead, data dependent measures to select block resolution. In such systems, as in most previous GPU accelerated multi-resolution schemes, rendering of multi-resolution volumes on graphics hardware is accomplished by separate rendering of blocks and frame buffer composition. For instance, Guthe et al. [Guth 04] exploits a decomposition into wavelet compressed blocks, uses block resolution to determine inter-slice distance, and introduces methods for empty space skipping and early ray termination. Li et al. [Li 03] propose to accelerate slice-based volume rendering by skipping empty blocks and exploiting an opacity map for occlusion culling. Slice-based implementations are, however, rasterization limited and hard to optimize from an algorithmic point of view. Furthermore, when applying a perspective projection the integration step size will vary along viewing rays when using planar proxy geometries, leading to visible artifacts. In order to solve some of these problems, other authors [Hong 05, Kaeh 06] separately render blocks using volumetric ray casting on the GPU and devise propagation methods to sort cells into layers for front-to-back rendering, therefore reducing frame-buffer demands. The separate rendering of blocks, however, is prone to rendering artifacts at block boundaries, and does not easily allow an implementation of optical models with viewing rays changing direction, as it does occur in refracting volumes, or with non-local effects, as it does occur in global illumination.

The technique proposed as part of this thesis is based on a full-volume GPU ray-casting approach [Krug 03, Roet 03], with a fragment shader that performs the entire volume traversal in a single pass [Steg 05]. Such an approach, made possible by modern programmable GPUs, is more general, but, until very recently, has been limited to moderate size volumes that fit entirely into texture memory. In this context, the issue of large volumes has been typically addressed by compressing data using adaptive texturing schemes to fit entire datasets into GPU memory in compressed form [Voll 06], or by using flat multi-resolution blocking methods [Ljun 06]. In the first approach, data is stored at various resolution levels using adaptive texture maps [Krau 02] to reduce storage needs, but sampling density is not adapted as the ray passes through different blocks of data. The flat multi-resolution blocking technique, instead, represents a volume as a fixed grid of blocks and varies the resolution of each block to achieve adaptivity. The disadvantage of this fine-grained approach in comparison with a hierarchical approach is that the number of blocks is constant and the method remains performing only if individual blocks are within a small range of sizes.

The proposed method, instead, relies on the ability to rapidly traverse an octree structure and is based on the stackless ray traversal method for kd-trees [Havr 98] recently extended to GPUs for surface rendering [Popo 07]. The method exploits the regular structure of octrees to reduce costly texture memory accesses by computing bounding boxes on-the-fly. In addition,

the proposed algorithm takes advantage in its fragment shader implementation of occlusion queries to avoid loading occluded data. Other authors have proposed using depth information to optimize full-volume GPU ray-casters, but, in general, the focus is on implementing early-ray termination in multi-pass methods by exploiting early z-tests features [Krug 03, Roet 03]. A fragment shader based scheme can exploit spatial and temporal coherence to schedule queries in an order that strives to reduce end-to-end latency, similarly to what is done for recent surface renderers [Govi 03, Bitt 04]. The central idea of these methods is to issue multiple queries for independent scene parts and to avoid repeated visibility tests of interior nodes by exploiting the coherence of visibility classification. The partitioning in the proposed scheme occurs in image space, rather than in object space.

At the time the proposed approach were published, there were not fully adaptive solutions and, with the exception of flat blocking schemes [Ljun 06], were not typically implemented within a single-pass ray casting framework, with increased frame buffer bandwidth demands and/or decreased precision and flexibility in the computation of volume integrals. Just after this work were published [Gobb 08], a close related work were also presented by Crassin et al. [Cras 09]. Similarities between both works were pointed out and discussed in their paper. More precisely, Crassin et al. [Cras 09] exploit multiple render targets to store a subset of the traversed nodes for each pixel and exploit spatio-temporal coherence trying to not miss visible nodes. Using mipmapping they address the aliasing to a large extent but increase the memory consumption and makes tree updates more challenging.

2.3 Method Overview

Since massive volumetric datasets cannot be interactively rendered by brute force methods, applications must ideally employ adaptive rendering techniques whose runtime and memory footprint is, as much as possible, proportional to the number of image pixels, not to the total model complexity.

For efficiently updating the rendering working-set, these methods require the integration of level-of-detail and visibility culling techniques. Out-of-core data management is used for filtering out as efficiently as possible the data that is not contributing to a particular image.

The technique separates the creation and maintenance of the rendering working set, which is performed on the CPU, from the actual rendering, which is fully performed on the GPU based on an efficient encoding of the current working set representation (see Fig. 2.3). In order to maximize CPU and bandwidth efficiency, a coarse-grained volume decomposition is employed, which allows to amortize decision costs over a large number of rendered voxels and to efficiently update the GPU representation with few calls.

The original volumetric model is decomposed into small cubical bricks, which are then organized into a coarse octree structure maintained out-of-core. The octree contains the original data at the leaves, and a filtered representation of children at inner nodes. Each node also stores the range of values, as well as, optionally, precomputed gradients. In order to efficiently support runtime operations that require access to neighboring voxels, such as linear interpolation or gradient computations, blocks are made self-contained by replicating neighboring samples. One layer is replicated for linear interpolation support, while two layers are replicated for additionally using central differences to compute gradients at rendering time. At runtime, an adaptive loader updates a view- and transfer function-dependent working set of

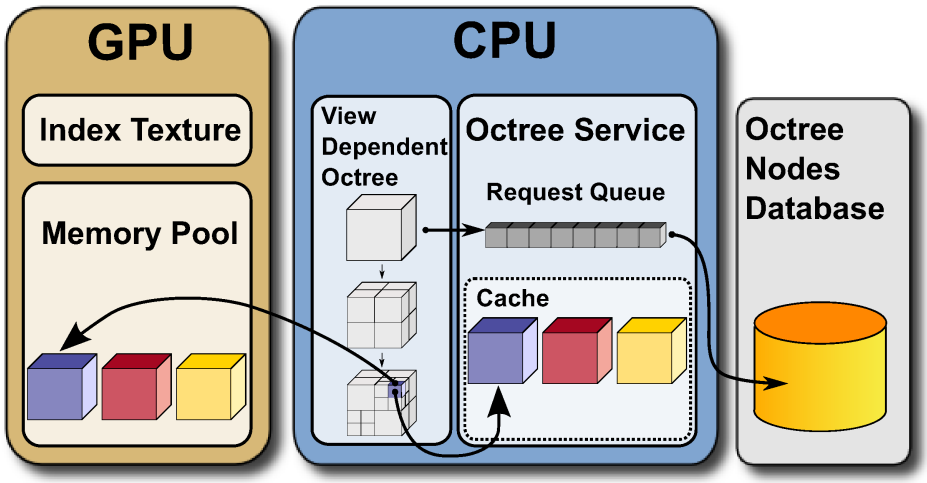


Figure 2.2: **Method overview.** At runtime, an adaptive loader, executing on the CPU, updates a view- and transfer function-dependent working set of bricks maintained on GPU memory by asynchronously fetching data from an out-of-core coarse-grained octree representation. A compact indexing structure that spatially organizes the current working set is exploited by an efficient stackless GPU ray-caster for image generation.

bricks incrementally maintained on CPU and GPU memory by asynchronously fetching data from the out-of-core octree. The working set is maintained by an adaptive refinement method guided by suitably computed node priorities (see section 2.3.1). At each frame, a compact indexing structure, which spatially organizes the current working set into an octree hierarchy, is encoded in a small texture. This structure is not a multi-resolution data representation, but simply spatially organizes the leaves of the current view-dependent representation into an octree with neighbor pointers. The inner nodes of this structure simply contain pointers to children, and only the leaves refer to volume data nodes stored in the memory pool. The spatial index structure is exploited by an efficient stackless GPU ray-caster, which computes the volume rendering integral by enumerating non-empty bricks in front-to-back order, adapting sampling density to brick resolution, and stopping as soon as the accumulated opacity exceeds a certain threshold, updating both the frame- and depth-buffer (see section 2.3.2).

Using an occlusion query mechanism designed to reduce GPU stalls when using a shader based implementation or taking advantage of the scatter memory write capability when using a GPGPU parallel architecture, feedback from the renderer can be exploited by the loader to avoid refinement and data loading of occluded zones (see section 2.3.3).

2.3.1 Generation of View- and Transfer-Function Dependent Working Sets

At each frame, an incremental refinement procedure constructs the current view-dependent working set by refining a sorted set of visible non-empty nodes initialized with the octree root. In the most basic case, the set is sorted by decreasing projected screen-space size of voxels, but it will be shown in section 2.3.3 how visibility information can also be incorporated in the process. Empty nodes are terminal ones in the refinement process, as well as nodes which fall outside of the view frustum. As in [Wilh 92], for isosurface rendering, a node is considered non-empty if the isovalue is within the range spanned by the minimum and maximum value

of the cell. In the case of volume rendering, as in [Scha 05], summed-area tables of the transfer function opacity are used to determine if the block is empty by taking the difference of the table entries for the minimum and maximum block values. The refinement procedure stops when all nodes are considered adequately refined, no data is currently available in-core to perform a refinement, or no more space is available in the GPU cache to contain a further subdivision. In order to hide out-of-core data access latency, all data access requests are performed asynchronously by a separate thread, and refinement continues only if data is immediately available.

At the end of the refinement process, all nodes in the current working set are present both in the CPU and GPU cache, the CPU cache being used as a larger level-2 cache that uses system memory to avoid disk accesses for recently used blocks. Both memory pools are managed using a LRU policy. The GPU texture cache is devised as a large preallocated 3D texture managed as a pool of blocks, or two of them when using precomputed gradients.

A possible implementation can be to encode the value texture as a 16bit single channel texture, while the gradient texture as a RGBA8 texture with normalized gradients in the RGB components and gradient norm in the A component. During incremental refinement, the GPU texture cache is incrementally updated with `glTexSubImage` calls to move data. Due to temporal and spatial coherence, the number of updated nodes per frame is generally small.

2.3.2 GPU Rendering

The incremental refinement procedure defines a cut of the octree hierarchy that is considered adequate for the current frame and stores all data blocks associated to non-empty leaves in GPU memory. In order to render an image with a single-pass GPU ray-caster, the fragment shader must be able to efficiently enumerate in front-to-back order for each fragment all the blocks pierced by the associated view ray. This goal is achieved by using an octree with neighbor structure to spatially index the current leaf blocks, and using this structure to accelerate ray traversal.

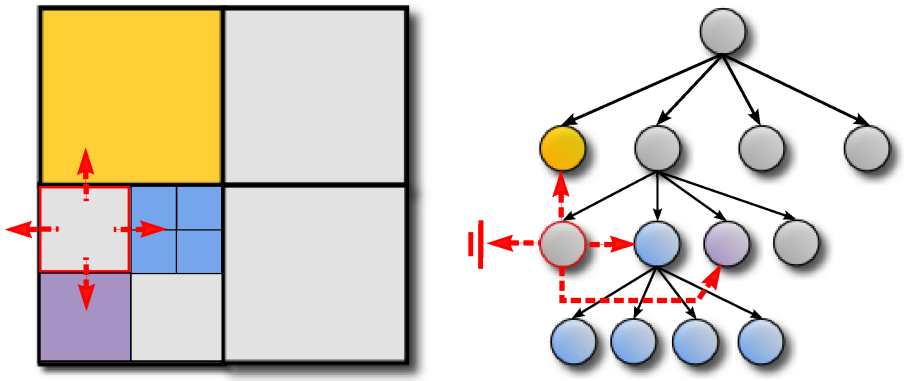


Figure 2.3: **Octree with neighbor pointers.** Neighbor pointers link each leaf node of the octree via its six faces directly to the corresponding adjacent node of that face, or to the smallest node enclosing all adjacent nodes if there are multiple ones.

2.3.2.1 Spatial Index Construction

The octree with neighbors structure augments a branch-on-need octree with links, so that a direct traversal to adjacent nodes is possible (see Fig. 2.3). In this structure, neighbor pointers directly link each leaf node of the octree via its six faces to the corresponding adjacent node of that face, or to the smallest node enclosing all adjacent nodes if there are multiple ones. Such a structure is created on-the-fly at each frame directly from the view-dependent octree, and encoded into a 3D texture that acts as a spatial index.

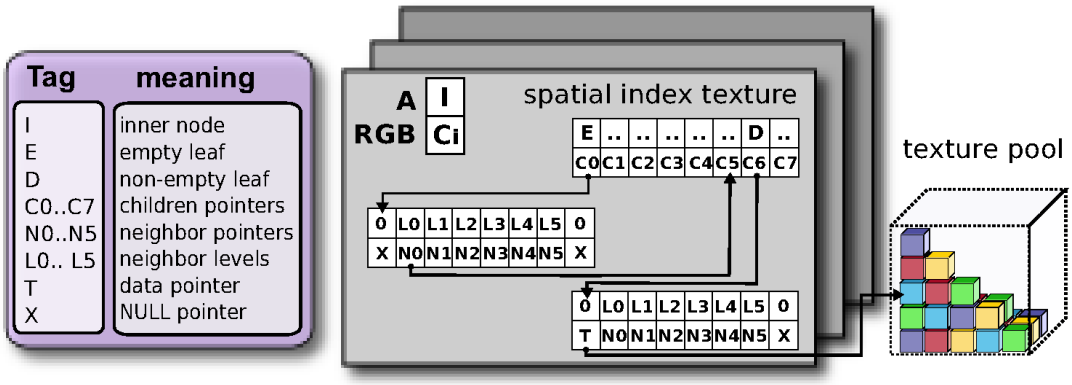


Figure 2.4: **GPU spatial index and memory pool textures.** Each octree node is encoded in 8 of consecutive texels arranged in the x direction.

The layout of the spatial index texture is designed to encode the minimum amount of data required for octree traversal (see Fig. 2.4). Similarly to Octree Textures on GPU [Lefe 05], an 8 bit $RGBA$ texture is used for encoding information in the RGB component pointer information, and various kinds of tags in the A components. With this encoding, it's possible to potentially address $16M$ nodes or data blocks, which is several orders of magnitude larger than what is needed.

The octree structure is encoded using a tagged pointer per node. The A component of the tagged pointer determines the node kind, and it is $A = 1.0$ for inner nodes, $A = 0.5$ for for data nodes, and $A = 0.0$ for empty nodes. Inner nodes used the RGB component of the tagged pointer to point to the first of 8 children arranged consecutively in the x direction. Leaf nodes use the RGB component of the tagged pointer to point to leaf information, which consists in a data pointer (if the node is not empty), and 6 consecutive texels storing pointers to neighbors. The last leaf data texel always contains a NULL pointer, used in our traversal code to simplify the handling of rays that do not exit from the current box and thus should not continue to neighbors. Neighbor pointers use their A value to encode the octree level of the neighbor, which can be the same as the level of current node or coarser. The level information is all that is required in our traversal algorithm to rapidly compute the bounding box information of neighbors during traversal.

The on-the-fly construction of the index texture is fast, since the employed structure is coarse-grained and the view-dependent tree is composed of only a few thousands leafs. Once the structure is constructed in a memory area, the GPU texture is updated using a `glTexSubImage` call, and the fragment shader implementing volume ray casting is activated by rendering a quad.

2.3.2.2 Spatial Index Traversal

The spatial index structure is exploited by an efficient ray casting algorithm, which computes the volume rendering integral by enumerating non-empty bricks in front-to-back order and adapting sampling density to brick resolution. The traversal algorithm is a streamlined octree extension of an efficient stackless ray traversal method for kd-trees [Havr 98, Popo 07], which reduces costly texture memory accesses by computing neighbor information on-the-fly (see Fig. 2.3.2.2). In our approach, children and neighbor bounding boxes are implicitly computed on-the-fly by the shader without any additional access to texture memory by exploiting the regular structure of the octree. The basic concept behind the stackless traversal is to start by performing a down traversal of the octree for locating the leaf node that contains the current sampling position, which, at the start, is the position at which the ray enters the volume. Then, the leaf node is processed by accumulating color and opacity by stepping through the associated brick if it contains data, or simply skipping the node if it is empty. If the ray does not terminate because maximum opacity is reached, the algorithm determines the face and the intersection point through which the ray exits the node. Then traversal continues by following the neighbor pointer of this face to the adjacent node, eventually performing a down traversal to locate the leaf node that contains the exit point, which is now the entry point of the new leaf node. This approach has the important advantage of not requiring a stack to remember nodes that still need to be visited, since the state of the ray only consists of its current node and its entry point.

```
while (!is_null(node_ptr) and color.a<1) {
  // Find leaf containing current sampling point
  P = ray.start+ray.dir*t_min;
  node = tex3d(spatial_index, node_ptr);
  while (is_inner(node.w)) {
    box_dim/=2; box_mid=box_min+box_dim;
    s=step(P,box_mid); box_min+=s*box_dim;
    child_offset=dot3(s,float3(1,2,4))*texel_sz;
    node_ptr=node.xyz+float4(child_offset,0,0);
    node=tex3d(spatial_index, node_ptr);
    ++octree_level;
  }
  // Clip ray to box and find exit face
  (box_t_max, exit_face_idx, exit_dir) =
     box_clip(ray, t_min, t_max, box_min, box_dim);
  // If non-empty block, access data and accumulate
  if (!is_empty(node.w)) {
    data_ptr=tex3d(spatial_index, node.xyz);
    (fragment.color, fragment.depth) =
      accumulate(fragment.color,
                 ray, t_min, box_t_max,
                 data_ptr, box_min, box_max);
  }
  // If ray exits from current block, move to neighbor
  neighbor_offset=float3(1+exit_face_idx,0,0)*texel_sz;
  neighbor=tex3d(spatial_index, node.xyz+neighbor_offset);
  node_ptr=neighbor.xyz;
  octree_level=neighbor.w;
  box_dim=exp2(-octree_level);
  box_min=trunc(box_min/box_dim)*box_dim;
  t_min=box_t_max;
}
```

Figure 2.5: **Stackless octree traversal on the GPU.** The code minimizes memory accesses by computing visited boxes on the fly.

In order to implement this approach, the information a node should provide in addition to tags and pointers consists in its bounding box, which is used for locating points during a down traversal, and exit ray positions during neighbor traversal. All the computations are

done in texture coordinates, and it's assumed that the octree subdivides the unit cube. During all the traversal steps, the current box is maintained, initialized with the unit cube, and also as the current octree level, initialized at 0.

A down traversal step from a node to the child containing a point P can be efficiently implemented by box subdivision, using a step function that compares P with the center of the current box. For each component of P, this function returns 0 if that component is less than the center's value, and 1 otherwise. The child box coordinates are thus obtained by translating the parent box origin by an amount $step_i \times child\_box\_dim_i$, where i is x, y or z. The values returned by the step function are also combined to access the proper children pointer inside the spatial index texture, which is stored at an offset $step_x * 1 + step_y * 2 + step_z * 4$ from the current pointed node. Each time the box is subdivided, the octree level is incremented by one.

Computing the bounding box of a neighbour relies on octree level tracking. When moving to a neighbor, the origin of the box is in a first step simply shifted in the direction of the box face from which the ray is exiting by translating it by $dir * box\_dim$, where the exiting face direction dir is either $\pm x$ or $\pm y$ or $\pm z$. This operation is sufficient to compute the neighbor box if it is at the same octree level. If, instead, the neighbor level is at a coarser level than the current node, the shifted box must be coarsened, an operation that can performed efficiently in closed form by first updating the box dimension to $box\_dim = 2^{-neighbor\_level}$ and then snapping the box origin to the $neighbor\_level$ grid by computing $box\_min = \lfloor box\_min/box\_dim \rfloor \cdot box\_dim$.

2.3.2.3 Adaptive Sampling

The stackless traversal technique allows the fragment shader to enumerate all non-empty leaves pierced by a ray in front-to-back order. Each time the ray enters a leaf with data, the ray chooses a step size matching the local voxel density and accumulates color and opacity information depending on the active rendering mode. Entry and exit points within the block are determined during the octree traversal process. For isosurface rendering, the program simply look for intervals that bracket the selected isovalue, while for semitransparent direct volume rendering, it uses a Phong illumination model with boundary enhancement and view-dependent transparency [Bruc 07a]. Stepping by a discrete number of intervals, which are directly associated to octree levels, enables the use of a compact precomputed 2D transfer function, using the octree level for the second dimension, where the transfer function opacity and color weighting are adjusted accordingly. Both the isovalue and the transfer function can be modified interactively. When the ray exits the block, the current opacity is checked, and, if it exceeds a certain threshold (0.99 in our tests), the ray terminates and the depth of the fragment is updated.

2.3.3 Incorporating Visibility Information

With the described techniques, it is possible to perform fully adaptive GPU ray casting: as a matter of fact the structure provides support for empty space skipping, adaptive sampling and occlusion culling through early ray termination. Occlusion culling during ray accumulation is performed by early ray termination. This approach optimizes computations but is not optimal in terms of data management, since occluded areas are discovered only at rasterization time, after the data is already part of the working set. In order to avoid wasting GPU memory

resources and exploit bandwidth, a feedback mechanism is incorporated in the system, that allows us to exploit visibility information gathered during rendering in the loader.

The basic principle of the method is to update at each frame the visibility status of the nodes in the graph, and, during the refinement cycle, only refine nodes that were marked as visible during the previous frame. Under this approach, the available GPU texture slots will be used mainly to refine nodes present in the visible part of the model, and load requests will not be posted for invisible ones. In order to gather node visibility information, the fragment shader write the depth of the last visited sample into the depth buffer. By issuing an occlusion query for the bounding box of non-empty leaves after volume rendering has finished, visibility information can be gathered by exploiting the rasterization hardware. If the occlusion query results indicates that the number of visible pixels is below a visibility threshold (4 pixels were used for the benchmarks), the node is marked not visible. The visibility information of leaves can then be propagated up to the root by considering an inner node visible only if at least one of its children is visible.

A straightforward implementation of this method, which would issue queries for all rendered nodes after the volume rendering call, is possible, but would be inefficient. It should be noted that, although queries are processed quickly using the rasterization power of the GPU, their results are not available immediately, due to the delay between issuing a query and its actual processing by the graphics pipeline, which would only occur when the rasterizers have finished with ray casting. For this reason, exploiting spatial and temporal coherence is preferred to schedule queries in a way that reduces end-to-end latency. Since we perform full-volume ray casting, the rendering procedure is separated into independent parts using a screen space subdivision. Given a budget of visibility queries per frame, the screen is recursively partitioned into tiles with the purpose of separating blocks for which visibility will be queried from all others (see Fig. 2.6). The partitioning uses a 2D binary space subdivision of the list of 2D rectangles that bound the block projection. Computing this set adds little overhead, since node bounding boxes are already projected onto the screen when computing node priorities. At each subdivision step, a split plane along one of the axes is placed at the position that separates the set into two equally sized subsets, defining two screen tiles. One of the two tiles is selected as containing the occlusion queried subset of octree nodes. The subdivision process continues only on that tile, until its size is below a certain threshold. At each frame, different decisions are taken when selecting the half-spaces to ensure that the entire set of octree nodes is covered after a minimum number of frames. At the end of this process, a list of tiles partitioning the original viewport is obtained, as well as a list of blocks whose projection is entirely contained within one of the tiles, placed at the beginning of the list. The rendering process continues by instructing the graphics pipeline to raycast each of the tiles, and to perform occlusion queries for the selected subset of octree nodes just after issuing the rendering command of the first one (see Fig. 2.6). This approach reduces end-to-end latency because of the interleaving of the processing of occlusion queries in the first tile with the rasterization of the other tiles. Visibility information is incorporated in the system with a delay of $N+1$ frames, where N is the number of partitions required to cover all blocks. In practice, this number is very small, 2 to 4 in our tests, since we use a coarse-grained octree subdivision and the working set if made up of few hundreds to few thousands nodes.

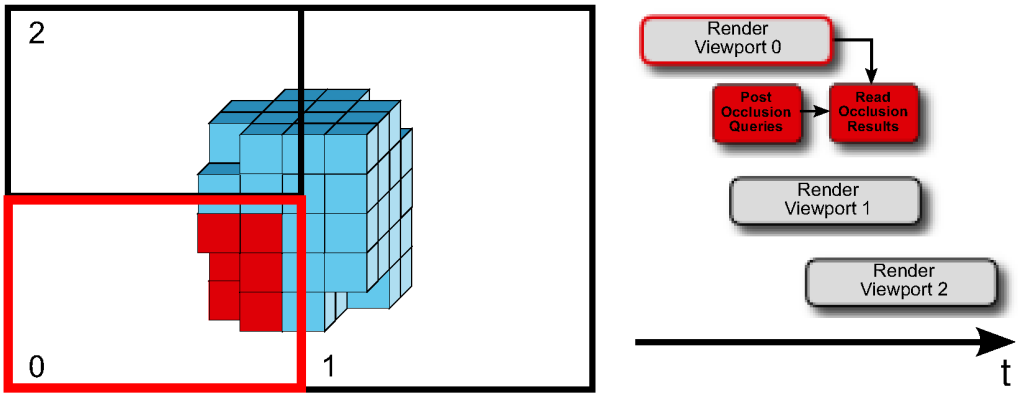


Figure 2.6: **Screen space subdivision and occlusion query scheduling.** The screen is recursively partitioned into tile for the purpose of separating blocks for which visibility will be queried from other blocks, thus enabling interleaving of query processing with rasterization.

2.3.4 Implementation and Results

An experimental software library and a rendering application supporting the technique have been implemented on Linux using C++ with OpenGL, and Cg 2.0. The octree is stored in an out-of-core structure, based on Berkeley DB, and data is lossy compressed with the LZO compression library.

We have tested our system with a variety of high resolution models. In this section, we discuss the results obtained with the inspection of a large volumetric model containing two high resolution X-Ray CT datasets of biological specimens [1]. The overall volume has a resolution of $2048 \times 1024 \times 1080$ with 16 bit/sample, and has been embedded in a 2048^3 cubical grid.

All tests have been performed on a Linux PC with a dual 2.4GHz CPU, 4GB RAM, a GeForce8800 Ultra graphics board and SATA2 disks storing the out-of-core models. The construction of the octree with precomputed gradients from the source datasets was performed using a granularity of 32^3 for octree bricks with 1 layer overlap. The pre-processor was instructed to filter out data by discarding all blocks with a value lower than 6400, in order to discard most of the noisy empty space. Data pre-processing took 95 minutes to complete on a single CPU and produced an octree database with an on-disk size of 4.1 GB.

We evaluated the rendering performance of the technique on a number of interactive inspection sequences. The qualitative performance of our adaptive GPU ray-caster is illustrated in an accompanying video available online [2]. Representative frames are shown in Fig. 2.7 and 2.8. Because of video recording constraints, the sequence is recorded using a window size of 640×480 pixels. In all recorded sequences, we used a 1 voxel/pixel accuracy to drive the adaptive renderer. As shown in the video, the system is fully interactive, and it is possible to translate, rotate, and scale the model as well as to change rendering mode, transfer functions, and isovalue parameters In Fig. 2.9, we show an example of our system working with a medical CT dataset [3]). The average frame rate of the DVR sequence varies between 12 Hz

[1] Source: Digital Morphology Project, CTLab and Texas Advanced Computing Center, University of Texas, Austin
[2] See the online video at: *http://vic.crs4.it/vic/cgi-bin/multimedia-page.cgi?id='145'*
[3] Source: Geneva University Hospital, Radiology Department

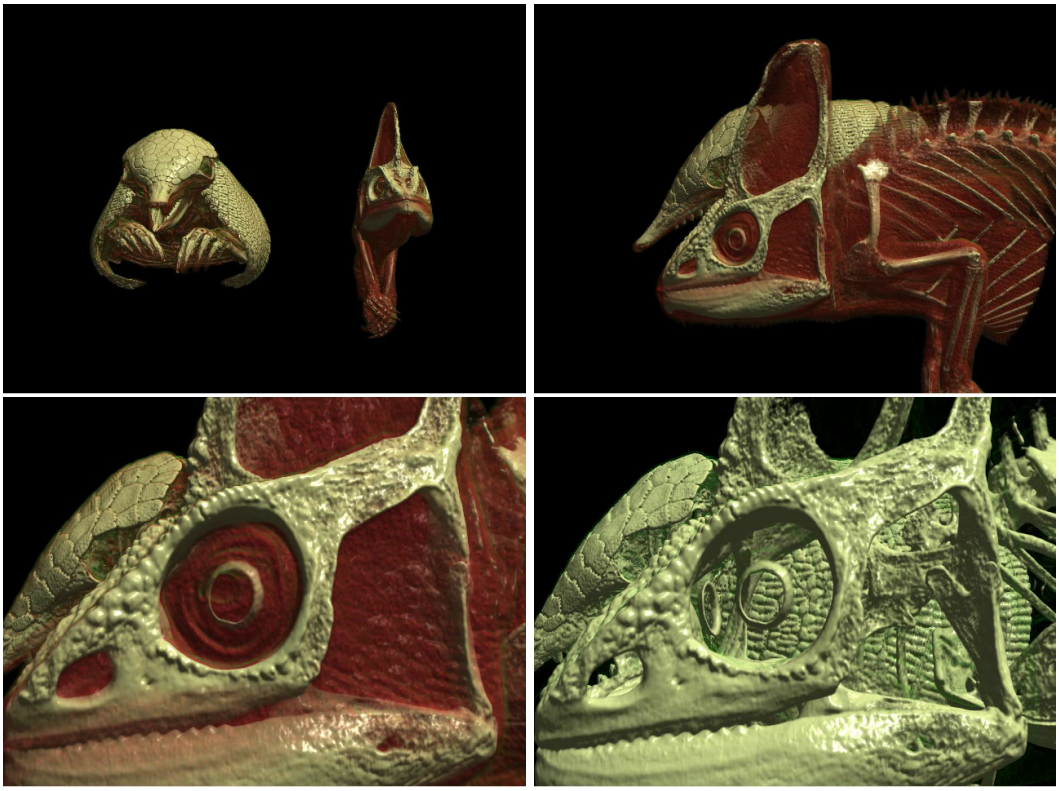


Figure 2.7: **Real-time inspection assigning semi-transparent materials.** These images show the rendering results for semi-transparent materials employing our multi-resolution out-of-core direct volume rendering approach on a test CT dataset. The overall volume has a resolution of $2048 \times 1024 \times 1080$ with 16 bits/sample.

and 30 Hz, with an average of 16 Hz. A few occasional frames have a delay of 200 ms, which correspond to cases in which many textures have to be updated in response to rapidly varying view conditions. These frame-rate jitters could be avoided by introducing a texture upload budget and stopping refinement when this budget is exceeded. In the case of isosurfaces, the frame rate is higher, 20 Hz on average, with peaks of up to 40 Hz. We repeated the same tests on a 1024×1024 window, and obtained an average slowdown of a factor of 3, roughly corresponding to the increase in number of pixels.

The higher performance of isosurface rendering is due to the simplicity of the inner accumulation loop, that has only to bracket the isovalue and accesses the gradient texture only once per fragment to shade the detected surface. By contrast, the direct volume rendering code requires an additional texture look-up for implementing the transfer function and accumulates more samples per fragment in the case of semitransparent materials. This latter fact is also reflected in the higher texture memory needs of semitransparent volume rendering, caused by the decreased effectiveness of visibility culling. During the entire inspection sequences, the resident set size of the application is maintained within the 600 MB of preallocated cache size by the out-of-core data management system. Both the isosurface and volume rendering sequences have a minimum texture memory occupation of about 475 octree bricks, corresponding to the first few frames with a view from a distance. However, the isosurface rendering sequence had an average memory occupation of 1560 bricks and a peak of 1820, while the direct volume rendering sequence had a average memory occupation of 1890 bricks and a peak of 2450.

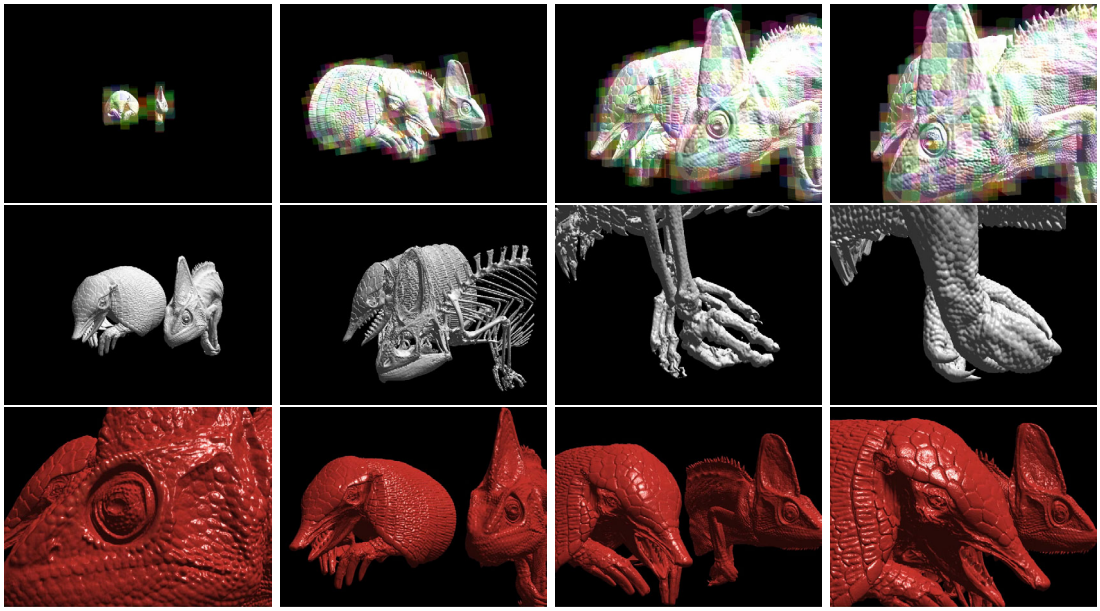


Figure 2.8: **Snapshots during a real-time inspection session.** These images, taken from the accompanying video [2], show successive instants of interactive exploration of the test CT dataset. The overall volume has a resolution of $2048 \times 1024 \times 1080$ with 16 bits/sample.

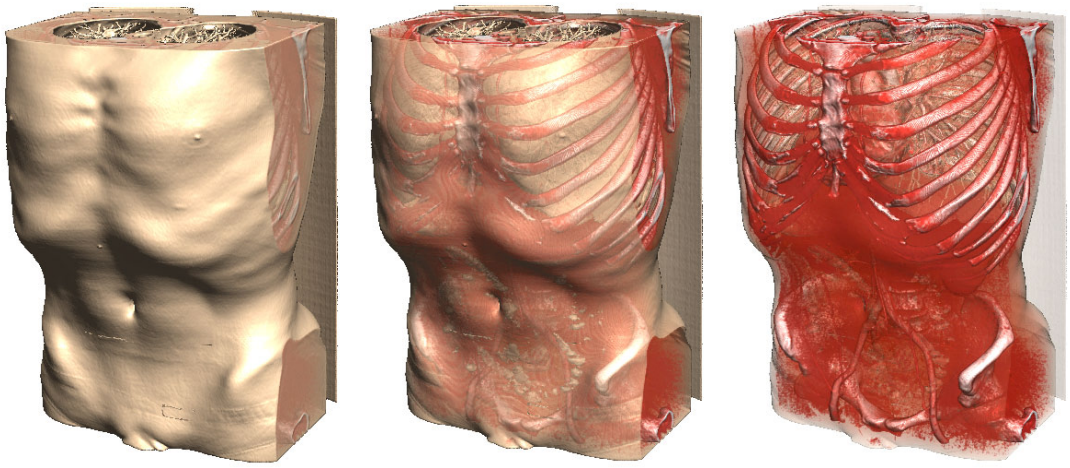
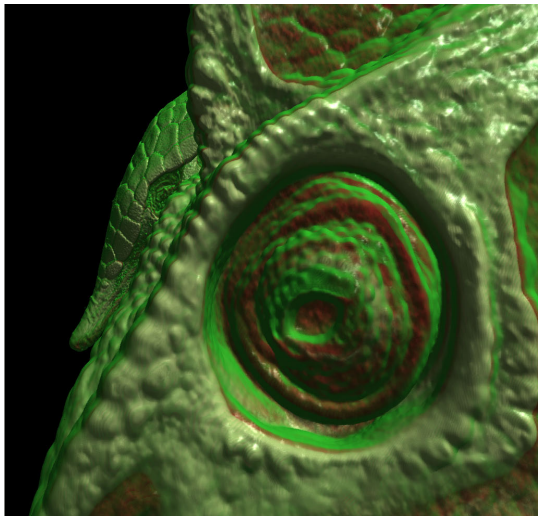
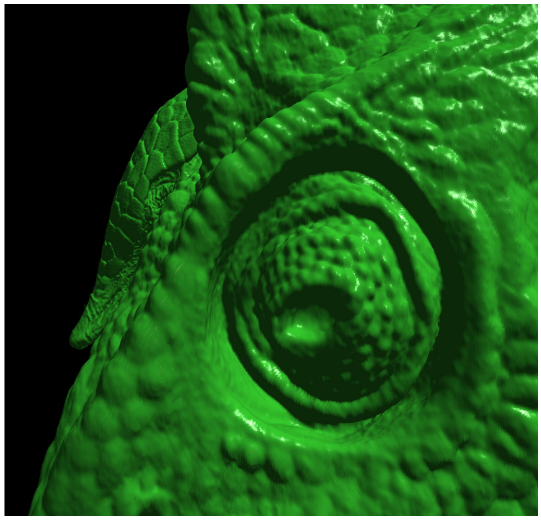


Figure 2.9: **Rendering results of our system working with medical CT datasets.** Results of our multi-resolution out-of-core rendering framework working with a $16bit/sample$ CT thorax study containing $512 \times 512 \times 743$ voxels. Different possible settings of the transfer function varying the opacity of the different materials are shown in this case.

The occlusion query mechanism has proved to be capable of reducing the size of the working set, especially when using isosurface rendering or transfer functions with moderate to high opacity. A simple illustration of the benefits of visibility feedback is given in Fig. 2.10, which shows two direct volume rendering images with different transfer functions, rendered on a 1024×1024 window. Visibility culling reduces the working set by about 50% in the semitransparent case, and by about 60% when surfaces get more opaque.



(a) 2505 blocks without occlusion culling, 1334 blocks with occlusion culling



(b) 2455 blocks without occlusion culling, 1038 blocks with occlusion culling

Figure 2.10: **Impact of the occlusion culling feedback.** Direct volume rendering images with different transfer functions, rendered on a 1024×1024 window. Visibility culling reduces the working set by about 50% in the semitransparent case, and by about 60% when surfaces get more opaque.

2.4 Conclusion

This chapter presented an adaptive out-of-core technique for rendering massive scalar datasets within a single-pass GPU ray casting framework. The method separates the adaptive incremental maintenance of the rendering working set, which is performed on the CPU, from the actual rendering, which is fully performed on the GPU by a stackless ray-caster that traverses a spatially indexed version of the current working set maintained in texture memory. Our results demonstrate that the resulting method is able to interactively explore gigavoxel datasets on a desktop PC. Besides optimizing and improving the proof-of-concept implementation, we plan to extend the presented approach in a number of ways. In particular, we are currently working on incorporating compression in the GPU representation to reduce GPU memory costs, as well as techniques to further reduce the working set through a more aggressive visibility culling based on tighter bounding volumes. We are also exploring ways to exploit the capability of our system to perform a full-volume ray tracing to produce higher quality images that incorporate more advanced shading effects.

2.5 Bibliographical Notes

Most of the contents of this chapter were taken from the paper [Gobb 08], where we introduced our novel single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. This was the first time that a volume rendering system was proposed combining GPU single-pass ray traversal and a multi-resolution out-of-core creation of and adaptive representation in the CPU. This approach had an important impact, and is cited in a number of follow-up works. The Gigavoxel approach by Crassin et al. [Cras 09] uses a very similar approach. Similarities between both works were pointed out and discussed in their paper. More precisely, their approach exploits multiple render targets to store a subset of the traversed nodes for each pixel and exploit spatio-temporal coherence trying

to avoid missing visible nodes. Using mipmapping, they address the aliasing to a large extent but increase the memory consumption and makes tree updates more challenging. We later improved on these methods using our GPGPU algorithm (see chapter 2). Hughes et al. [Hugh 09] presented another stackless technique, kd-jump, an approach for implicit kd-trees, which exploits the benefits of index-based node traversal, without incurring extra node visitation. Liu et al. [Liu 09] proposed the use of proxy spheres, instead of cubical bricks, in order to provide more control over the rendering process by introducing tighter ray segments for ray casting, while at the same time avoiding the introduction of some rendering artifacts. Lux and Froelich presented in [Lux 09] extended the ray casting framework for rendering multiple arbitrarily overlapping multi-resolution volume datasets. More recently, Pantaleoni et al. [Pant 10] described a novel architecture for precomputing sparse directional occlusion caches used for accelerating a fast cinematic lighting pipeline. As in our approach, duly cited in their work, they decompose their computation into a CPU-based LOD selection phase and a GPU-based rendering phase. Their system was used as a primary lighting technology in the movie Avatar.

Chapter 3

Taking Advantage of GPGPU Architectures

GPGPU architectures offer additional possibilities with respect to the classical graphics pipeline. These possibilities can be exploited to enhance rendering methods. This chapter describes a series of improvements on the previously presented multi-resolution volume rendering method (see chapter 2). The new rendering system is constructed around a configurable GPU ray casting kernel, exploiting the advanced characteristics of modern GPGPU architectures to achieve both flexibility and performance.

3.1 Introduction

Driven by the market demand for real-time, high-definition 3D graphics, the programmable Graphic Processor Unit or GPU has evolved into a highly parallel, multi-threaded, many-core processor with great computational power and very high memory bandwidth. The GPU architecture is specialized for intensive highly parallel computation, exactly what graphics rendering is about, and therefore designed such that more transistors are devoted to data processing rather than data caching and flow control.

The desired scenario when using the GPU presents a bigger ratio of arithmetic operations with respect to memory operations. Since the same program is executed for each data element, there is a lower requirement for sophisticated flow control, and because it is executed on many data elements and has high arithmetic intensity, the memory access latency can be hidden with calculations instead of big data caches.

The basis of data-parallel processing consists on mapping data elements to parallel processing threads. Nevertheless the GPU architecture and the provided programming APIs have not been always enough flexible to support this direct mapping between data and the processing cores. Some CPUs have also support for Single-instruction/Multiple-data (SIMD) operations that work in parallel on vectors. However, if a GPU is available, which is the norm today, then this reduces the workload on the CPU.

The GPUs have a parallel throughput architecture that emphasizes executing many concurrent threads slowly, rather than executing a single thread very quickly. This approach of solving general purpose problems on GPUs is known as GPGPU, where GPGPU stands for General-Purpose computation on Graphics Processing Units. The GPGPU platform we have specifically used is the computing engine of the NVIDIA® GPUs. CUDA™ is an acronym for Compute Unified Device Architecture and is a parallel computing architecture developed

by NVIDIA that is accessible to software developers through variants of industry standard programming languages.

In order to improve the GPU-accelerated multi-resolution volume rendering method presented in Chapter. 2, we propose a new CUDA-based ray-caster, which traverses an adaptively maintained working set of volume bricks organized in an octree. In the proposed GPGPU approach, the template specialization is exploited to flexibly customize ray traversal and compositing operations to achieve a variety of realistic and illustrative effects. Scatter-memory write operations within the kernel are exploited to feed visibility information from the rendering kernel back to the adaptive loader for implementing occlusion culling.

Rendering performance is improved by reducing the amount of pixels fully recomputed at each frame. For such purpose, we developed a technique for improving frame rates by recomputing only a fraction of the pixels at each frame and reconstructing full images using an adaptive spatio-temporal filter.

3.1.1 CUDA: The NVIDIA GPGPU Architecture

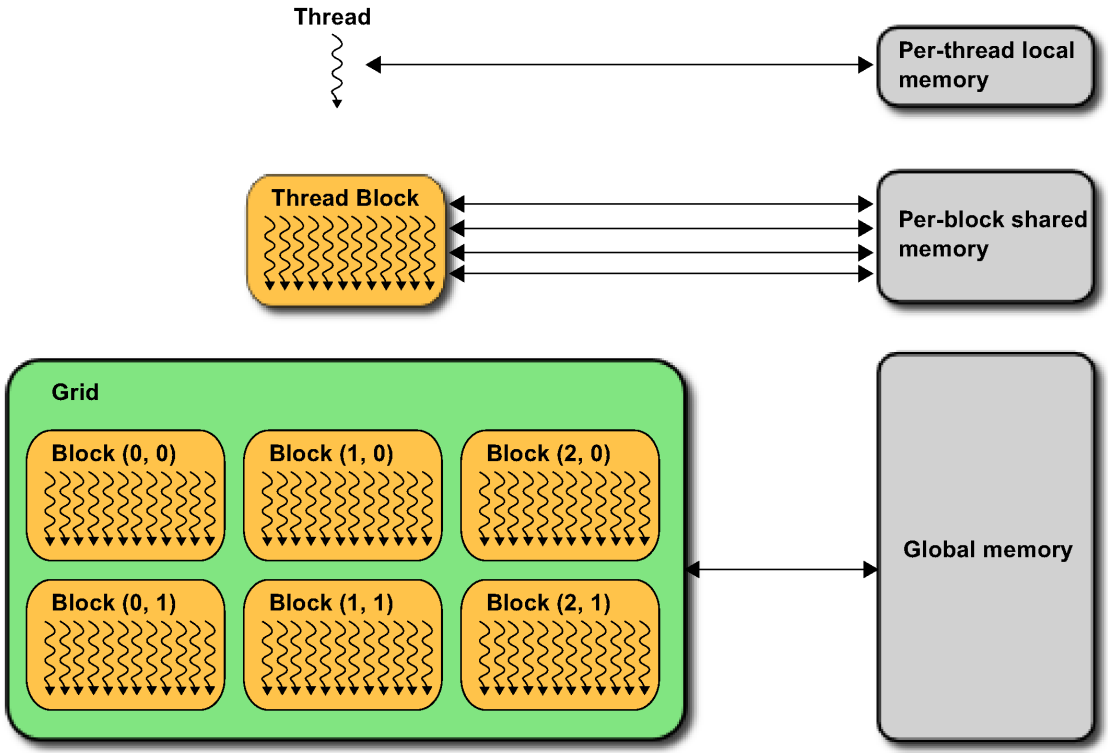


Figure 3.1: **The CUDA hierarchy of threads, blocks, and grids.** Hierarchy of threads, blocks, and grids, with corresponding per-thread private, per-block shared, and per-application global memory spaces.

CUDA is an acronym that stands for Compute Unified Device Architecture and is the parallel computing architecture developed by NVIDIA that is accessible to software developers through variants of industry standard programming languages. A CUDA program calls parallel kernels. A kernel executes in parallel across a set of parallel threads (See Fig. 3.1.1). The programmer or compiler organizes these threads in thread blocks and grids of thread blocks. The GPU instantiates a kernel program on a grid of parallel thread blocks. Each thread within

a thread block executes an instance of the kernel, and has a thread ID within its thread block, program counter, registers, per-thread private memory, inputs, and output results. A thread block is a set of concurrently executing threads that can cooperate among themselves through barrier synchronization and shared memory. A thread block has a block ID within its grid. A grid is an array of thread blocks that execute the same kernel, read inputs from global memory, write results to global memory, and synchronize between dependent kernel calls.

In the CUDA parallel programming model, each thread has a per-thread private memory space used for register spills, function calls, and C automatic array variables. Each thread block has a per-block shared memory space used for inter-thread communication, data sharing, and result sharing in parallel algorithms. Grids of thread blocks share results in global memory space after kernel-wide global synchronization.

The compute capability of a device defines the processing features supported by a particular graphics board model. For example the first NVIDIA 8800 series have the lower possible compute capability, 1.0. Current Fermi architecture instead, have compute capability 2.0. Differences are, for example, the support or not of atomic operations. An atomic function performs a read-modify-write atomic operation on one 32-bit or 64-bit word residing in global or shared memory. This feature can suppose important performance differences.

The NVIDIA CUDA framework has several advantages over traditional GPGPU architectures. It supports the scatter-read and scatter-write operations, so several threads can read from arbitrary addresses or write in specific memory of the GPU. It provides access to shared memory, that is, a fast shared memory region (up to 48KB in size in latest GPU models) that can be shared amongst threads. This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture look-ups. Furthermore, the memory allocation on GPUs and data transfer between CPUs and GPUs is performed in a more generic and efficient way. In addition, CUDA supports integer and bitwise operations, including integer texture look-ups.

3.2 A Flexible Ray Caster on GPGPU Architectures

In this section we present an improved version of our multi-resolution ray-caster presented in chapter 2. The new rendering system (see Fig. 3.2) is constructed around a flexible GPU ray casting kernel, exploiting the advanced characteristics of GPGPU to achieve both flexibility and performance. Ray traversal and compositing operations are flexibly customized using template specialization to achieve a variety of realistic and illustrative effects. The rendering kernel feeds back visibility information to the adaptive loader using scatter memory write operations. Furthermore, the application responsiveness is improved by recomputing only a fraction of the pixels at each frame and reconstructing full images using an adaptive spatio-temporal filter.

3.2.1 CUDA Stackless Octree Ray Caster

At run-time, an adaptive loader updates a view- and transfer function-dependent working set of bricks incrementally maintained on CPU and GPU memory by asynchronously fetching data from the out-of-core octree. The working set is maintained by an adaptive refinement method guided by suitably computed node priorities, determined using information fed back

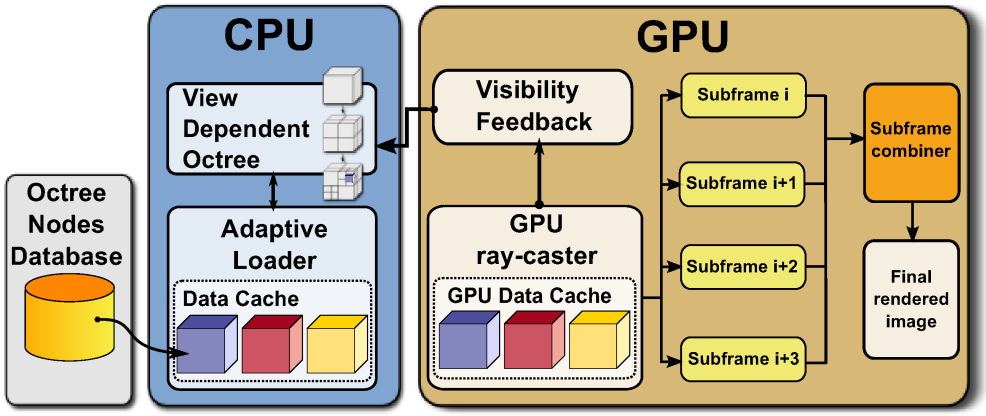


Figure 3.2: **Main structure of the improved CUDA ray caster framework.** Volume rendering architecture and structure of the new rendering subsystem integrating the adaptive frame reconstruction.

from the renderer. Following our previous work (see chapter 2, we assume that the adaptive loader maintains on GPU a cache of recently used volume bricks, stored in a 3D texture, and constructs at each frame a spatial index for the current working set in the form of an octree with neighbor pointers.

Our new octree encoding is more compact and efficient than the one presented in the previous chapter (see subsection 2.3.2.1, and consists in a single $6 \times N$ 2D texture with 32bit components (see Fig. 3.3). Texture columns index the N nodes of a linearized octree, while each row contains a different attribute. The first row contains spatial index nodes, with a 16bit index pointing to the node's parent, and a 16bit index pointing to the first of eight consecutive children (null for leafs). The second row contains the index of associated data in the current brick pool, or null for empty nodes. The third row contains the minimum and maximum values used for mapping the 8bit data values contained in the referenced bricks to the input domain. The fourth, fifth, and sixth rows contain the $(-x,+x),(-y,+y),(-z,+z)$ neighbor pointers. The cost of this representation is of 192bits/node, improving over the 256bits/node of Gobbetti et al. [Gobb 08], even though the storage of data ranges, parent pointers, and data pointers for all tree nodes, as they enable the ray-caster to perform high-quality filtering. The approach by Crassin et al. [Cras 09], which also supports multilevel access to the hierarchy, uses only 64bits/node (but without data ranges), but forces the usage of kd-restart, leading to many redundant traversal steps. The spatial index texture is traversed by a ray-caster implemented in CUDA, with a CUDA thread associated to each ray. Traversal starts by transforming the ray to volume coordinates and by clipping it to the volume extent. Empty and non-empty bricks are then enumerated in front-to-back order, adapting sampling density to brick resolution, and stopping as soon as possible. The brick enumeration strategy follows the stackless approach presented in chapter 2. Each time a brick is traversed, data is accumulated according to the current rendering mode (see subsection 6.2), and the position along the ray upon brick exit is updated to the entry point of the next visited node, which is found by following neighbor pointers.

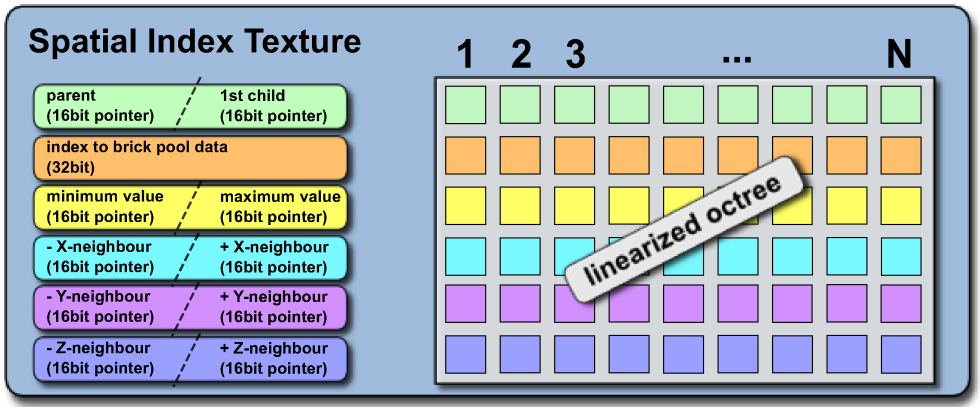


Figure 3.3: **The CUDA version of our new encoding for the spatial index structure.** Single $6 \times N$ 2D texture with 32bit components which represents the new linearized octree encoding, being more compact and efficient than our previous approach.

3.2.2 Flexible Traversal and Compositing

Our new renderer employs scalar transfer functions to associate optical properties to scalar values. The renderer works with extinction weighted colors, and we thus have a transfer function $\tau(s) \in [0, \infty[$ for the extinction coefficient and a transfer function $c(s)$ for the color, that has to be multiplied by $\tau(s)$ to yield an actual color intensity $\tilde{k}(s) = \tau(s)c(s)$ for a given scalar value s. In addition, we have a transfer function $\iota(s) \in]0, 1[$ for the importance of scalar value s, and $\nu(s) \in]0, \infty[$ for the refraction index. In order to deal with high frequency transfer functions, we use a preintegration based technique. In our approach, each transfer function is stored in a $2 \times N$ 2D texture. The first row of the texture contains a look-up table for a function $f(s)$, while the second row stores a look-up table for the function $F(s) = \int_0^s f(s)\,ds$, f being one of the functions $\tau, \tilde{k}, \iota, \nu$. We actually maintain a total of three textures: a 4-components one for $\tilde{k}(s), \tau(s)$, and two 1-component ones for $\iota(s)$ and $n(s)$.

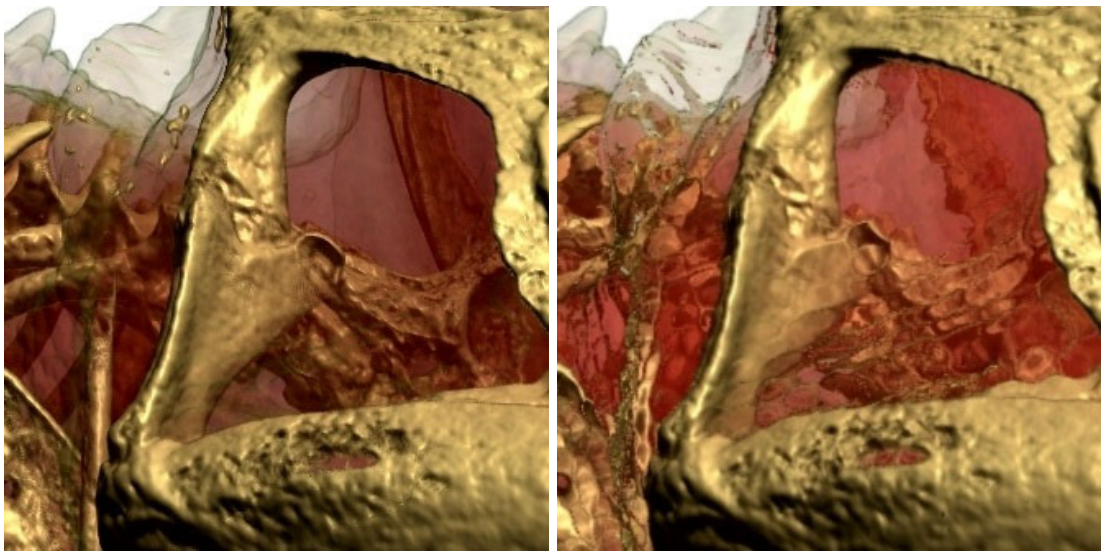


Figure 3.4: **Screenshots showing the continuous refraction effect.** With refraction enabled, ray direction is changed at each integration step according to Snell's law, and fine details in transparent objects become visible. The view-dependent effect is particularly evident when seen on the light-field display.

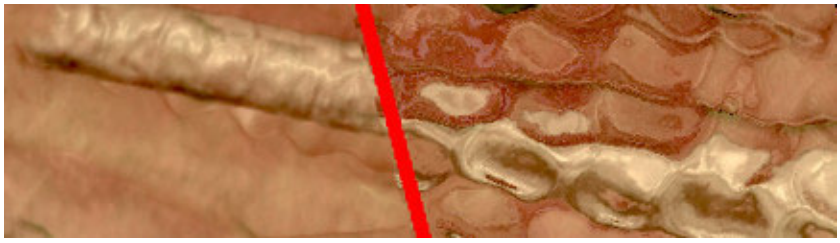


Figure 3.5: **Detail comparison showing the refraction effect.** With refraction enabled (right), ray direction is changed at each integration step according to Snell's law, and fine details in transparent objects become visible. The view-dependent effect is evident when seen on the light-field display.

The final color for each pixel is computed by performing compositing operations while following a view ray $\mathbf{r} = \mathbf{O} + t\mathbf{v}$. At each integration step i, we consider a segment of length d_i along $\mathbf{r}$, sample the scalar texture at the end of the segment to retrieve a scalar $s_i = s(t_i)$, and compute each of the attributes f_i we require for compositing with the following equation:

$$f_i = \begin{cases} \frac{F(s_i)-F(s_{i-1})}{s_i-s_{i-1}} & \text{if } \|s_i - s_{i-1}\| > \epsilon \\ f(\frac{s_{i-1}+s_i}{2}) & \text{otherwise} \end{cases} \tag{3.1}$$

where s_{i-1} is the scalar value at the beginning of the segment, which is recorded from a previous step together with $F(s_{i-1})$, and $0 < \epsilon \ll 1$, e.g., $\epsilon = 10^{-3}$. The overall method is numerically stable and the main assumption behind it is that the scalar data s along the small segment is well approximated by a linear interpolation between s_{i-1} and s_i. Once the average attributes for the segment have been computed, the extinction weighted color $\tilde{k}_i$ and the extinction τ_i are transformed into opacity weighted colors $\tilde{c}_i$ and opacity α_i using the relation $\alpha_i = 1 - e^{-\tau_i d_i}$. We then compute the gradient ∇s_i using central differences at $t = \frac{t_{i-1}+t_i}{2}$. As for GigaVoxels[Cras 09], we can exploit multilevel access to the hierarchy (see Fig. 3.7) to sample both s_i and ∇s_i at multiple levels in the octree hierarchy, and blend them using quadrilinear interpolation for continuous filtering. In addition, we employ the multiple values and gradients for multilevel shading effects, such as exaggerated shading [Cign 05] and normal unsharp masking [Rusi 06].

All the computed attributes, the current ray position and direction, and the material and light properties (ambient, diffuse, and specular factors, plus shininess exponent) are then made available to "enhancer" objects, which, much as shaders in a rasterization pipeline, can freely modify them to implement a variety of effects (see section 6.4 for examples). All the active enhancers are chained together, each receiving the result of the prior one as input. After the enhancers have produced their final values, the actual shading and compositing steps can be performed. The shading step simply takes colors and gradients and computes a shaded color. The accumulation and compositing scheme is similar to MIDA [Bruc 09], even though in our case we accumulate importance rather than intensity, leading to a *Maximum Importance Difference Accumulation* approach. First, we compute the current importance difference $\delta_i = \max(\iota_i - \iota_{max_i}, 0)$, where ι_{max_i} is the current maximum importance value along the ray. We

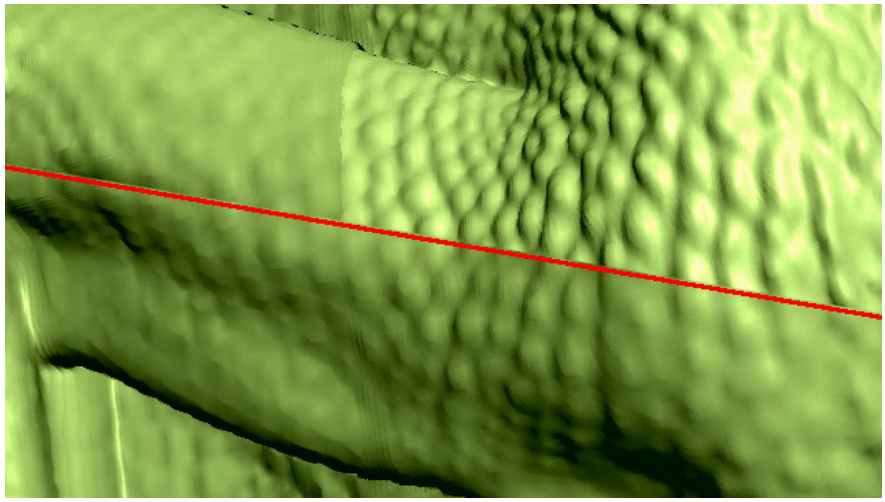


Figure 3.6: **Multilevel rendering.** Using a single LOD per brick may lead to discontinuities at brick boundaries, while quadrilinear interpolation ensures continuous rendering.

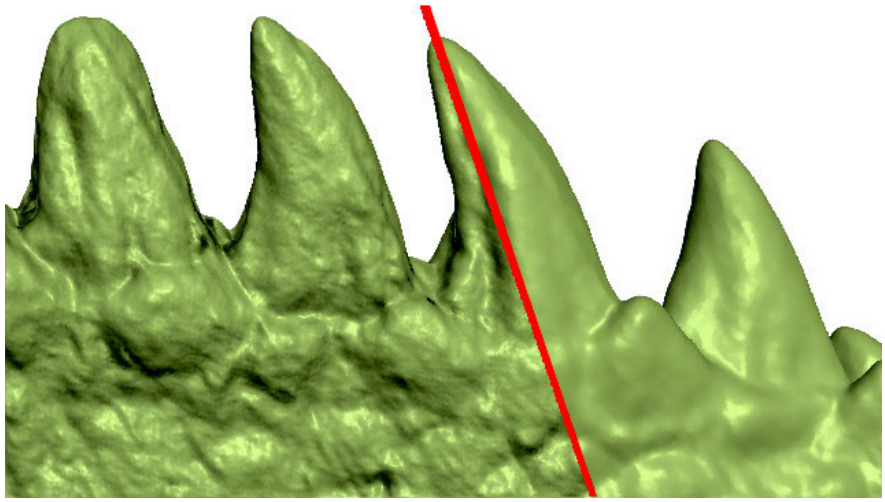


Figure 3.7: **Real-time volume unsharp masking.** Tiny details can be enhanced at run time by unsharp masking of the gradient field.

then compute $\beta_i = 1-\gamma\delta_i$, and perform the accumulation according to the following equation:

$$c_i^* = c_{i-1}^*\beta_i + (1-\beta_i\alpha_{i-1}^*)\tilde{c}_i \tag{3.2}$$

$$\alpha_i^* = \alpha_{i-1}^*\beta_i + (1-\beta_i\alpha_{i-1}^*)\alpha_i \tag{3.3}$$

where c^* and α^* denote accumulated colors and opacity, and $\gamma \in]0,1[$ is a parameter that allows us to smoothly interpolate from plain volume rendering (with $\gamma = 0$) to full maximum importance difference accumulation (with $\gamma = 1$). The effect of the above equation with $\gamma > 0$ is to have more important content shine through previously accumulated values. Equation 3.2 only differs from standard DVR compositing by the additional weighting with β_i of previously accumulated colors and opacity. Higher prominence is assigned to local maxima of the importance function, letting more important content shine through less relevant material (Fig. 3.8).

In this framework, particular care has to be taken to make the overall hierarchical bricking

scheme cooperate with the above preintegration based approach. In particular, upon entry and exit of empty bricks we perform an accumulation step with the default "empty" values (black color, null importance and opacity, unit refraction index). Upon entering a non-empty brick, the first accumulation step uses a distance d_{IN} corresponding to the distance between the previous and the current sampling position, then proceeds with a number of steps of constant length d equal to the local voxel size. If refraction is not enabled, the ray direction is maintained constant, and the number of steps performed within a brick is computed upon brick entry by intersecting the ray with the brick's bounding volume. If refraction is enabled (see Fig. 3.4 and 3.5), direction is changed after each step according to Snell's law, and the brick exit condition is checked at each step. In this approach, the management of empty bricks is consistent with the management of empty voxels within non-empty bricks only if enhancers are not active for empty voxels. We enforce this in our code by skipping enhancement if $\alpha < \epsilon$.

Moreover, since Equation 3.2 does not guarantee monotonously increasing opacity, early ray termination criteria must be revised. We thus estimate the maximum future opacity delta with $\Delta\alpha_i^* = 1 - (1 - \gamma(1 - \iota_{\max_i}))\alpha_i^*$, and stop ray traversal when $\Delta\alpha_i^* < \epsilon$. Note that if $\gamma = 0$, this test reverts to the usual accumulated opacity test. For performance reasons, this test is performed after traversing each non-empty brick, and not after each accumulation test.

3.2.3 Putting It All Together: Flexibility and Performance

The above flexible rendering structure requires particular care in implementation to ensure reasonable performance. In particular, depending on the configuration, the rendering kernel has to switch between refraction or straight ray propagation, and to apply a chain of enhancing filters, each of variable complexity, to the current sampled state. Writing a single rendering kernel with run-time switches would lead to a bulky code with poor performance and lacking extensibility. In our framework, we exploit the advanced programming capabilities of the CUDA framework to deal with the problem. In particular, C++ templates provide us with abstraction and flexibility while retaining the opportunity for optimal performance and compiler optimization. This implies that with a proper design we can achieve fine abstraction granularity, and still deliver the performance of hand-tuned implementations.

In our design, the rendering kernel is a generic function whose template parameters define the stepping behavior within non-empty bricks and a series of up to three chained enhancing behaviors. A multiple dispatching code run at each kernel launch, whose structure is illustrated in Listing 3.2.3, transforms all dynamic types into static types before kernel invocation. In the actual code, preprocessor macros are used to avoid replication of the multiple dispatching code, increasing code extensibility.

An enhancer is seen as a class providing three device functions of the current accumulated and sampled state that define the behavior at ray start, at each sampling step, and at ray termination. These device functions, called *enhance_initialize*, *enhance*, and *enhance_finalize*, operate on the rendering state, and also have access to a specific enhancer's constant and variable state. The constant state is a structure containing the current enhancer parameters, which is stored in device constant memory and shared among all threads. The enhancer variable state, instead, is a per-thread structure stored in device shared memory, and initialized by the kernel before ray traversal using the *enhance_initialize* entry point. Storing this data in shared memory, as well as other kernel long-lived variables (ray origin, direction, and accumulated state), reduces the number of registers required by a given thread invocation,

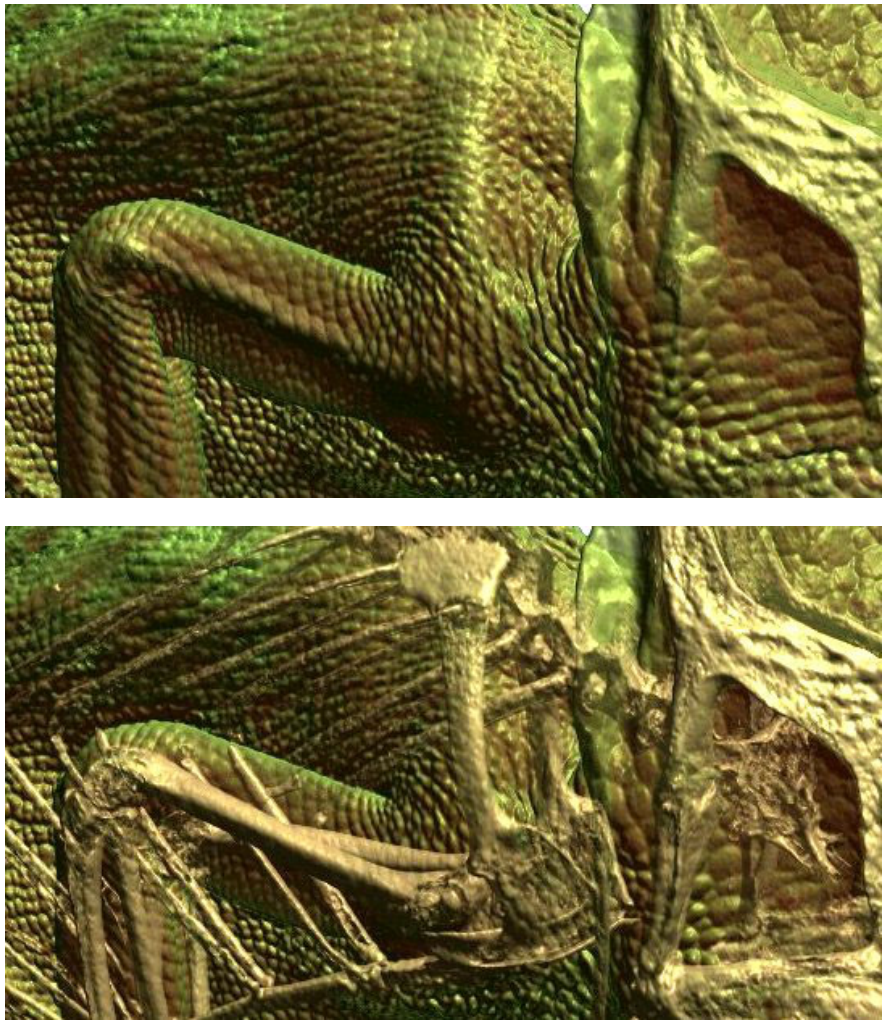


Figure 3.8: **Maximum Importance Difference Accumulation (MImDA) scheme.** In the image on top, all materials are given the same importance. In the image below, bone is given a higher importance and thus shines through previously accumulated material layers.

increasing device occupancy. In our design, a specific kernel is generated (and optimized by the compiler) for each of the possible combinations of steppers and enhancers. A special *null_enhancer* class, with empty constant and variable data and no operations performed in device functions allows us to implement a pass-through behavior, when not all enhancers are present.

Generating a different specialization for each possible combination of enhancers obviously leads to combinatorial explosion of generated implementations. This problem is common to most C++ template libraries. We thus keep the number of concurrently active enhancers low (2-3 in our implementations), and parametrize enhancers to reduce the number of classes on which to dispatch. Since the ray-casting kernel code is small, the problems are fully manageable. A fully optimized compilation of a system with four different enhancer types takes less than one minute when allowing three concurrently active enhancers.

```
// Generic CUDA raycasting kernel entry point
template <typename E1, typename E2, typename STEPPER>
__global__ void trace_kernel(uint *out_buffer, uchar* visibility_buffer) {
  accumulated_state_t state = ...;
  __shared__ typename E1::vdata_t e1_vdata[NTHREADS];
  __shared__ typename E2::vdata_t e2_vdata[NTHREADS];
  enhancer_initialize(state, *((E1::cdata_t*)(&cgpu_e1_cdata)), e1_vdata[TIDX]);
  enhancer_initialize(state, *((E2::cdata_t*)(&cgpu_e2_cdata)), e2_vdata[TIDX]);
  tracing loop ... {
    ...
    enhance(state, sample, *((E1::cdata_t)(&cgpu_e1_cdata)), e1_vdata[TIDX]);
    enhance(state, sample, *((E2::cdata_t*)(&cgpu_e2_cdata)), e2_vdata[TIDX]);
    ...
  }
  enhancer_finalize(state, *((E1::cdata_t*)(&cgpu_e1_cdata)), e1_vdata[TIDX]);
  enhancer_finalize(state, *((E2::cdata_t*)(&cgpu_e2_cdata)), e2_vdata[TIDX]);
  out_buffer[pixel_idx] = current_state.accumulated_color;
}

// Third level dispatching and kernel launching
template <typename E1, typename E2>
__host__ void render_level3(E1 e1, E2 e2) {
  cudaMemcpyToSymbol(cgpu_e1_cdata, e1.cdata(), sizeof(...));
  cudaMemcpyToSymbol(cgpu_e2_cdata, e2.cdata(), sizeof(...));
  if (refraction) {
    trace_kernel<E1,E2,refract_stepper><<NG,NB>>(out_buffer, visibility_buffer);
  } else {
    trace_kernel<E1,E2,basic_stepper><<NG,NB>>(out_buffer, visibility_buffer);
  }
}
// Second level dispatching
template <typename E1>
__host__ void render_level2(E1 e1, enhancer* e2) {
  if (!e2) { render_level3(e1, null_enhancer()); } else {
    switch (e2->tag()) {
    case TAG1: render_level3(e1, *((type1*)(e2))); break;
    case TAG2: ...
    }
  }
}
// Renderer entry point
__host__ void render(enhancer* e1, enhancer* e2) {
  if (!e1) { render_level2(null_enhancer(), e2); } else {
    switch (e1->tag()) {
    case TAG1: render_level2(*((type1*)(e1)), e2); break;
    case TAG2: ...
    }
  }
}
```

Figure 3.9: **Generic CUDA Kernel execution framework.** A multiple dispatching approach converts dynamic types to generic parameters.

3.2.4 Visibility-aware Adaptive Loader

The adaptive loader maintains in-core a view- and transfer-function- dependent cut of the out-of-core octree and uses it to update the GPU cache and spatial index. The basic principle of the method is to update at each frame the visibility status of the nodes in the graph based on rendering feedback, and, during the refinement cycle, only refine nodes that were marked as visible during the previous frame and are considered non-empty according to the current transfer function. Under this approach, the available GPU texture slots will be used mainly for nodes situated in the visible part of the model, and load requests will not even be posted for invisible ones (see Fig. 3.10).

The update procedure goes through different phases. First, the visibility feedback array, computed at the previous frame, is copied from the device into the host and the visibility status of the in-core nodes is updated accordingly. Since the array only stores data for visited leaves, visibility data is then pulled up to inner nodes by recursively combining visibility information of child nodes. Then, the graph is visited top-down, stopping each time a node is accurate

enough, is empty according to the current transfer function, or is marked as not visible in previous frame. As in [Scha 05], summed-area tables of the transfer function opacity are used to determine if a brick is empty by taking the difference of the table entries for the minimum and maximum block values stored with the node. On devices with CUDA capability 1.1 and above, a node is considered visible if traversed by more than a prescribed small number of rays. On older devices, a node is marked visible if the corresponding entry in the visibility array is non-zero. When traversal stops, eventually present sub-trees in the octree cut are considered over-refined and are thus removed. When leafs needing further refinement are reached, they are inserted in a refinement queue ordered by decreasing accuracy error. We then proceed by refining nodes present in the refinement queue, stopping when all nodes are considered adequately refined, no data is currently available in-core to perform a refinement, or no more space is available in the GPU cache to contain a further subdivision. In order to hide out-of-core data access latency, all data access requests are performed asynchronously by a separate thread, and refinement for a given node continues only if data is immediately available. Once refinement is completed, the graph is visited bottom-up, recursively merging all nodes that contain only empty leafs. At the end of the update procedure, all nodes in the current working set are present in the GPU cache. The structure of the in-core octree is encoded into the spatial index texture and transferred to the GPU before launching the CUDA kernel performing ray-casting. The update process is extremely fast, given the coarse granularity of the octree of bricks and the fact that visibility information is available at negligible cost as a renderer by-product.

3.3 Adaptive Frame Reconstruction

Rendering performance can be further improved by taking into account spatio-temporal coherence. In this section, we reduce the amount of pixels generated at each frame, and reconstruct full images based on samples acquired at different times. Our aim is to increase responsiveness, while not compromising the ability to render static images at full quality. Our method is purely image-based and works without making assumptions about the rendered scene and rendering process. No cooperation from the renderer is required, besides the ability of computing images. The basic idea is to use an adaptive spatio-temporal filter, which strives to provide spatially detailed imagery where the viewed scene is static, and low-latency (if blurred) imagery where it is dynamic. In order to fully exploit parallel rendering performance and to avoid destroying coherence among neighboring rays, we do not perform adaptive sampling, but group pixel batches in sub-frames aligned on a quincunx lattice. We explicitly avoided using more elaborate sparse/adaptive schemes because we want to reduce CPU/GPU communication. Moreover, it is generally better to trace many groups of very nearby rays to reduce thread divergence than selecting fewer sparse rays to trace. Screen pixels are thus assigned to N sub-frames (4 in our current implementation) made of interleaved pixels on x and y axes (see Fig. 3.3), which are sequentially rendered and buffered. At each frame presented to the viewer, only one sub-frame is re-rendered, and frame reconstruction is performed for presenting the image. The reconstruction process starts by copying the pixels of the just rendered sub-frame to the appropriate locations in the final image. All other pixel values are then generated by a spatio-temporal filter. The basic idea is to compute a new not-rendered pixel Q'_t by combining its previous version in time, Q_{t-1}, with its four cross

Figure 3.10: **Impact of the visibility culling.** Direct volume rendering images with different transfer functions, rendered on a 1024×576 window. Visibility culling reduces the working set from 1731 bricks to 1035 bricks in a almost opaque case, and from 1984 bricks to 1789 bricks when surfaces get more transparent.

neighbors in space, A_t, B_t, C_t, D_t (see Fig. 3.3). We first compute a spatial approximation $Q_t^{spatial}$ by averaging the two opposite neighbors with lower color difference (i.e., A_t, C_t or B_t, D_t). Then, we estimate the temporal change ΔQ of the pixel by taking the maximum absolute color difference among (A_t, A_{t-1}), ..., (D_t, D_{t-1}). The new pixel color is then computed by $Q'_t = Q_{t-1} + \text{smoothstep}(\Delta Q, \Delta_\text{l}, \Delta_\text{h})(Q_t^{spatial} - Q_{t-1})$, where the smoothstep function smoothly transitions from zero to one as ΔQ varies between Δ_l and Δ_h. The process is then repeated for all other missing pixels, by suitably shifting and rotating the reconstruction grid when applying the same reconstruction process. With such an approach, where the image is dynamic, more recent samples are emphasized, resulting in up-to-date, but possibly slightly blurred images. Where the image is more static, older samples dominate reconstruction, leading to sharp images. In static image areas, the method is guaranteed to converge to the same

image that would have been computed by performing a full-resolution ray-casting. We have purposely decided not to include motion compensation or re-projection steps, since we prefer increased blur in motion areas over artifacts when dealing with semi-transparent objects.

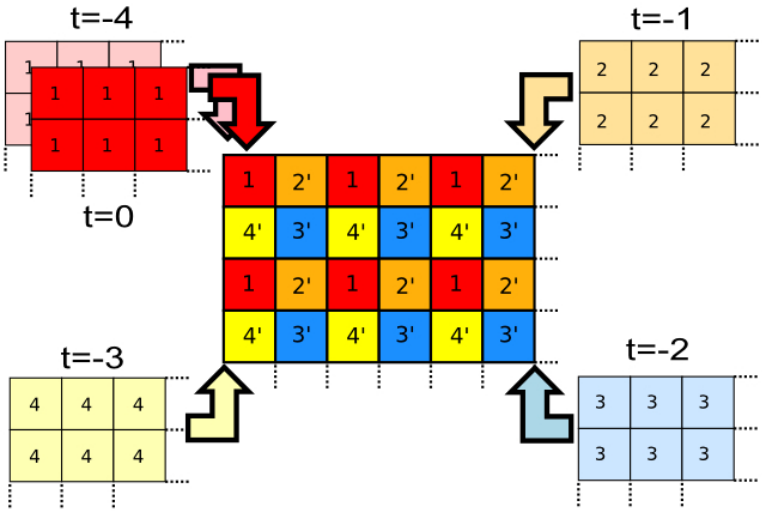


Figure 3.11: **Frame-reconstruction scheme using the quincunx pattern.** Sub-frame pixels are arranged according to a quincunx pattern. The last sub-frame, as well as the four previously rendered ones, contribute to the full image.

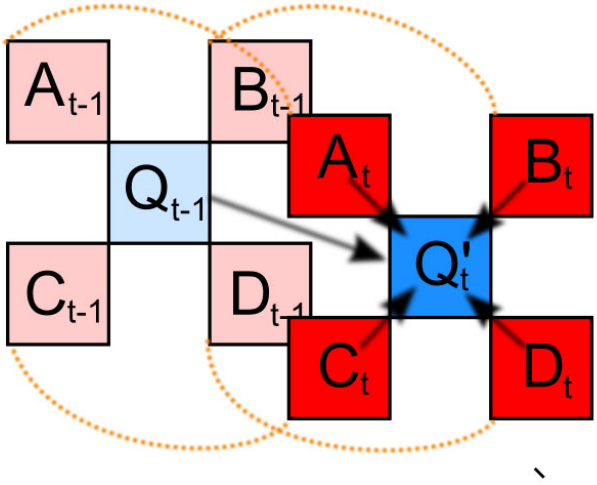


Figure 3.12: **The spatio-temporal filtering scheme.** A spatio-temporal filter reconstructs a full image considering previous and current values of a pixel and its neighbours.

The frame reconstructor is fully implemented in a single CUDA kernel. Each thread reconstructs a quad of 2×2 pixels using a three phase process. First, threads within the same thread block cooperate to load into shared memory the tile of present and past pixel values required for reconstruction, and copy to the output buffer the single up-to-date pixel associated to them. Then, each thread reconstructs the diagonal neighbor of that pixel. Finally, the horizontal and vertical neighbors are reconstructed. A *syncthreads()* barrier at the end of each phase ensures that the four neighbors of the next reconstructed pixels are up-to-date.

3.4 Implementation and Results

An experimental system has been implemented on Linux using OpenGL and NVIDIA CUDA 2.3. The out-of-core octree structure is implemented on top of Berkeley DB, with LZO compression applied to data nodes. The rendering kernel has been embedded directly in a single-process application, using a separate thread for asynchronous data loading. We have tested

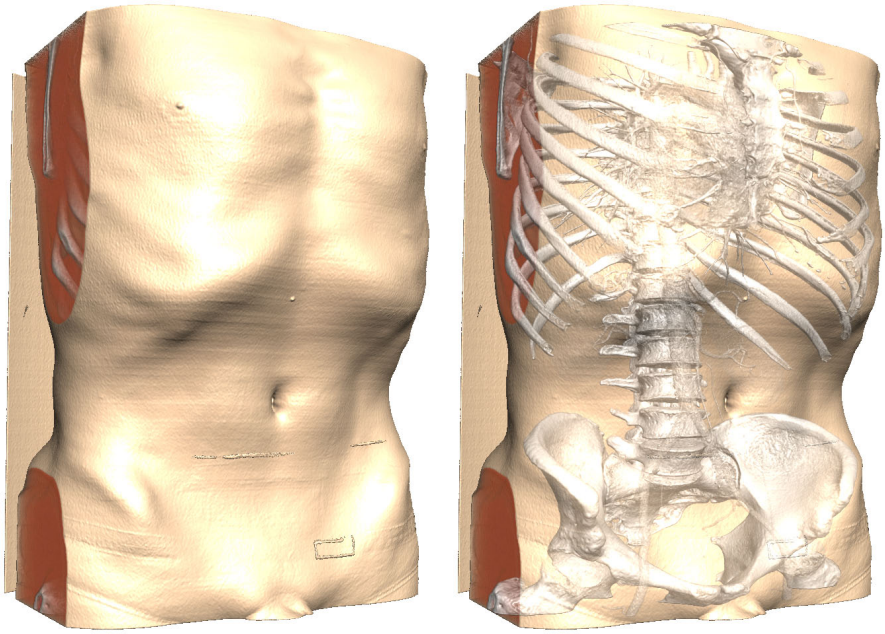


Figure 3.13: **Effect of the MImDA accumulation scheme demonstrated with a medical dataset.** In these images, we show the difference between using standard DVR accumulation and employing the proposed MImDA accumulation scheme, the effect demonstrate to improve the understanding of structures which naturally would be hidden.

our system with a variety of high resolution models and settings. In this section, we discuss the results obtained with the inspection of two 16bit/sample micro-CT datasets: a Veiled Chameleon ($1024 \times 1024 \times 1080$) and a Pichi Armadillo ($1024 \times 1024 \times 999$) specimens [1]. In order to test the system with extremely large and demanding datasets, the original data has been artificially zoomed by $4\times$ in a preprocessing step (i.e., to $\approx 4K^3$ voxels) by tricubic interpolation plus procedurally added detail. The construction of the octree from source data was performed using octree bricks of 32^3, instructing the preprocessor to use the compressed 8bit format and to filter out data with a value lower than 6400, in order to discard most of the noisy empty space. For the chameleon, data preprocessing took 12.3hours on a Linux PC 2.4GHz CPU and produced a 15GB octree database starting from an uncompressed source size of 135GB. The RMS error was 14.6, and the AMAX error was 116. The armadillo dataset produced similar results (13.8GB octree database, $RMS = 15.8$, $AMAX = 122$).

We evaluated single PC rendering performance on a number of interactive inspection sequences using the 4K models and measuring actual frame rates (i.e., not only raw rendering times, but frame-to-frame times). We report here on the results obtained when rendering on a PC with NVIDIA GTX 280 using a window size of 1024×576 pixels and 1voxel/pixel accuracy to drive the adaptive renderer. When using the moderately opaque transfer function

[1] Source: Digital Morphology Project, the CTLab and the Texas Advanced Computing Center, University of Texas, Austin

of the top image in Fig. 3.10, the frame rate of typical inspection sequences of the chameleon dataset varies between 23Hz for extreme close-up views to over 60Hz for overall views. Making the model highly transparent, as in the bottom of Fig. 3.10, reduces the performance to $11\text{Hz} - 39\text{Hz}$. Adding refraction leads to rendering rates of $7\text{Hz} - 32\text{Hz}$. Using interactive tools, which activate enhancers in the CUDA ray-caster, decreases the performance by about 10%. Interactive rates are thus guaranteed even in the most demanding situations. Part of the performance is due to the frame reconstruction approach, which guarantees a speed-up of $2.15\times - 3.3\times$ depending on the situation. The speed-up is sub-linear, since we reduce the number of traced pixels by 4. The additional overhead is mostly due to the reduced coherence of rays traced from a sparser pixel grid, leading to increased divergence among threads traversing the spatial index structures.

3.5 Conclusion

We have described in this chapter a flexible multi-resolution volume rendering system implemented on a GPGPU architecture that introduced a series of improvements on the previously presented multi-resolution volume rendering method (see chapter 2). The new ray-caster is constructed around a flexible CUDA rendering kernel, which is also valid in more standard visualization settings. Template specialization was exploited to flexibly customize ray traversal and compositing operations to achieve a variety of realistic and illustrative effects. Scatter memory write operations within the kernel were exploited to feed visibility information from the rendering kernel back to the adaptive loader for implementing occlusion culling. Finally, this chapter described a technique for improving frame rates by recomputing only a fraction of the pixels at each frame and reconstructing full images using an adaptive spatio-temporal filter.

3.6 Bibliographical Notes

This chapter expanded the content of the article [Igle 10], where we presented an adaptive out-of-core rendering engine based on CUDA containing a number of improvements over previous state-of-the-art GPU volume renderers [Gobb 08, Cras 09]. Notable improvements include a more efficient handling of visibility information and a more flexible compositing pipe-line.

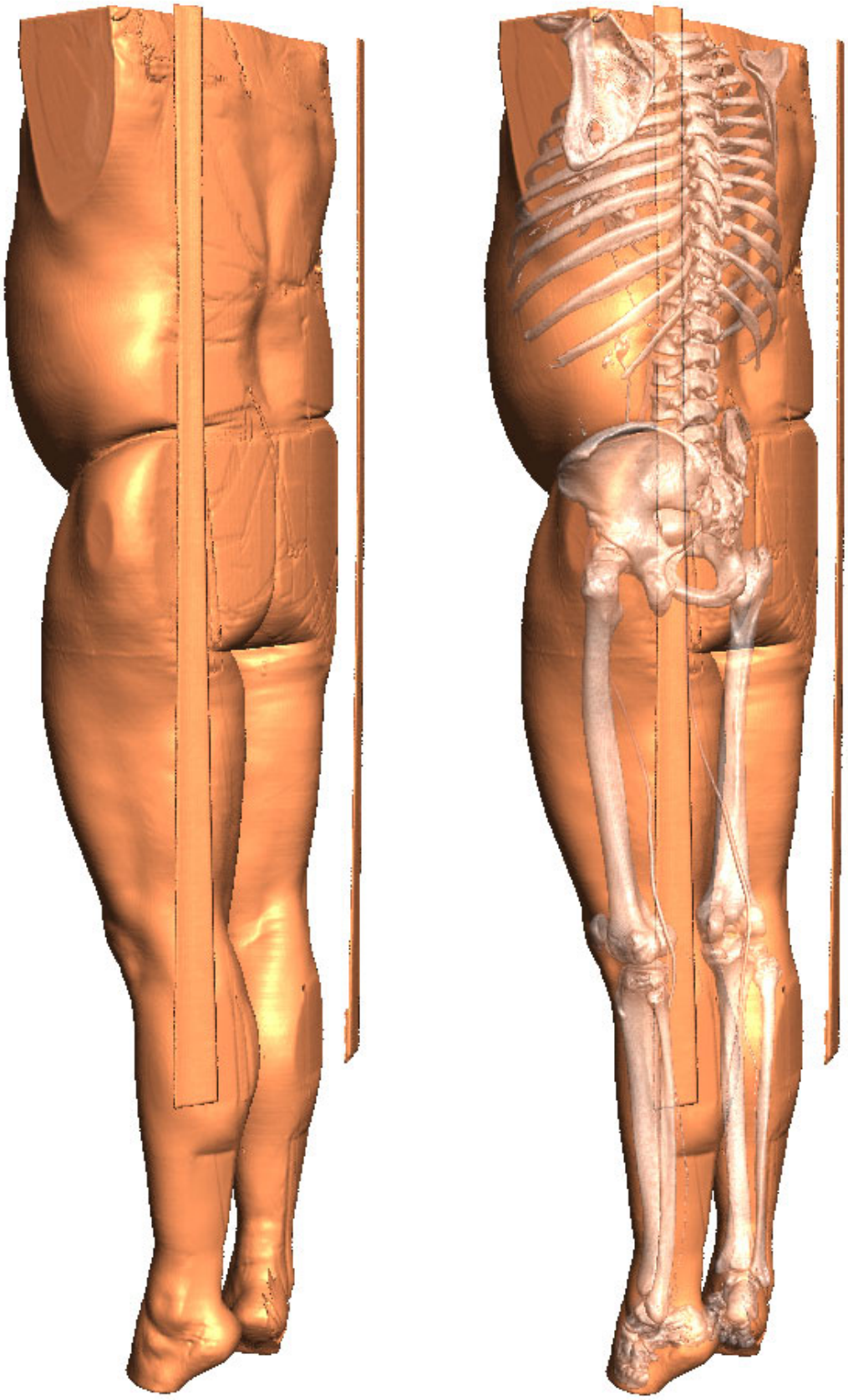


Figure 3.14: **Effect of the MImDA accumulation scheme with a full-body dataset.** The difference between using standard DVR accumulation and employing the proposed MImDA accumulation scheme demonstrate its usefullness for improving the understanding of inner structures which naturally would be hidden.

Chapter 4

Handling Discontinuous Datasets

The most common representation of volumetric models is a regular grid of cubical voxels with one value each, from which a smooth scalar field is reconstructed. However, common real-world situations include cases in which volumes represent physical objects with well defined boundaries separating different materials, giving rise to models with quasi-impulsive gradient fields. This chapter introduces a new volumetric primitive, named *split-voxel*, to handle such datasets containing sharp boundaries and discontinuities. We show how to convert a standard mono-resolution representation into an out-of-core multi-resolution structure, both for labeled and continuous scalar volumes using the *split-voxel* primitive. Furthermore, this chapter shows how to exploit such representation in a simple and efficient ray-casting accumulation scheme to interactively explore the resulting models using a multi-resolution GPU ray casting engine.

4.1 Introduction

The most common representation of volumetric models is a regular grid of cubical voxels with one value each, from which a smooth scalar field is reconstructed. This regularity enables efficient rendering, but is not well adapted to all data distributions. Common real-world situations include cases in which volumes represent physical objects with well defined boundaries separating different regions, giving rise to models with quasi-impulsive gradient fields. Boundary regions are prominently visible in volume rendered images, and the sampling nature of the voxelized representation can lead to aliasing, since voxel size restricts size and location of rendered details.

In order to overcome these limitations, we introduce a volumetric primitive, that we call *split-voxel*, which replaces blocks of N^3 voxels by one single voxel that is split by a feature plane into two regions with constant values. The feature plane provides a linear approximation to the strongest value discontinuity in the block, while the two values represent medians or averages which are not blurred over the discontinuity. This description is exploited in a multi-resolution GPU ray-casting framework able to handle large out-of-core datasets, and is used to represent both leaves and inner levels. Specifically, the main contributions of this approach are the following:

- The *split-voxel* volumetric primitive that encodes scalar data together with edge detection information;

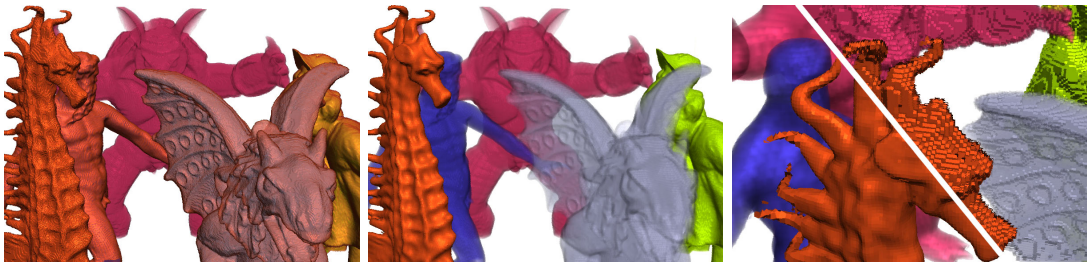


Figure 4.1: **Rendering of a labeled scene.** Left: The usual linear interpolation method suffers from color bleeding, since interpolated values generate false colors when accessing transfer functions even when using pre-integration. Middle: using split-voxels, all segmented models keep their color. Right: a close-up of the scene with low pixel tolerance shows how the split-voxel method preserves boundaries even at very low levels of details, instead the nearest method suffers from low quality edge representation.

- A hierarchical approach for converting a standard mono-resolution voxelized representation into an out-of-core multi-resolution structure based on *split-voxels*, both for labeled and continuous scalar volumes;
- A simple and efficient ray-casting accumulation scheme exploiting *split-voxels* incorporated within a GPU accelerated out-of-core renderer providing real-time exploration with dynamic transfer function editing.

This representation has little overhead over storing precomputed gradients, and has the advantage that feature planes provide minimal geometric information about the underlying volume regions that can be effectively exploited for volume rendering. In particular, *split-voxels* are able to track material interfaces, as they occur in many physical objects, and, when employed in a multi-resolution representation, provide accurate silhouettes even at very coarse levels of detail, reducing the data and time required to render understandable images (see Fig. 4.1). We show that the *split-voxel* primitive can efficiently model volume datasets containing intensity scalar values, as well as material labels. Since the method is applied to scalar values, the renderer is able to change the transfer function in real-time without the need to reprocess the data.

The *split-voxel* model is tuned for scalar fields containing sharp boundaries between different near uniform zones. One limitation of *split-voxels*. The representation is non-continuous, and volumes exhibiting smoothly varying gradient fields, e.g., the results of gas/fluids/fire simulations, are not managed as efficiently.

4.2 Related Work

Reconstructing the original data, as accurately as possible from a discretized representation, while maintaining a reasonable execution speed and limited memory needs, is a fundamental problem in volume graphics. The most common representation of volumes in (GPU) volume rendering applications is a regular grid of scalar values. The most direct way to reconstruct data from the sampled representation is to use a polynomial filter. Constant and linear filters, directly supported by the hardware, are by far the most commonly used representations, but have obvious limits in their reconstruction power. In case of rapidly varying data, too many samples are required to avoid aliasing problems, with problems in

terms of memory occupancy and traversal efficiency. Sparser representations can in principle be obtained with higher order filters. Even though these filters (typically the tricubic one) can be smartly implemented on top of linear interpolation to reduce their computational cost [Sigg 04], their run-time evaluation remains costly for an interactive renderer due to the many texture fetches. Ringing artifacts may also appear, and sharp boundaries are not directly represented. Moreover, with these continuous representations, sampling has still to be performed at a high frequency to avoid missing sharp value changes. Various approaches for reducing sampling rates are known in the literature. Pre-integrated volume rendering [Enge 01, Roet 03, Lum 04, Kye 08] deals with high frequencies in transfer functions rather than in the scalar volume itself. Adaptive sampling methods solve the boundary detection problem in real-time by adjusting sampling frequency during ray casting. For instance, Hadwiger et al. [Hadw 05] combined rasterization of min-max blocks with adaptive sampling and a secant method solver to ray cast discrete isosurfaces on the GPU, while Knoll et al. [Knol 09] employ a peak finding strategy which explicitly solves for iso-values within the volume rendering integral. The underlying sampled function is still considered continuous, and due to the cost of repeated evaluation, is at most a low order polynomial. A number of systems deal with segmented datasets (e.g., [Hadw 03, Roet 03]), but the focus is on blending separate gridded representations using different pre-determined transfer functions. Our method, instead, encodes minimal boundary information in the *split-voxel* primitive. This way, the primitive can be used for constructing multi-resolution structures while preserving boundary details even for low resolution scales, and the model can easily be employed in all kinds of accumulation strategies commonly considered for volume ray casting. The underlying concept of our *split-voxel* has been explored in different contexts before. In two dimensions, geometrical wavelets, which explicitly encode discontinuities, were introduced to properly catch edges often present in images [Dono 99, Will 03], and they have been applied to approximation and compression problems. Representations that enrich images with codes for local boundaries have also been introduced with the main purpose of combining vector and raster image features [Sen 04, Tumb 04, Rama 04]. Modified 2D interpolation rules requiring multiple fetches per texel are used to support up to four linear discontinuities in every pixel. The Pinchmap [Tari 05] improves over these methods by proposing a GPU-optimized 2D texture representation which separates the encoding of discontinuities from the encoding of the signal. The two-colored pixel approach of Pavic and Kobbelt [Pavi 10], developed in parallel to our work, also uses a representation using two colors per pixel separated by a feature line, and extends it to 3D for a video re-targeting applications. Other methods are instead targeted to extract feature preserving surface representations from volume data [Kobb 01, Ju 02]. While these approaches conceptually share similarity to ours, they cannot applied to volume rendering. In the context of volume rendering, Sereda et al. [Sere 06] introduced the LH space as a transfer function domain. It starts from the assumption that every voxel lies either inside a material or on the boundary between two materials with lower intensity FL and higher intensity FH, respectively. They show that the LH histogram conveys information about dataset boundaries in a more compact and robust way than common intensity-gradient histograms and therefore seems to be well-suited for volume exploration. Praßni at al. [Pras 09] improved this approach by deriving an efficient technique for the computation of LH values, which is fast enough to support post-classification at interactive frame rates. The underlying assumption is similar to our *split-voxel* concept, but while these methods deal with the generation of

efficient transfer functions, we instead focus on modeling and rendering volume data. With respect to the construction of the *split-voxel* model, our approach deals with the approximation of a small cubic scalar volume with two materials and a separation surface. To this end, various statistical methods can be employed to perform this classification, such as, for example, the well-known Support Vector Machines [Vapn 95] or the enumeration strategy of Pavic and Kobbelt [Pavi 10]. The *split-voxel* construction deals also with the 3D edge-detection problem, which has been investigated in depth in the literature. Our technique shares similarity with the moment-based operator [Luo 93]. Finally, the *split-voxel* model is incorporated into the multi-resolution CUDA rendering framework described in the previous chapter (see chapter 3).

4.3 The *Split-Voxel* Volumetric Primitive

We consider a generic scalar regular volume, i.e., a discrete representation over a cartesian grid of a continuous scalar function $s(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^3$. In this representation, the scalar value associated to each voxel is derived from some sort of measurement apparatus or simulation or digital processing, and it takes the form of an intensity measured value, in the case, for example, of data directly coming from medical scanning devices, or the form of a material label, when some classification has been performed over data. In order to enhance the appearance of surface structures in the volume, it is common to exploit the gradient ∇s to evaluate a surface shading model. High quality volume renderers, thus often precompute gradient values at each voxel position. Precomputing the gradients, rather than evaluating them at run-time, allows the renderer to use high quality filters with kernels larger than those which can be afforded for on-the-fly computations. Moreover, since the gradient is a bad predictor for the surface normal orientation in nearly homogeneous regions due to the increased influence of noise, gradients with low magnitude are typically filtered out. For instance, Bruckner el al. [Bruc 09] filter out all gradients whose norm is less than an eighth of the maximum gradient magnitude in the dataset. In order to better handle situations in which

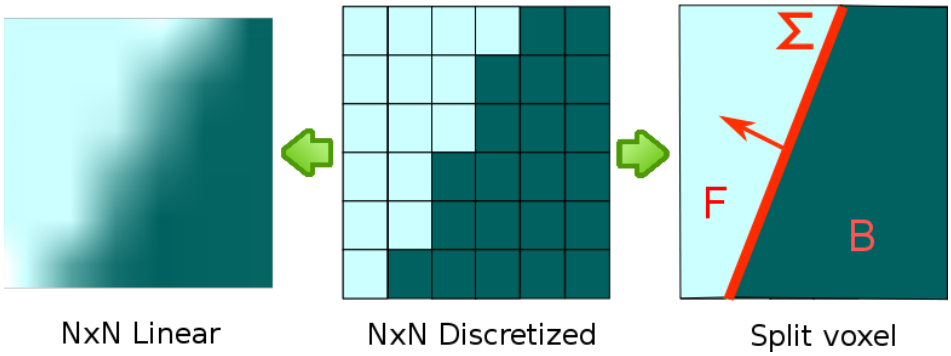


Figure 4.2: ***Split-voxel* primitive.** A *split-voxel* replaces a block of N^3 voxels by one single voxel that is split by a feature plane into two regions with constant values. The feature plane provides a linear approximation to the strongest value discontinuity in the block, while the two values represent medians or averages which are not blurred over the discontinuity.

the gradient changes abruptly, rather than simply precomputing and prefiltering gradients, we incorporate in our *split-voxel* representation a minimal approximation of the locations of the strongest value discontinuity within the *split-voxel* region. Specifically, a *split-voxel* is defined as a volumetric primitive representing the interface between two intensity values (or

label values in the case of segmented volume data). The following information is encoded inside a *split-voxel* (a 2D schematic representation is showed in Fig. 4.3): two scalar values F and B, and the equation of an oriented feature plane $\Sigma = (\alpha, \beta, \gamma, \delta)$ indicating the boundary between F and B. The same representation is also valid for voxels containing uniform materials, that are encoded by imposing that front material is equal to back material and by zero-ing the separation plane. The representation has the following properties:

- the feature plane Σ provides a linear approximation to the strongest value discontinuity in the block. It replaces the gradient as an indication of the local surface normal, and the availability of a plane constant provides additional information on the location of the discontinuity;
- the values F, B represent medians or averages which are not blurred over the discontinuity. Hence, they can be used for sharp value evaluation and exploited for implementing multi-resolution filtering schemes. In particular, a sharp value for a point $P(x, y, z)$ inside the block can be obtained by computing a signed distance $d = \alpha x + \beta y + \gamma z + \delta$, and selecting F if $d > 0$, and B otherwise;
- the storage overhead with respect to a typical precomputed gradient representation is limited to two values: an additional scalar and the plane constant; moreover, the additional information permits to render at coarser levels of detail, leading to reduced memory needs and frame rendering times. Each *split-voxel* can thus replace blocks of N^3 voxels.

The representation thus trades continuity of the reconstruction with the ability to rapidly track value changes at negligible cost during rendering algorithms. Boundary sharpness with *split-voxels* is infinite, and will not depend on the density of sample points as it might with plain voxels. It is clear that only a single discontinuity can appear in a single *split-voxel*, but this limitation is unavoidable for fixed-size representations. Limiting the representation to be linear makes it fast to compute and efficient for rendering.

4.4 Preprocessing Using the *Split-voxel* Primitive

The split-voxel construction replaces gradient precomputation and data filtering in the volume preprocessing pipeline of our volume rendering framework. The goal is to convert a standard mono-resolution voxelized representation into a out-of-core multi-resolution structure based on split voxels, both for labeled and continuous scalar volumes. The process is hierarchical and constructs an octree of volume bricks, in which each brick element is a split voxel. We first detail how a single split voxel is constructed from a gridded representation, and then we explain how the procedure is hierarchically applied to complete the multi-resolution hierarchy.

4.4.1 Constructing a *Split-voxel*

Since a *split-voxel* represents a generic small cubical volume portion, it can be derived from a standard gridded representation by considering a voxel block of N^3 with resolutions starting from $N = 2$. In order to increase continuity among adjacent *split-voxels*, a boundary overlap of M voxel layers can be also considered during construction (typically $M = 1$ or 2), thus resulting in grids containing $(N + 2M)^3$ voxels (see Fig. 4.3). The *split-voxel* construction

procedure specifically consists of identifying two main values (in the case of intensity based volumes) or labels (in the case of segmented volumes), and the plane which better separates the two material clusters.

4.4.1.1 Discrete Data

In the case of segmented data, material labels are intrinsically defined inside the volume grid, and no class identification is needed. Hence, value classification consists of identifying the one or two most frequent labels inside the block. When more than two materials are present inside the volume, region-growing approaches could be also considered, but here we decided to limit the definition of *split-voxel* to the two most important materials in terms of frequency. Once classes are identified, if the volume can be represented by a single value, the associated *split-voxel* is assigned the same front and back materials and a null separation plane, otherwise the separation plane is rapidly estimated by considering the geometric cluster barycenters $\mathbf{p_f}$ and $\mathbf{p_b}$ of the two identified materials, and the two material frequencies n_f and n_b. Specifically, the split surface is defined as the plane with normal $\mathbf{n}$ and passing through $\mathbf{p}$, where

$$\mathbf{n} = \frac{\mathbf{p_f} - \mathbf{p_b}}{\|\mathbf{p_f} - \mathbf{p_b}\|} \tag{4.1}$$

$$\mathbf{p} = \mathbf{p_b} + (\mathbf{p_f} - \mathbf{p_b})\frac{n_b}{n_f + n_b} \tag{4.2}$$

as depicted in Fig. 4.3. If the distance between $\mathbf{p_f}$ and $\mathbf{p_b}$ is too small (a fraction of the voxel size), it is considered that the region is not easily linearly separable and we assign a uniform *split-voxel* with the most frequent label. This happens, for example, in the case of noisy regions, of small convex clusters almost entirely wrapped by another material. In most situations, however, a good linear approximation of the separating surface exists. While more elaborate solutions are possible, we found that this method provides a good approximation of the orientation and location of the strongest discontinuity (see Fig. 4.4 left).

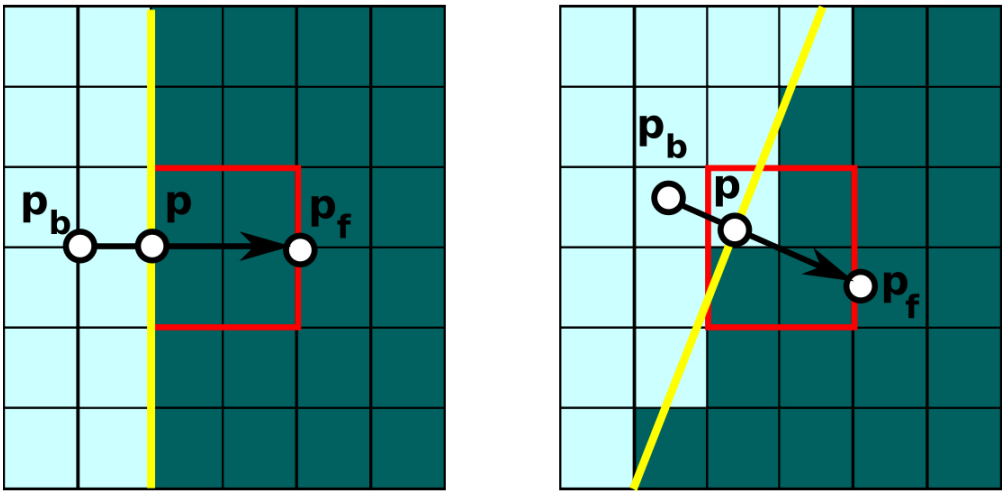


Figure 4.3: ***Split-voxel* construction scheme for segmented data.** Two examples of *split-voxel* construction. *Split-voxel* width N is 2 and boundary overlap M is also 2. Red part is the *split-voxel*, while the separation plane is depicted in yellow, and is computed according to equations 4.1, 4.2

4.4.1.2 Sampled Data

When volume data are intensity scalars, we choose the approach to locally convert the data to a segmented representation before applying the same procedure used for the discrete case. To this end, we employ the classical iterative k-means algorithm [Lloy 03] to cluster the intensity values. Specifically, two initial means are defined as the minimum and the maximum intensity values available in the grid. The iterative procedure creates the two clusters by associating every intensity value in the grid to the nearest mean, and it updates the means by computing the centroid of the clusters. Clustering stops when the the assignments to clusters no longer change or after a fixed large number of iterations. Generally, after few iterations the algorithm converges with two resulting intensity values representing the two materials. At this point, if these intensity values are similar (their difference is less than a given threshold), we assume that a uniform material is contained inside the volume, otherwise we classify the grid voxels with respect to the two final means and we refer back to the same procedure employed for label volumes, by computing the separation plane according to equations 4.1 and 4.2. The front and back values are then defined by each cluster median values. We verified that the planes found by this simple method provide in practice good local approximations of the surface (see Fig. 4.4 right).

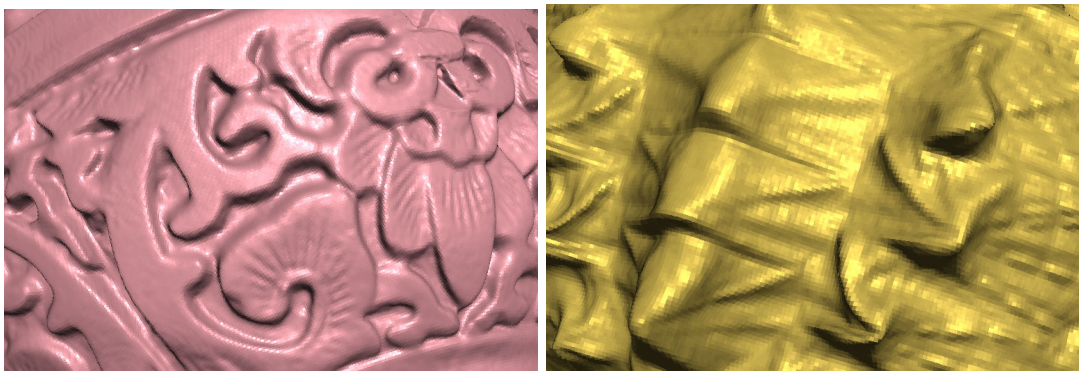


Figure 4.4: **Close-up views of segmented datasets.** The left image shows a segmented dataset ($1911 \times 1908 \times 1813$ resolution), while the right one shows a post-classified scalar dataset ($512 \times 512 \times 400$ resolution). Note the nice normals reconstructed by the simple *split-voxel* construction technique and the good matching between neighboring planes.

4.4.1.3 Encoding

In the case of typical 8-bit datasets, each *split-voxel* is stored in a 6-byte structure: the plane is represented in 4 bytes, while front and back values use a byte each one. For the sake of simplicity, and to maintain regularity in the structure and avoid too many indirections, we store uniform voxels using the same representation. The *split-voxel* dataset size has thus in the worst case at most 50% overhead over the typical scalar dataset stored with precomputed gradients at the same resolution (6 bytes in place of 4 bytes per element). However, since *split-voxels* are normally used to represent N^3-voxel resolution volume grids, the actual occupancy ratio is roughly $3/(2N^3)$. For example, in the case of *split-voxels* representing just 2-voxel width grids, the occupancy ratio is $3/16$, i.e. 18.75% with respect to the original dataset size.

4.4.2 Multi-resolution Hierarchy Construction

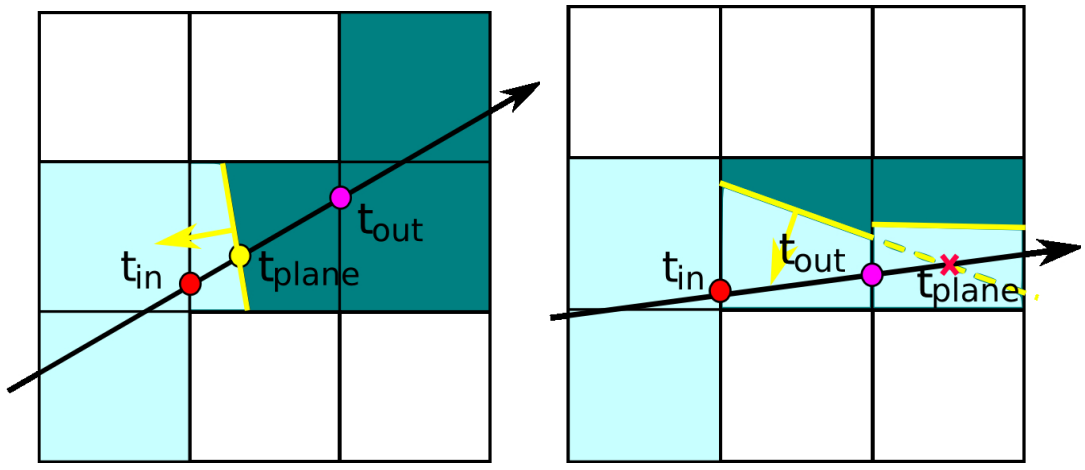
Our multi-resolution structure is a coarse-grained octree whose nodes are bricks of *split-voxels*, which are built with a bottom-up procedure. Each brick is a cubical block with an edge width of some tens of *split-voxels* [Gobb 08]. Leaf bricks are computed by sampling the input volume data and by encoding each brick component into a *split-voxel* representation using the technique of Sec. 4.4.1. In order to construct non-leaf octree bricks, instead, we first re-sample into a gridded representation the *split-voxels* of the next finer level which fall within the area of interest for construction. In this re-sampling procedure, each voxel receives the most dominant among the front and the back value of the associated *split-voxel* (estimated by sampling the voxel). Once this procedure is terminated, the volume subsumed by the brick (plus M layers of overlap), is represented as a regular voxel grid. The procedure detailed in Sec. 4.4.1 is then applied to convert it to the *split-voxel* representation. The procedure is hierarchically repeated until we reach the octree root. It should be noted that the effect of the procedure is dual. First of all, each discontinuous *split-voxel* ends up having an associated feature plane. Moreover, values are low pass filtered throughout the hierarchy while not blurring values over discontinuities. The procedure thus applies edge-preserving filtering for level of detail construction.

4.5 Rendering

We have integrated the *split-voxel* primitive in a GPU accelerated out-of-core multi-resolution renderer based on single pass octree ray-casting [Gobb 08, Cras 09, Igle 10].

4.5.1 Hardware Accelerated Out-of-core Multi-resolution Rendering

CPU and GPU strictly cooperate to manage octree traversal and ray-casting. Specifically, a CPU thread, given the current viewing parameters, has the responsibility of identifying an octree cut, by refining bricks whose projected voxel size is higher than a user-selected threshold, while updating the associated LRU cache of bricks in the GPU by loading data from out-of-core storage. The GPU cache is updated incrementally frame by frame, and each brick can be reused over several frames, exploiting temporal and spatial coherence, thus limiting the required frame bandwidth. Moreover, the CPU thread builds at each frame an octree spatial index for the current cut, which allows the GPU to perform the traversal of the current uploaded octree cut. The GPU performs ray-casting, using the spatial index to enumerate the current cut leaf nodes traversed by the ray in front-to-back order. If the visited node is empty, it is simply skipped. If the node is non-empty, the associated brick is selected in the texture cache, and all the relevant *split-voxels* are traversed, while accumulating color and opacity contributions according to the optical model employed. Octree traversal terminates when the ray leaves the volume bounding box, or when full opacity is reached. A variety of accumulation schemes can be employed for split voxels. In this work, we describe the classic DVR scheme using a 1D transfer function to map values to colors and opacities. Since our scheme exploits variable step sizes, the renderer works with extinction weighted colors, and we thus have a transfer function $\tau(s) \in [0, \infty[$ for the extinction coefficient and a transfer function $c(s)$ for the color, that has to be multiplied by $\tau(s)$ to yield an actual color intensity $\tilde{k}(s) = \tau(s)c(s)$ for a given scalar value s. We actually maintain a single 4-components texture



(a) Plane intersection inside voxel (b) Plane Intersection outside voxel

Figure 4.5: **The ray DDA traversal scheme.** Scheme of voxel traversal using the DDA algorithm to enumerate intersected voxels. In Fig. 4.5(a) the ray intersects the plane of the central voxel inside $[t_{in}, t_{out}]$. The front material is accumulated along $[t_{in}, t_{plane}]$, the back material along $[t_{plane}, t_{out}]$, and shading is computed for both by considering the plane normal. In Fig. 4.5(b) the ray-plane intersection lies outside the voxel: $t_{plane} > t_{out}$ and only front material is accumulated.

for $\tilde{k}(s), \tau(s)$. The opacity of a segment of length δt is derived from an extinction coefficient τ_i by $\alpha_i = 1 - e^{-\tau_i \cdot \delta t}$.

4.5.2 Non-empty Brick Traversal and *Split-voxel* Accumulation

Color and opacity accumulation is performed during non-empty brick traversal. We exploit a 3D digital differential analyzer (3DDDA) scheme [Fuji 85] to traverse all intersected split voxels in front-to-back order. For a given ray $\mathbf{r} = \mathbf{o} + \mathbf{d}t$, after an initialization step performed upon brick entry, all traversed voxels are enumerated, and at each step the intersection abscissae t_{in} and t_{out} between the ray and the current voxel are updated, and the length $\delta t = t_{out} - t_{in}$ of the current ray segment is computed. If the current split-voxel is uniform ($F = B$), we simply accumulate its unshaded contribution weighted with the ambient component. In order to do so, we fetch $\tilde{k}(F) = \tau(F)c(F)$, and convert them to opacity weighted colors associated to the length δt. When the split-voxel contains a feature plane, a shaded contribution is instead applied for the front and back values. We thus fetch both $\tilde{k}(F) = \tau(F)c(F)$ and $\tilde{k}(B) = \tau(B)c(B)$ and separately use them for color accumulation. First, we order materials with respect to the relative orientation of the plane normal $\mathbf{n}$ and of the viewing ray direction $\mathbf{d}$. When $\mathbf{n} \cdot \mathbf{d} \leq 0$, the front material is traversed first, otherwise the back material is traversed first. Second, we intersect the ray with the voxel plane to derive the t_{plane} abscissa value, and we accumulate the first material from t_{in} to t_{plane}, and the second one from t_{plane} to t_{out}, shading both the contributions by using the plane normal in the lighting equation, as indicated in Fig. 4.5(a). When t_{plane} lies outside the voxel, only one of the two materials is accumulated: the second material, if $t_{plane} < t_{in}$, the first one otherwise, as shown in 4.5(b). The rendering scheme thus trades continuity of the reconstruction with the ability to rapidly track value changes at negligible cost during rendering algorithms. Unrelated values are not combined together, and boundary sharpness is infinite and does not depend on resolution.

In many cases, this provides an improvement with respect to standard volume rendering techniques, which suffer when adjacent voxels contain very different intensity values, since interpolation tends to blur abrupt variations and to generate material values that are not physically present in original data in that position, often producing a color bleeding effect, as shown in Fig. 4.1 and 4.6. Finally, another advantage of our rendering scheme is that split-voxels can be projected to large screen areas, also for the coarsest level of details, since they are able to maintain a nice silhouette by shaping the boundaries with the voxel planes, as indicated in Fig. 4.6.

4.6 Results

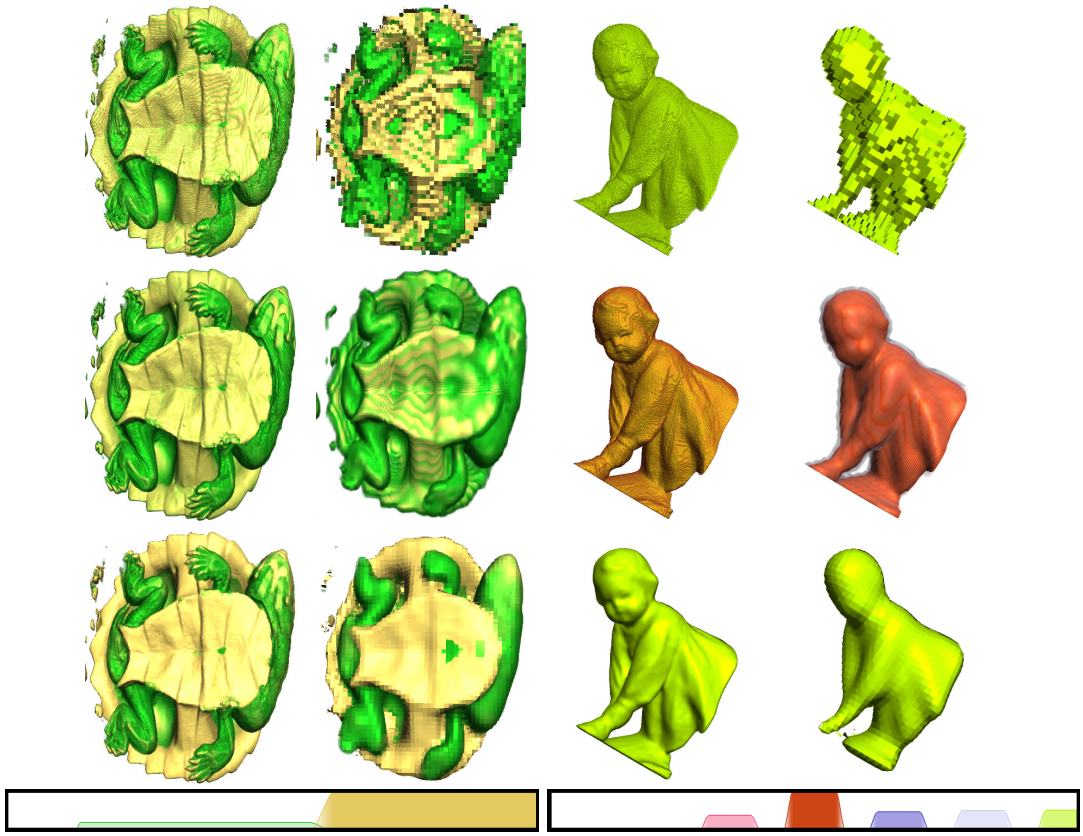


Figure 4.6: **Rendering quality comparison.** Volume rendering of a continuous intensity based dataset of a turtle CT scan, and a close-up view of the discrete statues volume (first columns pixel tolerance 2, second columns pixel tolerance 8). Top row: nearest method is not able to preserve features as resolution decreases, but produces the right color. Middle row: linear method keeps a smooth silhouette, but color bleeding effect is present and increases for decreasing resolution, and the boundary tends to blur for the coarser representations. Base row: split-voxel method preserves objects silhouette and sharp separation among different layers even at very low resolutions, while maintaining the correct colors. Bottom: transfer functions. The right transfer function is the same one as employed for Fig. 4.1.

A prototype software system implementing the presented techniques has been developed on a Linux system using C++ and CUDA. The out-of-core octree structure has been implemented on top of Berkeley DB, exploiting the LZO compression library to reduce memory occupancy of each split-voxel brick. We have tested our system with a variety of high resolution models and settings. In this section, we discuss the results obtained with the processing and inspection

of three 8-bit datasets: a $512 \times 512 \times 400$ micro-CT of a Mata Mata turtle specimen (Source: Digital Morphology Project, the CTLab and the Texas Advanced Computing Center, University of Texas, Austin), a $1911 \times 1908 \times 1813$ synthetic labeled volume containing various surface models of statues which have been voxelized, and a $404 \times 474 \times 512$ labeled volume containing a segmented leg reconstructed from MRI acquisitions (Source: MiraLab, Geneva).

4.6.1 Preprocessing

Datasets were processed on a Linux PC Intel Core 2, 2.66 GHz. The construction of the octree from source data was performed using octree bricks of 32^3 split voxels. Each split voxel was constructed from a discretized grid of 2^3 voxels with a 2 layers overlap (i.e., a 6^3 sampling grid). For the statue dataset, data processing took 7 hours and produced a $360MB$ octree database starting from an uncompressed source size of $3.1GB$, while for the turtle dataset, data processing took 15 minutes and produced a $60MB$ octree database from an uncompressed source size of $100MB$, and finally for the leg dataset data processing took 10 minutes and produced a $10MB$ octree database from an uncompressed source size of $100MB$ Processing times are comparable to those of other systems using high quality gradient precomputation (e.g., a Sobel 5^3 kernel) [Gobb 08]. The percentage of the split-voxel in the statue and leg dataset is 2% with respect to 98% of constant voxels. On the other side, for the generation of the turtle dataset we considered a separation threshold of 10 and we got a percentage distribution completely different: 72% split-voxels, with respect to 28% constant voxels. This distribution difference explains also the difference in compression ratio between label and scalar volumes. For comparison purposes, we also built multi-resolution datasets to be used with nearest and trilinear rendering. Datasets to be used with the nearest technique were produced by considering a median filter to reconstruct a voxel from its 8 children, while an average filter was employed for datasets to be used with the linear technique.

4.6.2 Rendering

The performance of our rendering system prototype was evaluated on a Linux PC Intel Core 2, 2.66 GHz, equipped with an NVidia GTX 280. We considered a number of interactive inspection sequences using the three models and measuring actual frame rates (i.e., not only raw rendering times, but frame-to-frame times). We report here on the results obtained when using a window size of 800×600 pixels. Our technique efficiently supports real-time transfer function manipulation: please refer to the accompanying video available online [1] for interactive sequences recorded live. When using transfer functions ranging from moderately opaque to highly transparent, we experienced that the frame rate of typical inspection sequences varies between 10Hz for extreme close-up views with transparency to over 40Hz for overall views. Interactive rates are thus guaranteed even in the most demanding situations. With respect to image quality, we compared our split-voxel rendering method to common direct volume rendering strategies, employing nearest filtering and trilinear interpolation filtering. The nearest filtering strategy is able to separate between different materials, but it has the problem that reconstruction quality is intrinsically poor and needs high resolutions to get good quality images. On the opposite side, trilinear filtering increases reconstruction quality at the cost of losing boundary features. Furthermore, it suffers from color bleeding,

[1] See the online video at: *http://vic.crs4.it/vic/cgi-bin/multimedia-page.cgi?id='155'*

since interpolated values generate false colors when accessing transfer functions even when using pre-integration. Instead, our rendering method exploiting split-voxels is able to keep separation between materials, so that each material keeps its color, also in the presence of an impulsive transfer function. Furthermore, boundaries are well preserved even at very low levels of details. Figure 4.6 compares the three rendering techniques applied to the datasets considered with respect to the ability of avoiding color artifacts, and the ability to reconstruct correct object silhouettes at various resolutions. In fact, for the segmented data, which is a detail of the statues label dataset, in the case of trilinear interpolation other parts of the transfer function modify the color of the model, while the split-voxel method keeps the correct color for each object in the scene (see also Fig. 4.1 and Fig. 4.8). Similarly, for the turtle

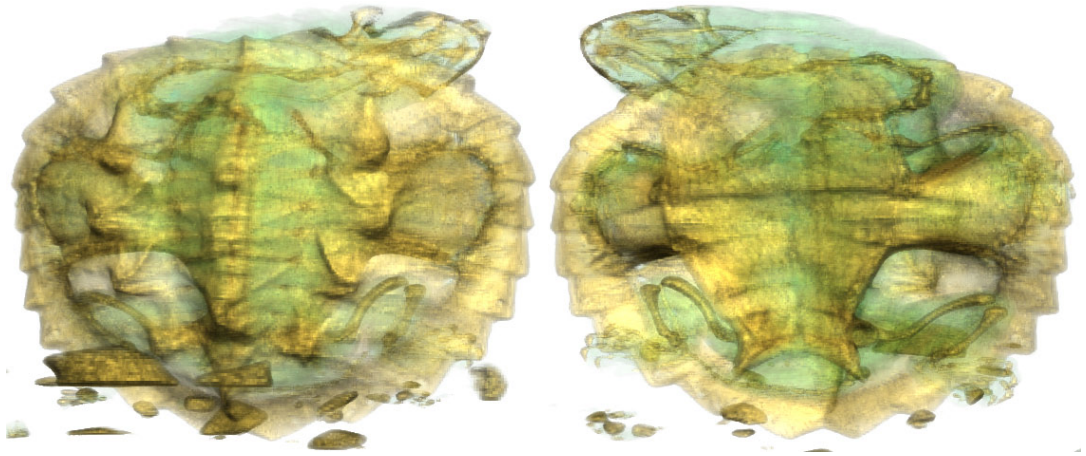


Figure 4.7: **Semi-transparent volume rendering.** Two views of the turtle dataset.

intensity-based dataset, when using trilinear interpolation the boundary between bone and air is degraded by the interpolated value, especially for the simplified model. Figure 4.7 shows two semi-transparent views of the turtle dataset. Figure 4.6 compares the three techniques also with respect to the ability to preserve features at low levels of detail. It appears evident that the nearest method suffers when resolution decreases, while trilinear interpolation keeps a smooth but thicker silhouette and introduces a color-bleeding problem, and finally the split-voxel method reconstructs nice silhouettes even for very low levels of detail, preserving the correct color associated by the transfer function. The low-resolution representations use very little memory. At extreme magnification levels, the discontinuous nature of the representation becomes evident, but the images remain understandable. The nice quality of low levels of details can be an advantage in a number of applications, e.g., for streaming and remote rendering.

4.7 Conclusion

We presented in this chapter a novel volumetric description, which trades continuity with the ability to model infinitely sharp value changes. This representation had little overhead over storing precomputed gradients. Separation planes provided in fact minimal geometric information about the strongest discontinuity in the underlying volume regions, which can be effectively exploited for multi-resolution data filtering and volume rendering. In particular, we were able to loosely track material interfaces, as they occur in many physical objects, avoiding

the mixing of unrelated values. When employed in a multi-resolution representation, nice silhouettes were preserved even at very coarse levels of detail, reducing the data and time required to render understandable images. We shew that the *split-voxel* primitive can be applied to volume datasets containing intensity scalar values, as well as material labels. Since the method was applied to scalar values, the renderer was able to modify the transfer function in real-time without the need to reprocess the data. Even though our implementation can be improved in many aspects, our approach was a novel attempt to model discontinuities inside a voxel primitive in the context of a ray casting framework.

4.8 Bibliographical Notes

The content of this chapter were mainly based on the paper [Agus 10b], where we presented the *split-voxel* primitive for discontinuity-preserving voxel representation of volumetric data by encoding scalar data together with edge detection information.

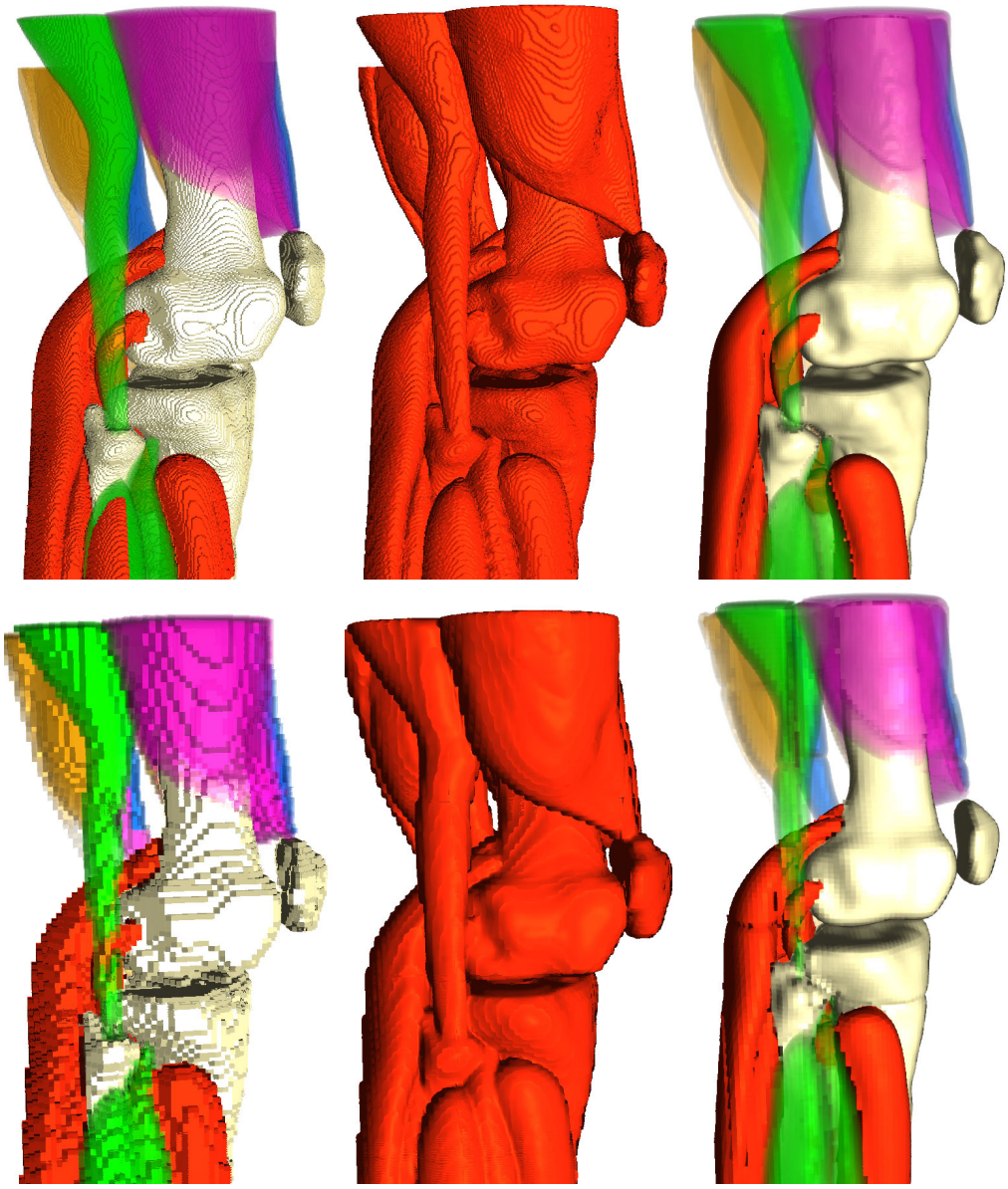


Figure 4.8: **Knee segmented model.** Top row: knee model at pixel tolerance 2, base row same model at pixel tolerance 8. As explained on Fig. 4.6 the nearest method (left) preserves proper colors but presents jagged outlines, the linear one (center) preserves boundaries but produces color-bleeding effect, while the split-voxels (right) produces proper colors while maintaining nice silhouettes also for the coarsest pixel tolerance.

Chapter 5

Rendering on Light-field Displays

Improved volumetric understanding can be achieved by presenting results on displays able to elicit more depth cues than the conventional 2D monitors. This chapter presents a GPU-accelerated volume ray casting system interactively driving a multi-user light-field display. The display is based on a specially arranged array of projectors and a holographic screen that provides full horizontal parallax. The characteristics of the display are exploited to develop a specialized volume rendering technique able to provide multiple freely moving naked-eye viewers the illusion of seeing and manipulating virtual volumetric objects floating in the display workspace. The method achieves interactive performance and provides rapid visual understanding of complex volumetric datasets even when using depth-oblivious compositing techniques. Perceptual experiments are also presented to evaluate the depth-discrimination capabilities of this technology with respect to two-view (stereo) and discrete multi-view designs.

5.1 Introduction

The rapid development of programmable graphics hardware is making it possible to interactively render volumes with high visual fidelity on commodity PCs. Resolving the spatial arrangement of complex three-dimensional structures in images produced by volume rendering techniques is however often a difficult task. In particular, medical data produced by CT or MRI scans often contain many overlapping structures, leading to cluttered images which are difficult to understand. Enhancing depth and shape perception in volumetric rendering is thus a very active research area, which is tackled under different angles. Recent contributions include methods for supporting real-time rendering and user interaction, improving rendering quality with advanced photorealistic models, e.g., including specular reflections and shadows, or developing non-photorealistic approaches to emphasize model features by illustrative techniques, e.g, vicinity shading, edge detection, halos, or cutaways. An orthogonal research direction consists of improving volumetric understanding by presenting results on displays able to elicit more depth cues than the conventional 2D monitor or providing improved color reproduction. For instance, Ghosh et al [Ghos 05] have shown how a high dynamic range display can substantially improve volume understanding through perceptually optimized transfer functions. In this work, we focus on enhancing spatial understanding of 3D data through perceptual cues for accommodation, stereo and motion parallax delivered by a light-field display, i.e., a display supporting high resolution direction selective light emission. This direction looks very promising, since there is evidence that ego- and/or model-motion as well as stereopsis are essential cues to achieve rapid direct perception of

volumetric data [Bouc 09, Mora 04]. Recent advances in 3D display design demonstrate that high resolution display technology able to reproduce natural light-fields is practically achievable [Balo 05, Jone 07]. Rendering for such displays requires generating a large number of light beams of appropriate origin, direction, and color, which is a complex and computationally intensive task. Moreover, the displays optical characteristics impose specialized rendering methods.

In this chapter, we present and demonstrate a GPU accelerated volume ray casting system interactively driving a multi-user light-field display based on projection technology. The display, driven by a single programmable GPU through a DVI link, is based on a specially arranged projector array placed behind an anisotropic holographic screen. This setup provides full continuous horizontal parallax in a sizeable zone in front of the screen. The restriction to horizontal parallax reduces light-field complexity, making the real-time rendering problem more tractable. The characteristics of the display are exploited by a specialized rendering technique able to provide multiple freely moving naked-eye viewers the illusion of seeing virtual volumetric objects floating at fixed physical locations in the display workspace (see Fig. 5.5.1).

Central to our approach is a GPU ray-caster that follows rays generated by a multiple-center-of-projection (MCOP) technique, while sampling prefiltered versions of the dataset at resolutions that match the varying spatial accuracy of the display. Our main contributions thus include:

- a general MCOP technique for producing perspective correct images on a class of horizontal parallax only light-field displays based on anisotropic diffusers; the solution provides a correct solution for viewers at a known distance and height from the screen and a good approximation for all other positions;
- a GPU accelerated framework implementing volume ray-casting on a light-field display using a combination of vertex and fragment shaders;
- the description and demonstration of a prototype hardware/software system achieving interactive performance on non-trivial datasets on a single PC configuration;

Although not all the techniques presented here are novel in themselves, their elaboration and combination in a single system is non trivial and represents a substantial enhancement to the state-of-the-art.

Continuous automultiscopic displays represent a promising technology, able to drive users into really involving and compelling experiences. In this chapter, we report on perceptual experiments carried out to verify whether the greater complexity of continuous multiview systems is worthwhile also from a perceptual point of view, when compared to stereo or discrete multiview displays.

5.2 Related Work

In the following section, we briefly describe the state-of-the-art in the fields of interactive 3d display technology, suitable image generation methods and existing GPU-accelerated volume visualization system exploiting multi-view displays.

5.2.1 Interactive 3D Display Technology

A huge number of approaches have been proposed to support naked-eye multiscopic visualization, and a full review of the subject is out of scope for this thesis. We provide here only a rapid survey of the subject, with a particular emphasis on the most closely related approaches. For a good review on the subject we refer the reader to [Dodg 05]. The key technical feature characterizing 3D displays is direction-selective light emission, which is obtained most commonly by volumetric, holographic, or multi-view approaches. Volumetric displays synthesize light-fields by projecting light beams on refractive/reflective media positioned or moved in space (e.g., [McKa 00, Fava 01, Robe 00]). Commercial displays are readily available (e.g., from Actuality Systems). The main disadvantages are the limited scalability of the approach, and the difficulty in presenting occlusion effects. The latter problem has been recently solved in the displays presented by [Jone 07] and [Coss 07], which employ an anisotropic diffuser covering a rapidly spinning mirror illuminated by a single high speed video projector synchronized with mirror rotation. Such a setup allows for 360° viewing, but, because of mechanical constraints, is practical only for limited image sizes and model complexity. Pure holographic techniques are based on generating holographic patterns to reconstruct the light wavefront originating from the displayed object, e.g., using acousto-optic materials [St H 95], optically addressed spatial light modulators [Stan 00], or digital micro-mirror devices [Hueb 03]. Although this approach can theoretically provide the most compelling imagery, the principle itself imposes limitations on realistically achievable image sizes, resolution, speckle, with consequent narrow fields of view, alongside enormous computing capacity required to reach acceptable refresh rates for true interaction. In current prototypes, still confined in research labs, the display hardware is very large in relation to the size of the image (which is typically a few centimeters in each dimension). Typical multi-view displays, often based on an optical mask or a lenticular lens array, show multiple 2D images in multiple zones in space. They support multiple simultaneous viewers, but at the cost of restricting them to be within a limited viewing angle. Multi-view displays are often based on an optical mask or a lenticular lens array. Optical masks introduce significant light loss when there are more than two views. Moreover, the barrier structure becomes visible as the number of views increases. On the other hand, lenticular displays magnify the pixel matrix of the projecting devices creating dark zones between viewing slots. The Cambridge multi-view display is a classic design in this area [Dodg 00], and a number of manufacturers (Philips [van 96], Sharp [Wood 00], Opticality [Relk 05], Samsung, Stereographics, Zeiss) produce monitors based on variations of this technology. Typical state-of-the-art displays typically use 8–10 images, i.e., directions, at the expense of resolution. Matusik et al. [Matu 04] demonstrated a prototype based on this technology and assembled with sixteen 1024x768 projectors and a lenticular screen. As in our case, the setup requires one projector per view. However, their screen achieves vertical diffusion not by diffusing light vertically from the screen as in our display, but by focusing light horizontally onto a diffuse surface, yielding a different projection geometry. A 3D stereo effect is obtained when the left eye and the right eye see different but matching information. The small number of views of multi-view systems based on masks or lenticulars produces, however, cross-talks and discontinuities upon viewer's motion [Dodg 96]. The displays used in this work [Balo 05], are produced by the Holografika company, uses the distributed image generation approach of projector-based multi-view technology, but removes some of the intrinsic optical limitations, as it offers a fully continuous blend among views thanks to the

light shaping capabilities of a holographically recorded screen. One limitation of most 3D display solutions, also shared by the displays used in this work, is that only horizontal parallax is provided. Yang et al.[Yang 06] presented an improved system based on a cluster of projectors that provides parallax both horizontally and vertically. This solution provides a more faithful light-field reconstruction but requires the generation of a much larger number of rays to achieve the same spatial accuracy, which makes it currently practical only for very small image areas or narrow fields of views.

5.2.2 Projecting Graphics to the 3D Display.

The image generation methods employed in conjunction with 3D displays must take into account the display characteristics both in terms of geometry and resolution of the reproduced light-fields. In our work, we employ a multiple-center-of-projection technique to produce images that exhibit correct stereo and motion parallax cues. Our projection algorithm relates to previous work in light-field rendering and holography. The multiple-viewpoint-rendering approach [Hall 98] harnesses perspective coherence to improve the efficiency of rendering multiple perspective image sequences. Halle et al. [Hall 91] proposed a method where static holographic stereograms account for the viewer's distance but not their height. The light-field display recently presented by Jones et al. [Jone 07] also uses a MCOP approach, similar to ours, but their display geometry is radically different. Previous work in rendering for light-field displays have used, typically, standard orthographic or perspective projections [Rask 98, Coss 07]. This approach simplifies rendering using a fixed function graphics pipeline but produces perspective distortions when applied to displays based on anisotropic light shaping elements [Jone 07]. None of these works takes into account the finite angular size of the light beams to adapt sampling rates as a function of distance from the screen. A framework for studying sampling and aliasing for 3D displays has recently been proposed by Zwicker et al [Zwic 06].

5.2.3 GPU-accelerated Volume Visualization on Multi-view Displays

Many sophisticated techniques for real-time volume rendering have been proposed in the past, taking advantage of CPU acceleration techniques, GPU acceleration using texture mapping, or special purpose hardware. In the last few years, improvements in the programmable and performance capabilities of GPUs have made GPU solutions the main option of choice for real-time rendering. We refer the reader to the recent book of Engel et al. for a recent survey [Enge 06]. Our initial approach was based on the GPU ray-casting approach [Krug 03, Roet 03] evolving later to a system compatible with the out-of-core multi-resolution volume rendering engine described in chapter 2. As in recent single-pass GPU ray-casters [Steg 05], we exploited GPU vertex shaders to render proxy geometry that activates a fragment shader performing the actual ray-casting. Our factorization of the ray computation operations is however different, since our MCOP rendering pipeline cannot rely on the interpolation performed by the rasterizer to pass down combined 3D and projected data from the vertex shader. Our technique for MCOP rendering using GPU shaders is also related to work of Hou et al. [Hou 06], which focuses on simulating reflections and refractions, and Jones et al. [Jone 07], which, however, exploits only vertex shaders for geometry projection. for In our approach, adaptive

performance is obtained by controlling image sizes and sampling rates. Others have proposed acceleration methods for stereo volume rendering, which mainly use the information from one view re-projected to the other [He 96, Koo 99, Wan 04]. This results in improved speed, which is however obtained at the cost of image quality [He 96, Adel 94]. These re-projection approaches are not typically applied to auto-stereoscopic rendering [Port 00], since they can lead to considerable noise when switching from a view to the next. Since our angular sampling rate is very high ($< 1°$), an adaptation of these approaches to rendering on a light-field display is worth exploring.

5.3 The Light-field Display Concept

The display used in this work is based on projection technology and uses a specially arranged projector array and a holographic screen. We summarize here the main concepts behind it. More information on the technology is presented elsewhere [Balo 05].

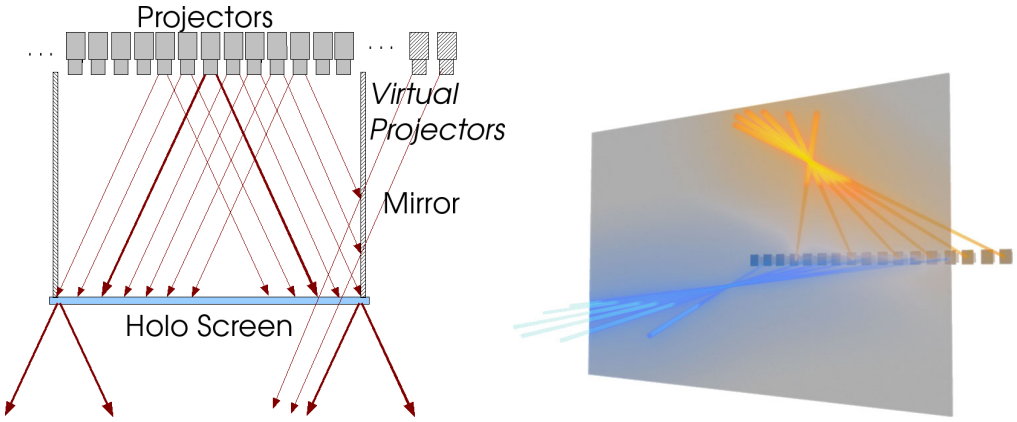


Figure 5.1: **Light-field display concept.** Left: Each projector emits light beams toward a subset of the points of the holographic screen. Side mirrors increase the available light beams count. Right: A large number of light beams can create a spatial point (voxel).

The projectors are densely arranged behind the screen, and all of them project their specific image onto the holographic screen to build up a light-field (see Fig. 5.1). By positioning mirrors at the sides of the display, it is possible to reflect back onto the screen the light beams that would otherwise be lost, thus creating virtual projectors that increase the display field of view. Each projector emits light beams toward a subset of the points of the holographic screen. At the same time, each screen point is hit by more light beams coming from different projectors. The holographic screen is the key element in this design, as it is the optical element enabling selective directional transmission of light beams. It is a holographically recorded, randomized surface relief structure able to provide controlled angular light divergence. The light diffusion characteristic of the screen is the critical parameter influencing the angular resolution of the system, which is very precisely set in accordance with the system geometry. In the horizontal parallax design, the projectors are arranged in a horizontal linear array and the angular light distribution profile induced by the screen is strongly anisotropic. Horizontally, the surface is sharply transmissive, to maintain a sub-degree separation between views. Vertically, the screen scatters widely so the projected image can be viewed from essentially any height. The angular light distribution profile introduced by the holographic screen is characterized by a wide plateau and steep Gaussian slopes precisely overlapping in a narrow region in the

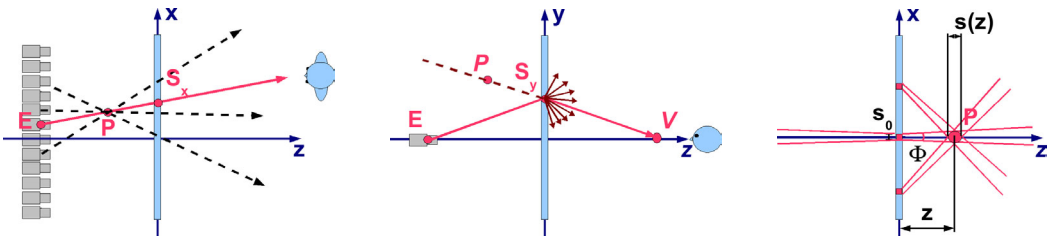


Figure 5.2: **Light-field geometry description**: Left: horizontally, the screen is sharply transmissive and maintains separation between views. Center: vertically, the screen scatters widely so the projected image can be viewed from essentially any height. Right: the finite angular size of the light beams determines the voxel dimension as a function of distance from the screen.

horizontal direction. This results in a homogeneous light distribution and continuous 3D view with no visible crosstalk within the field of depth determined by the angular resolution. A full parallax system would be created using a screen with narrow transmission profiles both in the horizontal and vertical direction. This design would require, however, a matrix of projectors for generating a much larger number of rays, significantly increasing the computational cost of image generation. Since humans perceive depth using horizontally-offset eyes and move their viewpoint more easily from side to side than up and down, the horizontal parallax only approach is adequate for most applications and provides significant speed-up.

5.3.1 Projecting Graphics to the Display

This simple linear perspective model defines how light is projected onto the screen, but is not sufficient to define how a 3D graphics application should project their models to the display, because it ignores the transformation performed by the holographic screen. Since the screen is selective only in the horizontal direction, but scatters widely in the vertical one, the displayed light-field's dimensionality is reduced, and the application must decide how to deal with the missing degree of freedom. In practice, at any moment in time, a given screen pixel has the same color when viewed from all vertical viewing angles. In order to provide a full perspective effect, the vertical viewing angle must thus be known, which amounts at fixing the viewer's height and distance from screen. Therefore, the renderer assumes a virtual viewer at height V_y and distance V_z from the screen. We assume that the screen is centered at the origin with the y axis in the vertical direction, the x axis pointing to the right, and the z axis pointing out of the screen. Given a virtual observer at $\mathbf{V}$, the ray origin passing through a point $\mathbf{Q}$ is then determined by:

$$\mathbf{O} = ((1-\eta)(V_x) + \eta(E_x + \frac{Q_x - E_x}{Q_z - Ez}(V_z - E_z)), V_y, V_z)) \tag{5.1}$$

where $\mathbf{E}$ is the position of the currently considered projector, and η is a interpolation factor, which allows us to smoothly transition from standard single view perspective rendering (with $\eta = 0$) to full horizontal parallax (with $\eta = 1$). In the latter case, the solution is exact for all viewers at the same height and distance from the screen as the virtual observer and proves in practice to be a good approximation for all other viewing positions in the display workspace. When $0 \ll \eta < 1$, the 3D effect persists and the visible depth range of the presented images is compressed. This reduces the direct mapping between real and virtual world, as the image

is not exact for any observer position, but allows us to map large objects within the restricted display range while maintaining a 3D effect for better spatial understanding Knowing how to compute **O** for every 3D point **Q** allows us to determine where **Q** should be drawn on a given projector to produce a perspective correct image using either a ray-caster or a rasterizer. A similar approach is taken by Jones et al. [Jone 07] for their 360° light-field display. Their approach also employs a MCOP perspective that combines two different viewpoints **P** (for horizontal coordinates) and **V** (for vertical coordinates). In their case, however, the geometry of the display is significantly more complex, and computing the projection requires both a ray/plane and a ray/cylinder intersection per 3D point.

5.3.2 Depth Dependent Spatial Resolution

The display design has consequences not only on the projection equation but also imposes limits on spatial resolution that depends on depth. Each beam leaving the screen has (approximately) a finite angular size Φ. This, however, introduces a finite resolution effect – that is independent from the screen pixel resolution – in the reconstructed three dimensional scene. In fact, the size s of the smallest voxel that can be reproduced depends on the distance of its center from the screen and from the beam angular size Φ, and can be approximated by

$$s(z) = s_0 + 2\|z\| \tan(\Phi/2) \tag{5.2}$$

where s_0 is the pixel size on the screen surface (see Fig. 5.2 right). In other words, the achievable spatial resolution decreases with the distance from the screen. This is intuitive because the illusion of existence of a particular spatial point is generated by pyramidal beams crossing at a specific 3D position (see Fig. 5.2 right). This fact dictates how volumes should be sampled when rendering for the display and also practically limits the field-of-depth of the display, i.e., the maximum distance from the screen at which objects are faithfully reconstructed. Furthermore, we employ $s(z)$ to determine the level of detail at which volumes have to be sampled and the amount of blending for mipmapping by quadrilinear filtering (see section 3.2.1).

5.4 GPU-based Volume Ray Casting

We exploit the characteristics of the display to develop a specialized volume rendering technique able to provide multiple freely moving naked-eye viewers the illusion of presence of virtual volumetric objects floating at fixed physical locations in the display workspace. Our light-field display aware volume rendering process follows the two-pass approach typical of contemporary multi-projector displays [Brow 05] (see figure 5.4). In the first rendering pass, a per projector view of the scene is rendered off-screen to a frame buffer object using the idealized light-field geometry model. In the second pass, performed purely in 2D image space, small non-linear view and color distortions are corrected by streaming the first pass texture through a fragment shader that warps the geometry and modifies colors thanks to per-pixel look-up tables stored as precomputed textures.

Our 3D rendering framework is based on the real-time volume rendering engine, which has been presented in chapter 2 and 3. In this approach, per projector images of a volume are rendered by casting rays through each pixel, and performing re-sampling and compositing of

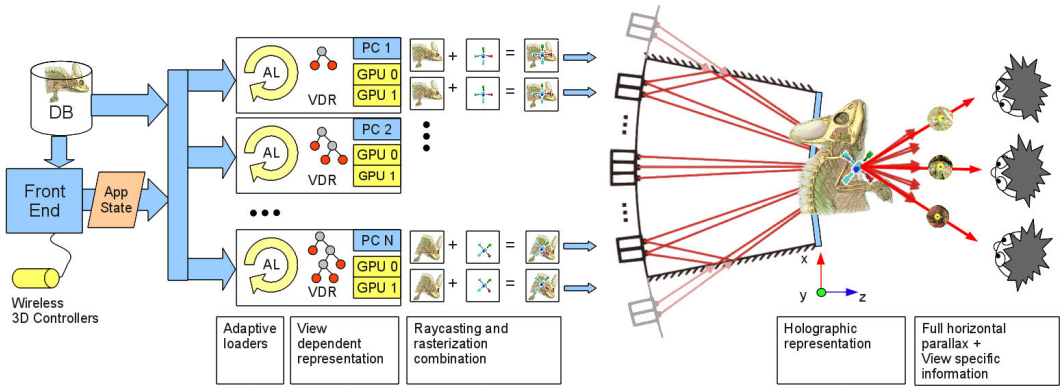


Figure 5.3: **Virtual environment concept**: Presented images combine ray-traced volumetric data with rasterized surface-based information, such as labels or user interface tools. Visualization is both view-dependent, since it provides full parallax, and view-specific, since it delivers different information in different areas of the workspace.

successive sample contributions along these rays. By applying an appropriate optical model many desired kinds of interaction between light and the volumetric object can be realized.

5.5 Implementation and Visualization Results

We have implemented a prototype hardware and software system based on the design previously discussed in this chapter.

5.5.1 Implementation of a Small-scale Prototype Using a DVI Channel

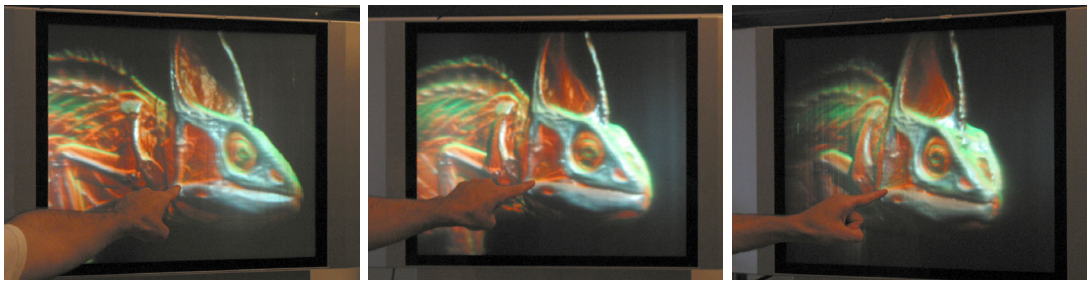


Figure 5.4: **Volume ray-casting results with the chameleon dataset**. Images of the light-field display taken with a hand held video camera moving in the display workspace. Note the parallax effects and the view-dependent illumination effects.

The software system proposed in this section consists of a framework written in C++ and OpenGL, a set of Cg shaders that implement the basic ray-casting engine, and a number of shader functions that implement different compositing techniques. The display hardware is manufactured by Holografika and is capable of visualizing 7.4M beams/frame by composing optical module images generated by 96 fast 320x240 LCD displays fed by FPGA input processing units that decode an input DVI stream. The on screen 2D pixel size of the display is $s_0 = 1.25mm$, and the angular accuracy is $0.8°$.

The DVI channel feeding the display works at 1280x1024 at 75Hz. Each 1280x1024 frame collects 16 320x240 projector images, plus a color-encoded header in the top rows that encodes the ids of the projectors that have to be updated. A full 3D frame is created by sequentially

generating all the projector images into the frame buffer. In this work, the graphics application runs on an Athlon64 3300+ PC with a NVIDIA8800GTX graphics board working in twin-view mode. One DVI output is used for control on a 2D monitor, while the second one feeds the 3D display. It is obviously impossible to fully convey the impression provided by our

Figure 5.5: **Selected frames from a live recording interaction session.** These images, taken from the accompanying video [1], show successive instants of interactive exploration of different CT datasets.

holographic environment on paper or video. As a simple illustration of our system's current status and capabilities, an accompanying video [1] shows interactive sequences recorded live using a moving camera. Representative video frames are shown in figure 5.5. The sequences were recorded with a hand held video camera freely moving in the display workspace. In order to assess the distortion caused by the MCOP approach, the video includes sequences presenting both vertical and horizontal motions, with the camera moving far from the virtual viewer position. We recorded short inspections and free-hand manipulation of different public domain datasets at the resolution of 256^3 voxels at 8bit/sample: The Visible Human Male head [2], the Chameleon CT scan [3], and a contrasted rotational angiography of a human head with aneurysm [4]. Currently, our prototype volume ray caster implements a number of

[1] See the online video at: *http://vic.crs4.it/vic/cgi-bin/multimedia-page.cgi?id='144'*
[2] Source: The National Library of Medicine, USA
[3] Source: Digital Morphology Project, CTLab and Texas Advanced Computing Center, University of Texas, Austin
[4] Source: Volume Dataset Repository at the WSI/GRIS, University of Tübingen, Germany

composition strategies, that include Maximum Intensity Projection (MIP), Simulated X-Ray, as well as Direct Volume Rendering with a Phong illumination model, boundary enhancement and view-dependent transparency [Bruc 07b]. For rendering, the volume is stored in a single 4 component texture that contains the unscaled precomputed gradient in the RGB part and the voxel value in the A part. An additional one-dimensional RGBA texture contains an interactively modifiable look-up table. In the accompanying video, the Chameleon and Visible Human head datasets are rendered using a shaded volume rendering, while the rotational angiography dataset employs a maximum intensity projection.

As demonstrated in the video, objects appear to moving viewers floating in the display space and can be manipulated by translating, rotating, and scaling them with a six degree of freedom tracker, as well as by modifying the transfer function. Note the parallax effects and the good registration between displayed object space and physical space, which demonstrate the multi-user capability of the display and the good performance of the MCOP viewing approach. Please also note that specular highlights correctly follow the recording camera's viewpoint, contributing to volume readability. As illustrated by the video, the perceived image is fully continuous. This is qualitatively very different from other contemporary multiview technologies, which force users into approximately fixed positions, because of the abrupt view-image changes that appear when crossing discrete viewing zones [Matu 04].

The main limiting factor during interaction was given by the single GPU volume renderer performance, since at full resolution (1 ray/pixel and 1 sample/voxel), many millions of rays need to be propagated for hundreds of steps through the volume. In order to improve interactive frame rates, we have thus chosen the solution of lowering the sampling rate during interaction, by reducing the pixel count (2.5mm precision on screen) and doubling the integration step-size. Since the display projectors are sequentially updated in batches of 16 320x240 images, a misalignment between tiles can become visible when objects are moved with a too slow refresh rate (see figure 5.6). It is important to note that even when a single static 3D view is displayed, users can exploit accommodation, stereo and motion parallax to gain understanding of complex shapes. Some of these cues can be also obtained with traditional systems, but only by incorporating interactive manipulation in the rendering system. In that case, users will have to move the object or the viewpoint to provide the visual system with enough depth information. The task is not simple and immediate, and depth information is easily lost when the user stops interacting.

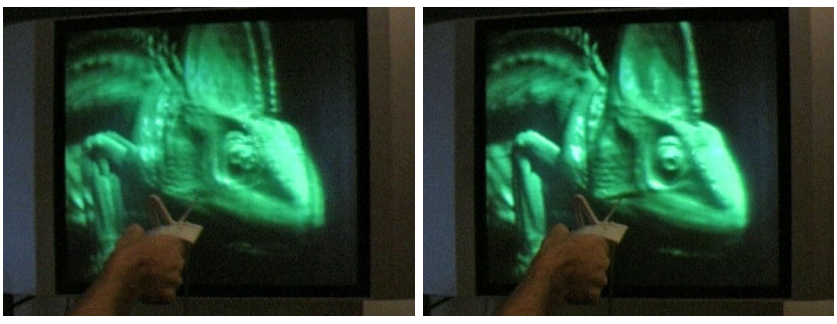


Figure 5.6: **Dynamic tile misalignment effect.** The 96 display projectors are updated in batches of 16 projectors each, which can lead to artifacts in dynamic motion if the rendering rate is too slow (left). These artifacts disappear when the image is static (right).

5.5.2 Implementation of a Large-scale Prototype Using a GPU-cluster

An experimental system has been implemented on Linux using OpenGL and NVIDIA CUDA 2.3. The out-of-core octree structure is implemented on top of Berkeley DB, with LZO compression applied to data nodes. OpenMPI 1.2.6 is used for inter-process communication in the rendering cluster. When running on a desktop configuration, the rendering kernel is embedded directly in a single-process application, using a separate thread for asynchronous data loading. We have tested our system with a variety of high resolution models and settings.

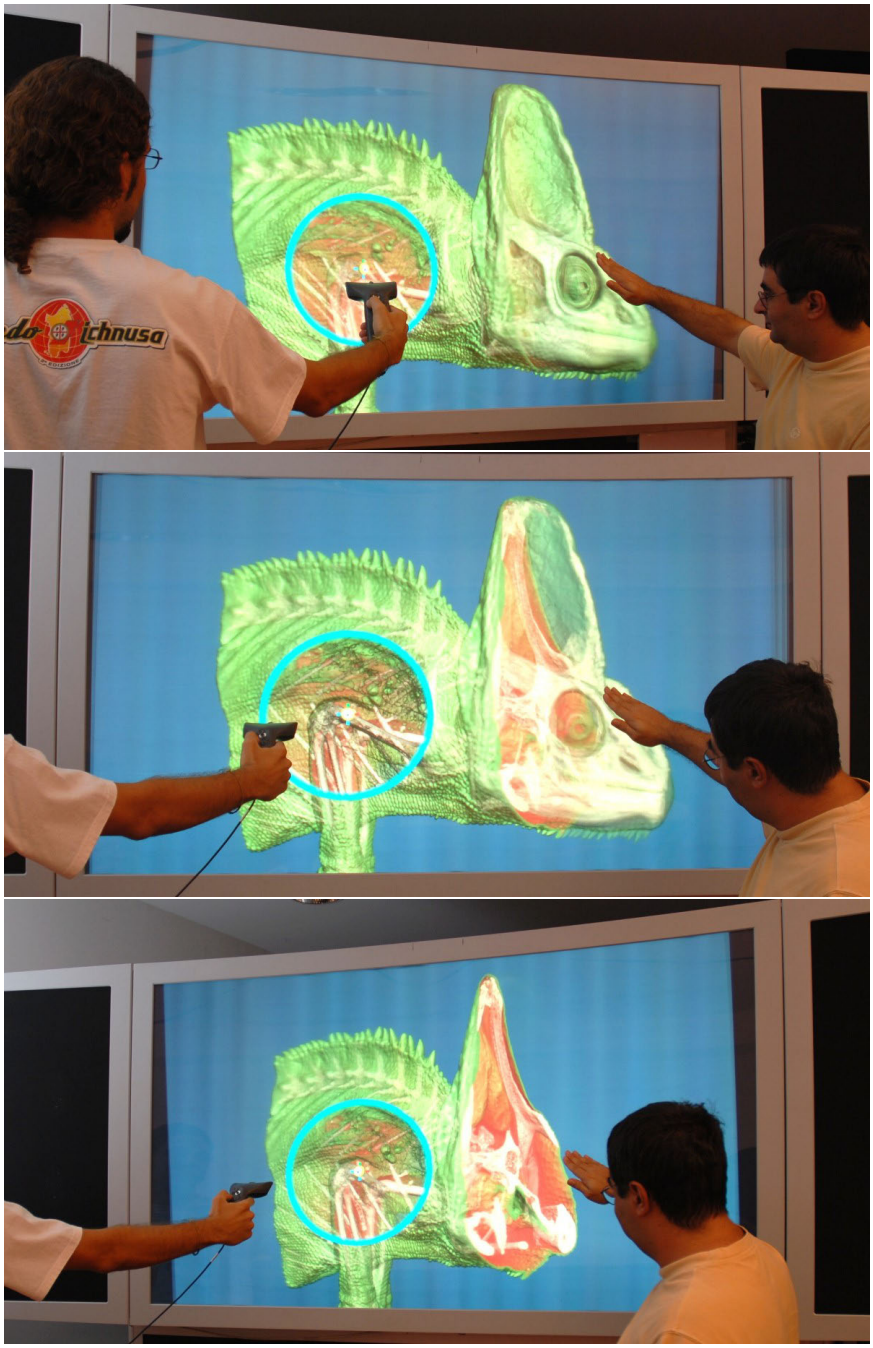


Figure 5.7: **Multi-user interactive exploration of a 64-GVoxel dataset on a 35-MPixel light-field display.** Users freely mix and match 3D tools creating view-dependent illustrative visualizations. Objects appear floating in the display workspace, providing correct parallax cues while delivering direction-dependent information.

In this section, we discuss the results obtained with the inspection of two 16bit/sample micro-CT datasets: a Veiled Chameleon ($1024 \times 1024 \times 1080$) and a Pichi Armadillo ($1024 \times 1024 \times 999$) specimens [5]. In order to test the system with extremely large and demanding datasets, the original data has been artificially zoomed by $4\times$ in a preprocessing step (i.e., to $\approx 4K^3$ voxels) by tricubic interpolation plus procedurally added detail. The construction of the octree from source data was performed using octree bricks of 32^3, instructing the preprocessor to use the compressed 8bit format and to filter out data with a value lower than 6400, in order to discard most of the noisy empty space. For the chameleon, data preprocessing took 12.3hours on a Linux PC 2.4GHz CPU and produced a 15GB octree database starting from an uncompressed source size of 135GB. The RMS error was 14.6, and the AMAX error was 116. The armadillo dataset produced similar results (13.8GB octree database, $RMS = 15.8$, $AMAX = 122$).

Our 3D display is capable of visualizing 35-MPixels by composing images generated by 72 SVGA LED commodity projectors illuminating a 160×90cm holographic screen. The display provides continuous horizontal parallax within a $50°$ horizontal field-of-view with $0.8°$ angular accuracy. The pixel size on the screen surface is 1.5mm. The rendering back-end is currently running on an array of 18 Athlon64 3300+ Linux PCs equipped with two NVIDIA 8800GTS 640MB (G80 GPU) graphics boards running in twin-view mode. Each back-end PC has thus to generate $4 \times 800 \times 600$ pixels using two OpenGL graphics boards with CUDA capability 1.0 and based on an old G80 chip. It is obviously impossible to fully convey the impression provided by our system on paper or video. As a simple illustration of our system's current status and capabilities, we analyze the behavior of the multi-user application that allows users to interactively position, scale, and explore volumetric models using Intersense IS900 trackers. We recorded the performance using a hand-held video [6] camera freely moving in the display workspace. Representative video frames are shown in Fig. 5.7 and Fig.5.8. Note the parallax effects, the view-dependent illustrative visualizations, and the good registration between displayed object space and physical space, which demonstrate the multi-user capability of the display. Thanks to stereo and motion parallax, MImDA images proved easy to understand on the 3D display even though they provide mixed depth cues (See Fig. 5.9). The various view-dependent illustrative tools proved intuitive to use and able to reveal important volumetric details. Combining view dependent rendering mechanisms with stereo projections inherently leads to the issue of presenting two images to the viewer that differ by more than just the view point. However, we did not notice problems related to this fact. This is because our visualizations have a strong correlation between views, and subtle differences are extremely well tolerated by the human visual system. Even though the rendering hardware is definitely not high end, the application remained interactive for large models. The average frame rate during the recorded interaction sequences was of 9.8fps. During the entire inspection sequences, the resident set size of each rendering process is maintained well within the 600 MB/board of pre-allocated cache size by the out-of-core data management system. The average octree size in the render nodes was of 1549 bricks/process, while the most loaded process handled 2087 bricks. Thus, introducing a load balancing strategy could (slightly) improve the system's performance by $\approx 30\%$. The good load balancing achieved by the static image distribution is caused by the geometry of the display, with all

[5] Source: Digital Morphology Project, the CTLab and the Texas Advanced Computing Center, University of Texas, Austin

[6] See the online video at: *http://vic.crs4.it/vic/cgi-bin/multimedia-page.cgi?id='152'*

projectors typically looking at the same portion of the displayed object. The visibility feedback mechanism has proved to be able to reduce working set size, especially when using transfer functions with moderate to high opacity. It should be noted that the graphics boards used in our cluster are two generations older than the one used for desktop testing, leading to a noticeable slowdown, that we have quantified to $2.3\times - 3.2\times$, the higher figures being for the most demanding situations.

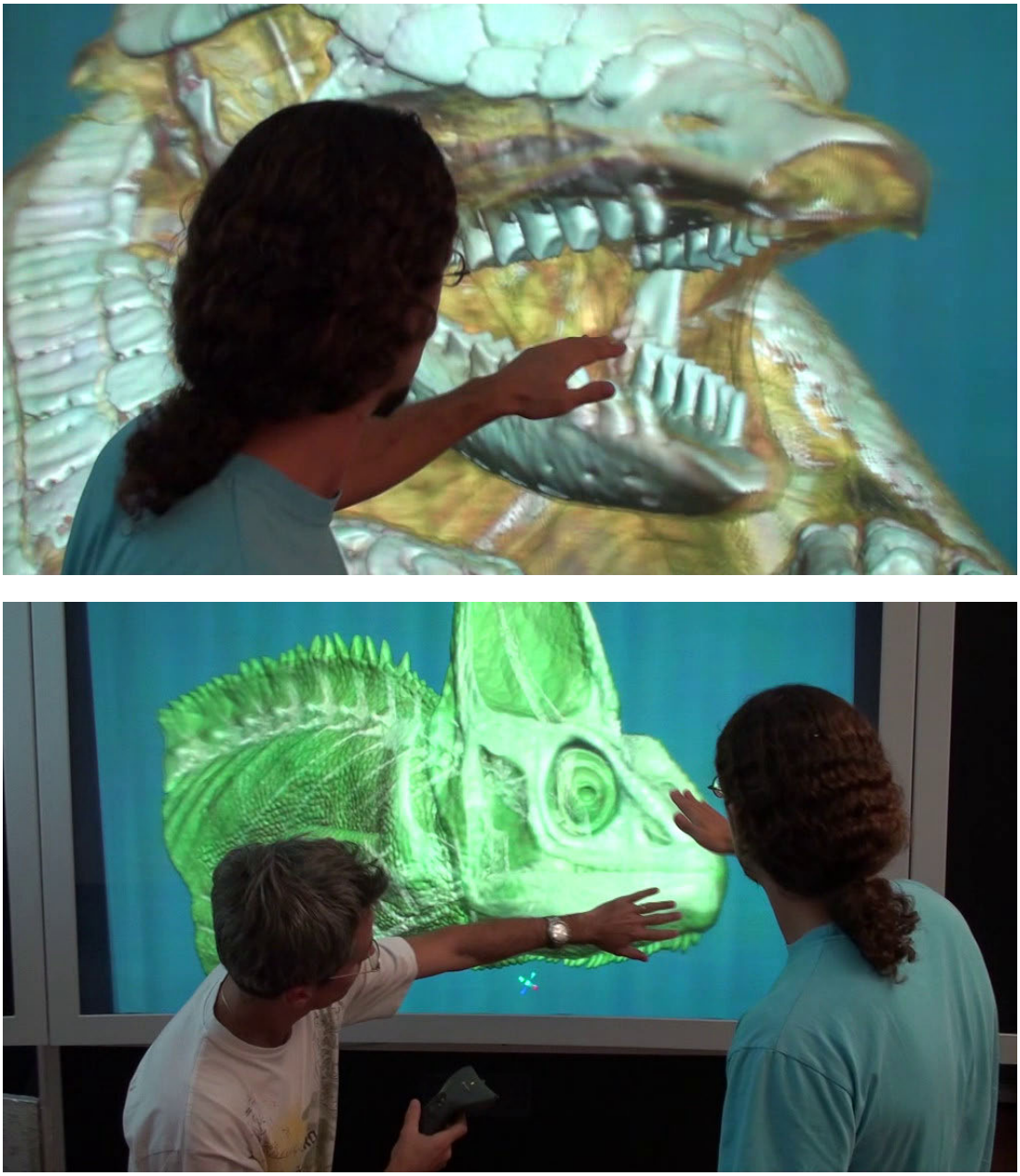


Figure 5.8: **Multi-user interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display.** Users freely mix and match 3D tools creating view-dependent illustrative visualizations. Objects appear floating in the display workspace, providing correct parallax cues while delivering direction-dependent information.

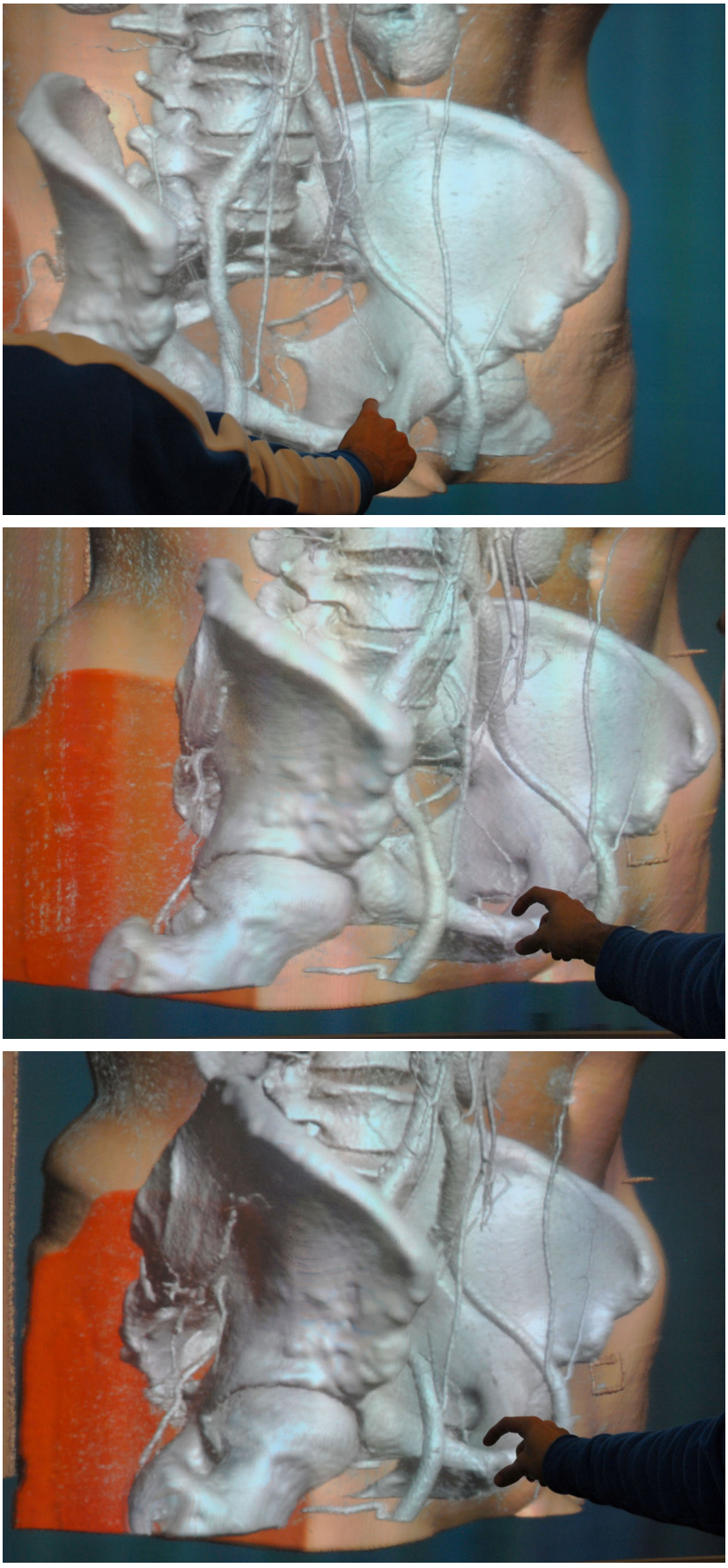


Figure 5.9: **Exploration exploiting the MImDA accumulation scheme with medical data.** The MImDA accumulation scheme is exploited to enhance the bone structures over the skin and flesh tissues.

5.6 Perceptual Evaluation

The human visual system makes use of a wide variety of cues in understanding a 3D scene. The goal of the experiments described in this section is to show that the light-field display enhances understanding by providing cues not available in traditional display systems. To quickly evaluate whether the light-field rendering infrastructure is able to help in rapidly gaining understanding of complex 3D shapes we performed a series of simple experiments. Those tests are not fully formal tests, in the sense that they could be more extensive and exhaustive but even with such limitations they can already advanced some interesting conclusions.

With respect to the image generation, we focused on a order-independent rendering context, i.e., on depth-oblivious techniques that do not provide occlusion cues. Such techniques are frequently found in medicine, Maximum Intensity Projection (MIP) being the most common one. It is a simple variant of direct volume rendering, where, instead of composing optical properties, the maximum value encountered along a ray is used to determine the colour of the corresponding pixel [Mora 05]. MIP is considered very useful for displaying structures that have attenuation higher than those in the neighbourhood, such as contrast enhanced vessels and ureters, and it is thus the option of choice for CT angiography and CT urography. However, it does not provide depth information, and, thus, in normal 2D displays users cannot reliably detect the 3D relationships of depicted structures. As a matter of example,

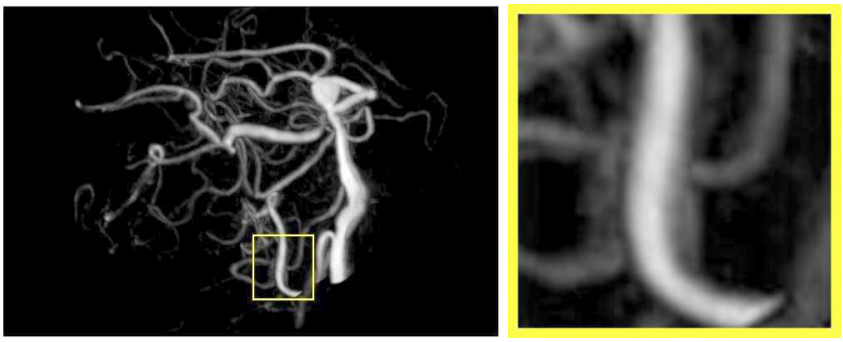


Figure 5.10: **Depth oblivious MIP angiography rendering.** MIP volume rendering of a rotational angiography scan of a head with aneurysm. The positions and the crossings of vascular structures are not detectable, or wrongly interpreted.

we can consider the analysis of a rotational angiography scan of a head with aneurysm (Fig. 5.10). In a 2D view, the positions and the crossings of vascular structures can be wrongly interpreted. For instance, in figure 5.10 we can see two crossing blood vessels, with the one at the front having an intensity less than the one at the back. From this point of view, the back one is visible at the intersection and hides the front vessel because of its higher intensity, providing a wrong impression. It has already been shown that understanding can be improved by presenting results on displays eliciting more depth cues than conventional 2D monitors [Mora 04, Bouc 09, Agus 09]. Instead, in this thesis, the questions we are considering are:

- *Is continuous multiview technology able to provide depth cues needed for layout discrimination in the context of order independent rendering? What are, if any, the differences with respect to standard disparity-based stereo systems?*
- *Is the viewing space discretization a critical factor for depth discrimination? Does continuous parallax provide advantages for typical spatial understanding tasks?*

To answer these questions, we did a series of evaluation tests employing order-independent rendering: a typical layout discrimination perceptual test, with stimuli containing various confusing perceptual hints, and a path tracing performance test, to measure how much horizontal parallax provided by continuous multiview displays helps in the context of the interpretation of tree structures.

5.6.1 Depth Cues Analysis

The software system consists of an integrate applications built around a multi-resolution volume processing and rendering framework written in C++ and OpenGL, a set of Cg shaders that implement the basic ray-casting engine, and a number of shader functions that implement different composing techniques. Data loading and DICOM connectivity is implemented using the OFFIS DCMTK library allowing the access to standard radiology PACS.

The system discussed in this work uses a 26" HoloVizio display by Holografika. The display hardware employed here is capable of visualizing 7.4M beams/frame by composing optical module images generated by 96 fast $320x240$ LCD displays fed by FPGA input processing units that decode an input DVI stream. The display is fed by 4 DVI channels working at 1280x1024 at 75Hz. Each 1280x1024 frame collects 16 320x240 projector images, plus a color encoded header in the top rows that encodes the ids of the projectors that have to be updated. A full 3D frame is created by generating all the projector images into the frame buffer. This is obtained by employing a solution in which each DVI channel is fed by a graphic node (PC equipped with a NVIDIA 8800 GTX graphics board), which is charged with filling a subset of the projectors. The on-screen 2D pixel size of the display is $s_0 = 1.25mm$, and the angular accuracy is $0.8°$. The screen width is $L = 500$mm. and its horizontal viewing angle is $\theta = 50°$.

First of all, in order to demonstrate that the light-field display is able to provide correct and reliable stereo cues, we replicated the tests performed by Kersten et al [Kers 06]. We found that the light-field system is able to provide appropriate stereoscopic rendering. Eleven subjects obtained success rates in the discrimination of the rotation direction of a cylinder filled with Perlin noise equal to 83% for MIP visualization and 91% for X-RAY visualization over 10 trials. However, ideally the contributions that disparity-based cues make to depth perception should be measurable in the absence of those cues to 3D that are usually available in 2D displays (e.g. object occlusion, surface shading, perspective foreshortening and texture gradients) or other stereo based display systems. Yet it is difficult to find naturalistic scenes in which disparity based cues exist in isolation. We thus decided to resort to an artificial technique employed in the field of visual psycho-physics to generate scenes that mask out those visual cues that are not disparity based. Specifically, we designed a synthetic benchmark using random-dot masking in a simplified version of Julesz's spiral ramp surface: a 3-layer cylindrical wedding cake model that subjects viewed along its concentric axis (see Fig. 5.11 right). The random dot textures block the usual pathways along which vision proceeds: the locations of surface boundaries within the scene are lost in the "cloud of dots". Vision can then only make sense of the scene after achieving binocular fusion, which gradually reveals the correspondence between the random dots as seen from two different viewpoints, e.g. the left and right eyes. The expectation here is that on the light-field display such a scene will be seen almost instantaneously, thus demonstrating the added value that it brings. By adjusting the model's parameters and converting it to a rectilinear volume, two sets of model stimuli were rendered: one with a uniform large field of depth ($\pm 10cm$ centered on the display screen) and one where

the field was almost flat ($\pm 1cm$). Eleven, pre-screened, subjects completed four experiments, each consisting of eight trials in a two-interval forced-choice (2IFC) design whereby they indicated in which interval they perceived the greatest field of depth. The experiments tested one eye static, one eye head-swaying, two eye static, and two eye head-swaying observation in that order. Scores improved also in that order: from 49%, i.e., indistinguishable from a random answer in the binary test, with a monocular static view, to 82% correct scores for the monocular head-swaying test, up to 100% when all cues are available. The results indicate that in the absence of cues normally available in 2D displays the light-field rendering system elicits useful stereoscopic and motion parallax depth cues.

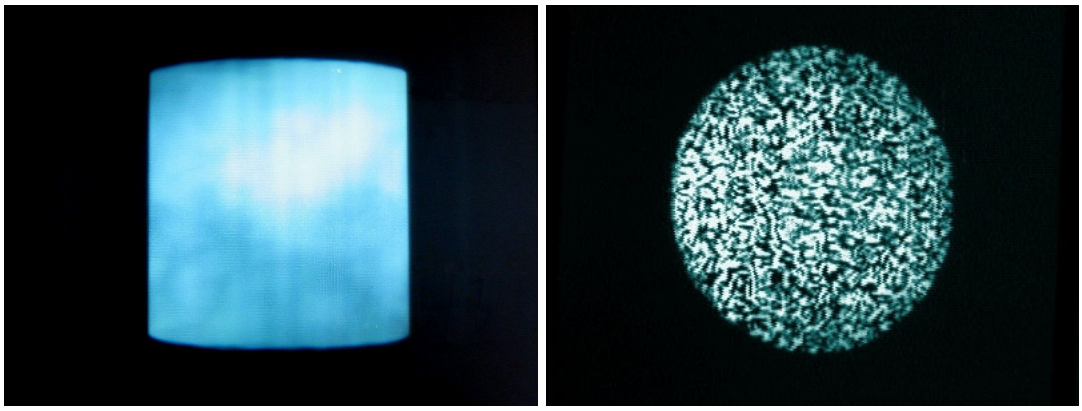


Figure 5.11: **Depth cues evaluation tests**: left: cylinder filled with Perlin noise; right: 3-layer cylindrical wedding-cake model with random-dot texture.

5.6.2 Layout Discrimination Performance

The large scale continuous multiview display employed for tests can visualise 35-MPixels by composing images generated by 72 SVGA LED commodity projectors illuminating a 160×90cm holographic screen. The display provides continuous horizontal parallax within an approximately 50° horizontal field-of-view, with 0.8° angular accuracy. The pixel size on the screen surface is 1.5mm. The same continuous multiview system is customised to simulate discrete systems by employing the observer space discretization described in Fig. 5.14. In this work, we evaluated various discrete multiview configurations obtained by increasing the view width between adjacent views, in order to verify whether this factor influences depth perception capabilities.

5.6.2.1 Stereo vs Horizontal Parallax

We first evaluated the motion parallax effect provided by continuous multiview technology with respect to depth discrimination. We proved that this technology is more effective in eliciting depth cues than stereoscopic displays, especially in conjunction with depth-oblivious techniques, such as maximum intensity accumulation. We verified this assertion by carrying out a series of psycho-physical tests. We used a typical 2 forced-choice (2FC) psycho-physical discrimination task, where 10 pre-screened subjects were asked to indicate the closest one between two partially overlapping disks, rendered in red and blue (Fig. 5.12). Viewers were located at 120cm from the screen. For a given trial, the two disks were assigned a depth of $\pm D$ with respect to the screen plane and placed amid other disks at varying depths. The

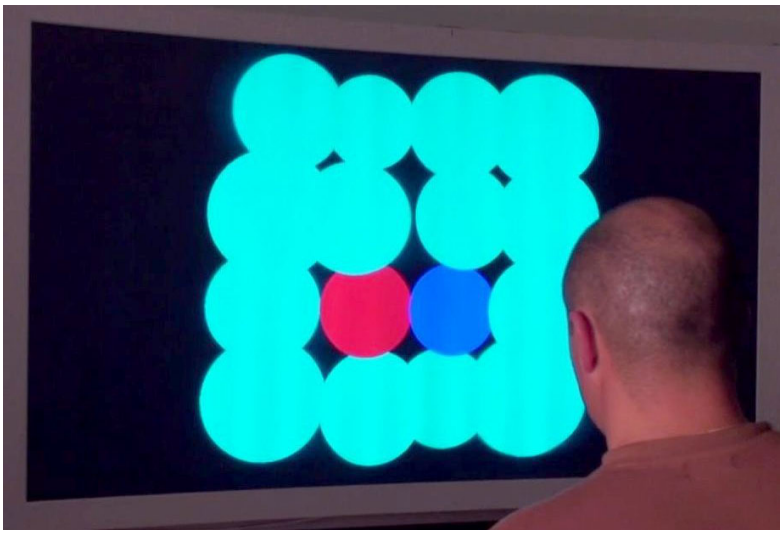


Figure 5.12: **Disks discrimination test.** Subjects were asked to discriminate in depth between two disks, red and blue, partially overlapping and symmetrically displaced with respect to the screen. Maximum intensity is employed to provide confusing false occlusion.

entire scene was rendered using maximum intensity projection, with intensity unknown to the viewers. Other confusing hints were added in order to eliminate potential biases coming from other cues: relative sizes, false occlusions and colours. We first performed the tests using two different display configurations: a two-view stereoscopic setting and a full continuous horizontal parallax setting. All the display configurations were obtained with the large scale display by suitably constraining the virtual observer position.

Results We report on statistical analysis of hit rates obtained by 10 subjects for the considered depth distances and viewing conditions.

D(mm)	Hit Rate(S)	Hit Rate(C)	p
5	0.53 ± 0.03	0.59 ± 0.03	0.17
10	0.55 ± 0.03	0.71 ± 0.02	< 0.001
20	0.67 ± 0.04	0.79 ± 0.02	0.02
50	0.81 ± 0.03	0.91 ± 0.02	0.09
100	0.95 ± 0.02	0.96 ± 0.02	0.55

Table 5.1: **Statistical results for disk depth discrimination test.** The first column contains the depth differences considered, while the second and third contain mean hit rates scored by 10 subjects over 10 trials with disparity based stereo (S) and continuous multiview horizontal parallax (C). Last column contains the results of ANOVA with respect to the viewing condition.

Table 5.1 contains numerical results for five depth differences and two viewing conditions. Specifically, first column contains the depth differences considered, while the second and third contain mean hit rates together with standard errors scored by the 10 subjects over 10 trials with disparity based stereo (S) and continuous multiview horizontal parallax (C). Last column instead contains results of analysis of variance with reference to the viewing condition. ANOVA clearly shows that there is no significant effect of viewing condition for depth differences 5mm and 100mm ($p > 0.1$), while a main effect is experienced for depth differences 10mm, 20mm and 50mm ($p < 0.1$). This fact, together with the mean hit rates values, suggests us the main depth ranges for discrimination: above 50 mm stimuli are almost

perfectly recognisable for each viewing condition, under 10 mm stimuli cannot be recognised for each viewing condition. In the range between 10 mm and 50 mm, there is difference between disparity based stereo and continuous parallax, suggesting that with continuous parallax the JND (just noticeable difference) is smaller.

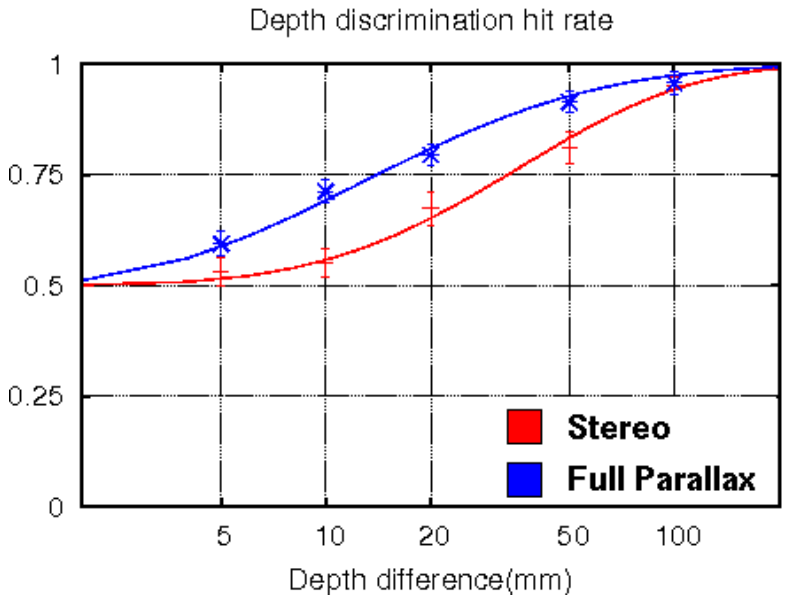


Figure 5.13: **Stereo vs parallax disks discrimination results.** Hit rates (standard error bars and psychometric fit) obtained with stereo and full parallax display configuration.

Fig. 5.13 shows mean hit rates and standard error bars on a logarithmic scale, together with the psychometric function fits obtained by employing the psignifit package [Wich 01a, Wich 01b]. The psychometric function fits were obtained by considering the following function:

$$\Psi(x;\alpha,\beta,\gamma,\lambda) = \gamma + (1-\gamma-\lambda)F(x;\alpha,\beta) \tag{5.3}$$

where γ gives the lower bound of Ψ, and can be interpreted as the base rate of performance in the absence of a signal, while λ is the upper bound of the psychometric function representing a reflection of the rate at which observers lapse, responding incorrectly regardless of stimulus intensity. In the case of depth discrimination tasks, the Weibull distribution was employed:

$$F(x;\alpha,\beta) = exp(-(\frac{x}{\alpha})^{\beta}). \tag{5.4}$$

The psychometric functions represented in Fig. 5.13 were obtained by considering as stimulus value $x = ln(D)$. Considering statistical results and psychometric function fits obtained with the disk discrimination test, it appears evident that continuous multiview provide a better discrimination with respect to binocular stereo viewing. Threshold levels are significantly different ($\Psi_C^-1(0.75) = 14mm < \Psi_S^-1(0.75) = 32mm$), clearly showing a sensible perceptive improvement for continuous multiview.

5.6.2.2 Evaluating Discrete Multi-view Designs

After showing that continuous parallax is a decisive factor for depth discrimination, we focused on the analysis of discrete multiview designs. The target of this evaluation work is to compare continuous and discrete multiview display technologies with reference to depth

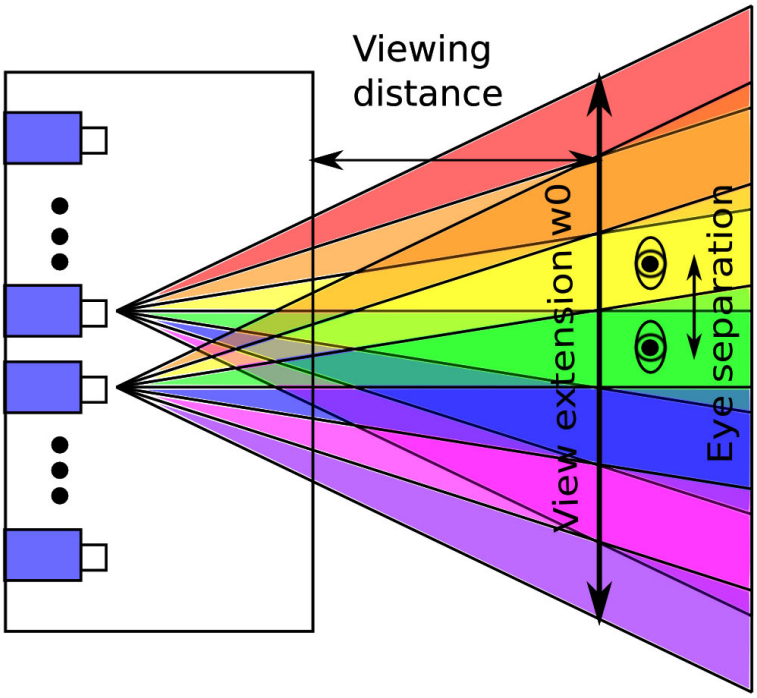


Figure 5.14: **Discrete multiview scheme.** the general discrete multiview with parameters related to equation 5.7: each zone is represented by a different colour, inner zones are shared among different projectors [Dodg 02].

discrimination cues involved by display geometry differences. To simulate discrete designs it is possible to conceptually divide the user viewing space into a finite number of windows, called viewing zones, in each of which only a single image is visible, while still retaining both stereo and movement parallax cues. To this end, the multiple-center-of-projection equation can be simplified by reducing the number of views provided. Referring to equation (5.1), the ray origin can be corrected according to the geometry design of the display to be simulated. For example, considering a standard stereo display, only two views have to be generated and the ray origin abscissa of equation 5.1 needs to be corrected in the following way:

$$O_x = \begin{cases} -\delta & \text{if } O_x < 0 \\ +\delta & \text{if } O_x > 0 \end{cases} \tag{5.5}$$

where δ is the average half inter-pupillary distance (about 35 mm). In the same way, by adequate quantisation of the observer space, the correction can be applied to simulate generic discrete multiview designs. Considering N not overlapping viewing zones having center γ_i and width ρ_i, the correction equation is the following:

$$O_x = \begin{cases} \gamma_0 & \text{if } O_x < \gamma_0 + \rho_0 \\ \gamma_i & \text{if } |O_x - \gamma_i| < \rho_i \\ \gamma_{N-1} & \text{if } O_x > \gamma_{N-1} - \rho_{N-1} \end{cases} \tag{5.6}$$

As indicated by Dodgson [Dodg 02] the viewing zone widths ρ_i at the viewing distance should be less than the inter-pupillary distance σ so that in each position the two eyes can perceive

two different images, thus producing the stereoscopic effect (see Fig. 5.14). If we have N views which subtend an extension w_o the previous relation can be expressed as

$$w_o < N \times \sigma \tag{5.7}$$

If inequality is not satisfied there will be some zones where users lose depth perception, because both eyes are hit by the same image. Thus, in a normal usage of multiview system, users stay at fixed positions instead of exploiting parallax by ego-motion, and tend to avoid to interfere with zones where the stereoscopic effect is unstable, thus producing annoying artifacts. As we can see in Fig. 5.14, each projector splits its image in a certain number of viewing zones, highlighted here with different colours. In areas where different colours interfere, users perceive a wrong image, while at the correct viewing distance, images match correctly without introducing any unwanted interference.

A multiview design can be obtained by reducing the number of views in two ways: first, by reducing the working zone width w_o (see equation (5.7)), resulting in a reduction of the field of view. In this case, image quality is preserved and depth cues are maintained with the cost of a limited working zone. However, in this way the potential of multiview technology is reduced, since the display can be used by a limited number of users. The other method for reducing the number of views consists instead of maintaining the full field of view and increasing the distance ρ between adjacent views (see equation (5.6)). We evaluated the depth discrimination capabilities of the multiview technology with respect to this factor, by considering a typical scenario of a user positioned at distance $Z = 1200mm$ and observing a scene centered at $D = 100mm$ from the screen. Three different viewing widths were considered: $\rho = 60mm$ corresponding to 12 views, $\rho = 30mm$ corresponding to 24 views and $\rho = 10mm$ corresponding to 72 views, the last one being equal to the maximum angular resolution obtainable with the light-field display considered.

Qualitative evaluation We first carried out a qualitative evaluation in order to prove that a limited number of views degrades the image quality thus resulting in annoying artifacts. In a preliminary analysis of a scene with a depth complexity of $D = 100mm$ (we employed the same scene employed for depth discrimination task with an offset of D along z direction), 10 subjects were asked to indicate if transition artifacts were perceived during motion, for the three view widths considered. All subjects experienced annoying artifacts for $\rho = 60mm$, the same artifacts dramatically reducing when $\rho = 30mm$. This fact suggests that $\rho = 30mm$, corresponding to about half the inter-pupillary distance, is the inferior limit for a discrete system that is supposed to provide a compelling 3D experience, and a smooth transition between views during motion.

Perceptual evaluation Once assumed that image quality depends on the number of views employed, we carried out a perceptual analysis to evaluate whether a limited number of views degrades also the depth discrimination performance, or whether horizontal parallax cue is correctly provided even with a reduced number of views. To this end, we considered the same disk discrimination test employed in subsection 5.6.2. The same 10 subjects were asked to discriminate the depth of the two coloured disks for the three view width configurations ($\rho = 60mm$, $\rho = 30mm$, and $\rho = 10mm$)

$\rho(mm)$	D(mm)	Hit Rate $\pm$ **SE**
60	100	0.93 ± 0.03
	50	0.76 ± 0.05
	20	0.71 ± 0.04
	10	0.54 ± 0.04
30	100	0.93 ± 0.02
	50	0.88 ± 0.04
	20	0.72 ± 0.05
	10	0.57 ± 0.08
10	100	0.93 ± 0.04
	50	0.9 ± 0.04
	20	0.81 ± 0.06
	10	0.7 ± 0.05

Table 5.2: **Statistical results for disk depth discrimination test: comparison of three different multiview configurations.** The first column contains the multiview configurations considered, while the second column contains the depth differences employed and the third column contains the mean hit rates scored by 10 subjects over 10 trials together with standard errors.

Results Table 5.2 summarises numerical results of hit rates obtained by the subjects for four different depth differences and the three different viewing conditions:$\rho = 60mm$, $\rho = 30mm$, and $\rho = 10mm$. As expected, results obtained for $\rho = 10mm$ are very similar of those obtained during first perceptual test in the case of full horizontal parallax (see table 5.1). Furthermore, it appears evident that starting from disk depth differences below 50 mm, error rates for $\rho = 60mm$ are considerably higher with respect to the other viewing conditions, while from depth differences below 20 mm, error rates for $\rho = 30mm$ are considerably higher with respect to those obtained with $\rho = 10mm$. This clearly shows that the number of views effects the depth discrimination capabilities of the system. An ANOVA with respect to the viewing condition was also performed and indicated that there is a main effect for depth differences under 50 mm. In order to highlight whether hit rates obtained with the different view conditions are statistically different, a Tukey post-hoc test was also performed, revealing a significant difference between view width $\rho = 30mm$ and $\rho = 10mm$ for depth differences below 20 mm ($p < 0.01$).

Finally, the Fig. 5.15 represents the hit rates with error bars on a logarithmic scale, together with the psychometric function fits obtained by employing the psignifit package [Wich 01a, Wich 01b]. Fits were obtained considering a Weibull distribution, with $x = ln(D)$. Threshold levels are significantly different ($\Psi^{-}1(0.75) = 13.4mm$ for $\rho = 10mm$, $\Psi^{-}1(0.75) = 24.2mm$ for $\rho = 30mm$, and $\Psi^{-}1(0.75) = 35.4mm$ for $\rho = 60mm$), clearly indicating that the number of views dramatically effects depth discrimination capabilities.

5.6.3 Performance Evaluation

Another way to quantify the performance of multiview technologies consists of investigating whether it improves the understanding of network structures.

Description To this end, we considered a perceptual test where users were asked to trace complex paths. Specifically, 10 subjects observed a scene composed by a number of white polylines paths randomly placed, each one containing a number of segments connected by green colour spheres (see Fig. 5.16). The paths were rendered employing maximum intensity

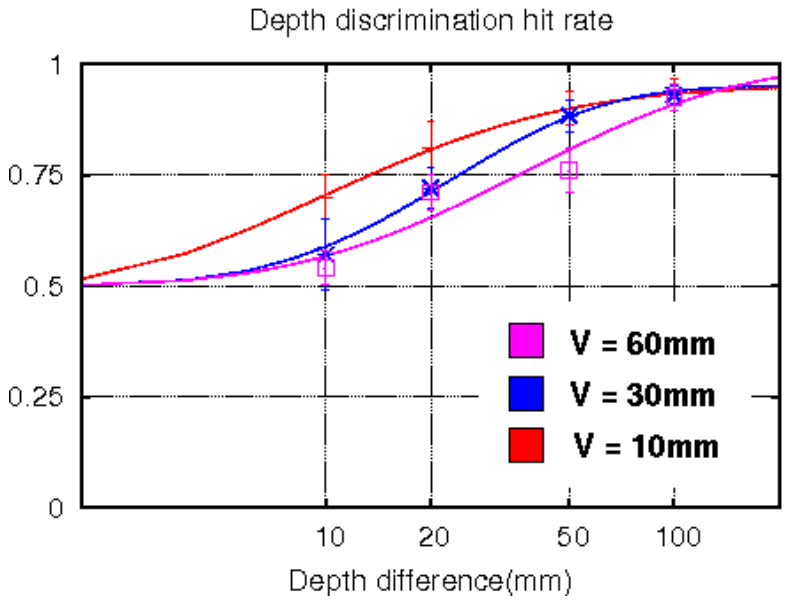


Figure 5.15: **Discrete multiview depth discrimination results.** Hit rates (standard error bars and psychometric fit) obtained with different discrete multiview configurations: $\rho = 60mm$, $\rho = 30mm$ and $\rho = 10mm$.

projection, with random intensity values. A 2FC design was considered, where subjects were requested to find the only polyline having red dots at their ends, and to count how many segments it contained (two or three). The following conditions were considered: four scene complexities ranging from 100 to 1000 nodes, and two display settings (continuous multiview, and disparity-based stereo). Graphs were generated without layout optimisation, with consequent difficulties for interpretation. Only geometric constraints were considered as to force segments lengths inside the range $[0.05, 0.5]\frac{d}{l}$, where d is the scene bounding box diagonal, and l is the number of segments for a given path.

N	Hit Rate (S)	Hit Rate (C)	p
100	0.94 ± 0.03	0.96 ± 0.02	0.79
300	0.88 ± 0.04	0.95 ± 0.02	0.30
500	0.66 ± 0.05	0.89 ± 0.02	$< 10^{-3}$
1000	0.55 ± 0.04	0.78 ± 0.04	$< 10^{-3}$

Table 5.3: **Statistical results for network interpretation performance test.** The first column contains the number of nodes in the graphs, while the second and third ones contain mean hit rates scored by 15 subjects over 10 trials with disparity based stereo (S), and continuous multiview horizontal parallax (C). Last column contains results of analysis of variance with respect to the viewing condition.

Results Table 5.3 summarises numerical results of hit rates obtained by subjects for different graph sizes (in terms of number of nodes) and different viewing conditions: stereo(S), and continuous multiview(C). Even in results of this specific test it appears evident that continuous (C) multiview provides performance improvements in terms of mean hit rates with respect to binocular stereo (S). The ANOVA on viewing condition also highlights that hit rates are statistically different for graph sizes starting from 500 nodes, indicating that for graphs under 300 nodes interpretation is considered easy independently from viewing condition, while the effect of multiview is perceived for graphs having bigger size. It is interesting to

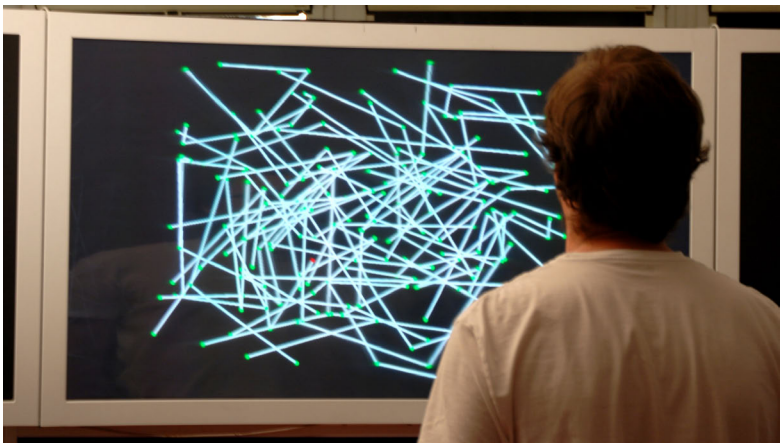


Figure 5.16: **Path tracing test.** Subjects were asked to find paths marked by red ends and to indicate whether they are composed by two or three segments. Maximum intensity is employed to provide confusing false occlusion.

note that with multiview technology, subjects are able to discriminate graphs with a number of nodes similar to those reported by Ware and others [Ware 08], even if our scenes were generated without considering any optimisation layout techniques [Herm 00]. Figure 5.17 plots

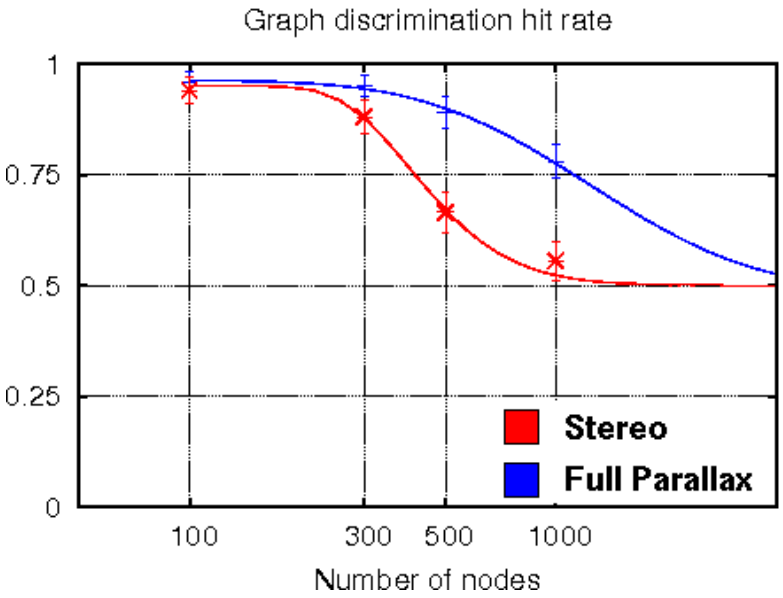


Figure 5.17: **Discrimination graph understanding results.** Hit rates (standard error bars and psychometric fit) obtained with different display configurations: stereo and continuous multiview.

hit rates together with error bars and psychometric function fits obtained with psignifit package [Wich 01a, Wich 01b]. Even in this case, the stimulus value is the graph size expressed on a logarithmic scale $x = log(N_{max}) - log(N)$ where $N_{max} = 10000$, and the psychometric function is assumed to follow the Weibull distribution. Graphs clearly highlight how horizontal parallax cues provided by multiview systems greatly help in graph understanding tasks.

5.7 Conclusion

The prototype discussed here is clearly meant to work as an enabling technology demonstrator, as well as a testbed for integrated volumetric rendering and light-field display research. A first conclusion than can be drawn from our work is that high quality volumetric rendering on light-field displays is currently achievable even when using single GPU desktop solution for the rendering task. In order to deal with large scale datasets, we have integrated a flexible multi-resolution volume rendering system capable to interactively drive large scale multi-projector light-field displays which can provide stereo and motion parallax cues in a setting supporting collaborative use. Continuous multiview technology provides stereo and motion parallax cues in a way that supports natural 3D vision and easy collaboration between users. In this chapter we reported on preliminary results of a series of evaluation tests aimed to compare the depth discrimination capabilities of discrete and continuous multiview systems. Depth discrimination tests indicate that parallax provides more depth cues, thus leading to a smaller just noticeable difference (JND) with respect to binocular stereo. However, further investigation is needed to find the precise JND, that we suspect to be related to the display characteristics (resolution and calibration), and to the observer distance. Furthermore, we compared performances for various discrete multiview designs. Our results indicate that the number of views affect the quality of experience (QoE) as well as depth discrimination performances. As a conclusion, it seems that extremely ego-motion is particularly helpful in path tracing tasks, and that this cue is delivered effectively by a continuous multiview design. As future work, we plan to perform an evaluation of the multiview technology in specific application contexts, such as diagnostic tasks or surgery planning in medicine. There is obviously more to interactive volumetric rendering on 3D displays than employing standard composition techniques. For example, it would be worthwhile to further investigate how to improve rendering speed by incorporating optimized algorithms that can exploit the high degree of coherence present in a 3D view.

5.8 Bibliographical Notes

The major part of the content of this chapter was based on paper [Agus 08c], where we presented a GPU-accelerated volume ray casting system interactively driving a multi-user light-field display. The content of the latest validation results came from reference [Agus 10a], where we presented a set of perceptual experiments completed to evaluate the depth discrimination capabilities with respect to two-view (stereo) and discrete multi-view designs. Previous results on the validation and more information about rendering applications on light-field displays can be found on [Agus 08a, Agus 08b, Agus 09], where we reported on a prototype system for medical data visualization and demonstrated the usefulness of the generated depth cues and the improved performance in understanding complex spatial structures with respect to standard techniques. Additional information regarding the scalability of the technique is available in [Igle 10], where we demonstrated the possibilities of the approach by the multi-user interactive exploration of massive volume datasets on a 35-MPixel light-field display driven by a cluster of PCs. More general considerations about human-computer interaction and elements affecting these kinds of virtual reality systems can be found in [JAIg 08].

Chapter 6

Illustrative Techniques

In this chapter, we report on a set of illustrative and non-photorealistic rendering (NPR) techniques that complement the work presented in chapters 2, 3 and 5. In the field of context-preserving volume rendering we propose an illustrative technique, named *context-preserving focal probes*, to focus the attention of the users in a region of interest while preserving the information in context. The focus and context information are separated by the assignment of different rendering styles that can be smoothly blended to provide a more continuous effect. We also introduce a new accumulation scheme for importance-driven volume rendering and a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of a light-field display. The possibilities of these techniques are demonstrated by the interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display driven by a cluster of PCs.

6.1 Introduction

Illustrative and NPR techniques typically mimic the style of traditional illustration taking advantage of the illustrators long experience in depicting complex structures or shapes in an easily comprehensible way. In this section a brief overview of the basic illustrative techniques employed in volume rendering will be presented.

First of all, it is important to distinguish the difference between surface-based versus volume-based methods. In volume rendering most of the NPR techniques are just an adaption of the local shading model employed for surfaces adjusted to work within the volume rendering integral. Surfaces can be described not only by a polygonal mesh but also by an isovalue or can be implicitly defined on a volumetric grid. At the end, the resultant image will look like a surface and thus, it can be shaded as such. However, there are also illustrative approaches for volume rendering that do not make use of the notion of a surface.

An important characteristic of the methods detecting surfaces is whether they operate in object or image space. Or equivalently, whether they operate directly over the 3D surface or over its 2D projection. They are normally known as object-space or image-space methods respectively. All the proposed illustrative and NPR methods in this thesis work are object-space, therefore most of the attention will be payed to those approaches, specially when discussing the related work.

Context-preserving volume rendering focuses on solving the problem of which part of the volume must be emphasized, and what kind of rendering style is more suitable to enhance this part. At the same time, it faces the problem of how to avoid the collateral effects of losing important context information, maintaining enough visual cues for de-emphasized or

less important parts. The main approaches generally rely on exploiting a way to decrease the importance of the less relevant information in favour of a region of interest.

The *c*ontext-preserving focal probes model exploit the pq-distance to give a focus shape consistent with a probe defined by a superquadric function. We propose to combine relief shading techniques for enhancing the focus region with a better shape depiction and silhouette darkening effects. For preserving context information, we use an adaptive voxel dependent transparency and perform the clipping of voxels in regions which can create image clutter.

Another important difference between illustrative methods is whether they depend on the relative location of the view point or not. Contours or silhouette are examples of rendering styles that depend on the relative view direction. Curvature, instead, is an intrinsic property of the surface itself. The methods are named view-dependent or view-independent accordingly to the previous characteristic.

It is important to highlight the different computational cost of storing view-dependent information whether a mesh is available or not. For example, it is easy to store view-independent information at mesh-vertexes in order to avoid re-computation. For methods such as ray casting that do not generate explicit geometry, however, most information needs to be re-computed each time. In our work the interest is aimed at methods that do not require an explicit mesh. This avoids the cost of computing this mesh which usually prevents interactive changes of the isovalue. Therefore, all the information that is required for rendering used in an illustrative style must be computable in real time.

The view-dependent characteristics of the light-field displays can be exploited to develop specialized interactive illustrative techniques designed to improve spatial understanding. A set of interactive illustrative tools based on flexible compositing engine presented in chapter 3 are presented to provide different contextual information in different areas of a light-field display. Furthermore, we also present a new accumulation scheme for importance-driven volume rendering.

6.2 Related Work

Traditional illustrative non-photorealistic rendering techniques typically mimic the style of traditional illustrations. They take advantage of the illustrators long experience in depicting complex structures or shapes in an easily comprehensible way. Ebert et al. [Eber 00] combine some physics-based illumination model with non-photorealistic techniques to enhance the perception of structure, shape, orientation, and depth relationships in a volume model. For similar purpose, different stylistic choices have been used in traditional medical illustrations, such as silhouette or contour enhancement, pen-and-ink, stippling, hatching, etc.

In this thesis, we are interested in techniques for emphasizing details of structures which are in focus and for reducing clutter while preserving useful context information in nearby areas.

Context preserving volume rendering focuses on solving the problem of which part of the volume must be emphasized, and what kind of rendering style is more suitable to enhance this part. At the same time, it faces the problem of how to avoid the collateral effects of losing important context information, maintaining enough visual cues for de-emphasized or less important parts. The main approaches generally rely on exploiting a way to decrease the importance of the less relevant information in favour of a region of interest. Hauser et

al. [Haus 01] propose the two-level volume rendering model allowing visualization of volume data with important inner structures together with semi-transparent outer parts as context information. They integrate different rendering modes and compositing types, such as DVR, MIP, or tone shading for the inner part of the volume and contour enhancement for the outer part. Cohen et al. [Cohe 04] discuss in depth how information visualization ideas can be applied to scientific visualization in an effective way. In particular, they combine the "focus and context" concept with distortion effects to improve understanding of large volume datasets. Viola et al. [Viol 04] introduce importance-driven volume rendering, where the emphasis can be automatically put on the part which has been assigned more importance. Ropinski et al. [Ropi 05] propose volumetric lenses to interactively focus regions of interest, rendering the parts of the volume intersecting the lens, which is defined by a convex 3D shape, using a different visual appearance. Bruckner et al. [Bruc 05] suggest to use cut-away views to focus the attention on the intersection region and ghosting views which tends to generally give a better impression of the spatial location of the object in focus.

Context-preserving in volume rendering can be also achieved by exploiting lighting intensity as an input to a function for opacity variation. Bruckner et al. [Bruc 06a] use this idea to reduce the opacity in rather flat regions oriented toward the light source. Parts of the volume receiving less lighting are rendered as semi-transparent silhouettes helping to preserve context information. Krüger et.al [Krug 06] present a method which enables the user to focus on a particular region, using defined parameters such as the size and location of the focus, a weight for the context, and material properties. An approach similar to ours is the one proposed by Zhou et. al [Zhou 04], which emphasizes a region according to the Euclidean distance from the sampled voxel to a sphere center. Tappenbeck et al. [Tapp 06] employ distance-based transfer functions, which allows to hide, emphasize or color structures based on their distance to a relevant reference structure. In our approach, we exploit the pq-distance to give a focus shape consistent with a probe defined by a superquadric function. Furthermore, we propose to combine relief shading techniques for enhancing the focus region with a better shape depiction and silhouette darkening effects. For preserving context information, we use an adaptive voxel dependent transparency and perform the clipping of voxels in regions which can create image clutter.

Silhouette enhancement is typically based on gradient or curvature estimations. Csbfalvi et al. [Csbf 01] visualize object contours based on gradient information as well as on the angle between viewing direction and gradient vector using depth-shaded maximum intensity projection. Kindlmann et al. [Kind 03] employ curvature along the view direction to achieve illustrative effects, such as ridge and valley enhancement. It is also common in medical illustrations to depict shape or surface details in a way that is inconsistent with physically-realizable lighting model. Inspired by the techniques used in cartography, Rusinkiewicz et al. [Rusi 06] develop a non-photorealistic rendering strategy based on multi-scale local toon shading. Based on it, we develop a non-photorealistic shading model applicable to volume rendering.

Our flexible renderer employs both photorealistic and illustrative techniques. Some techniques mentioned in this section, like the accumulation and pre-integration schemes or the opacity model were already introduced in chapter 3 as part of the GPGPU ray-caster, so we refer the reader to subsection for a full description on those engine characteristics.

The accumulation scheme is similar to MIDA [Bruc 09], even though in our case we accumulate importance rather than intensity. The usage of an importance transfer function has also been advocated by [Viol 05]. In order to deal with high frequency transfer functions, we employ a pre-integration approach for all attributes, rather than only for colors as is usually done, and use a numerically stable approach which only needs little memory requirements. Similarly to [Krau 08], we use extinction-weighted colors rather than opacity-weighted ones and reconstruct opacity on the GPU. Our view-dependent probe and clip planes bear a number of similarities with previous context-preserving techniques [Bruc 06b], but extend them with view-dependent effects tuned for 3D displays.

6.3 The Context-Preserving Focal Probe Model

In our focal probe model we provide the possibility to interactively define a region of interest composed of a part in focus and its context. The focus and context information are separated by the assignment of different rendering styles, which allow to easily distinguish between the different parts. Nevertheless, the rendering styles can be smoothly blended to provide a more continuous effect.

We define the region of interest using a distance-based function which basically decrease the opacity of the samples according to its distance to the center. The focus rendering style is inspired by relief shading techniques to perform enhancement of shape details combined with a silhouette darkening effect. Meanwhile, the context style performs silhouette detection combined with a view-dependent transparency modulation effect. Finally, we clip all samples before the probe and saturate all samples behind it, in order to avoid cluttering results for rays intersecting the probe.

The presented model allows the user to translate, rotate or scale the probe size according to the requirements for an adequate visualization of the datasets.

6.3.1 Background

We will briefly describe the ray casting process along one viewing ray for simplicity's sake. We assume a volumetric scalar field as $f(P) \in R \quad (P \in R^3)$, then denote the viewing ray direction by V and its normalized vector by $\hat{V}$. Let P_i be the i-th sample location along a viewing ray V, f_i be the data value at P_i, g_i be the gradient at P_i and $\hat{g}_i$ be its normalized vector. We denote the gradient magnitude by $\|g_i\|$, and the normalized gradient magnitude by $\|\hat{g}_i\|$, which is $\|g_i\|$ divided by the maximum magnitude $\|g_{max}\|$. Since the gradient has noise, we filter it using an interpolation function as below

$$w_i = \text{smoothstep}(\|\hat{g}_i\|, g_l, g_h) \tag{6.1}$$

where smoothstep is a cubic function defined by

$$\text{smoothstep}(t, a, b) = \begin{cases} 0 & \text{if } 0 \leq t < a \\ (\frac{t-a}{b-a})^2(-2(\frac{t-a}{b-a}) + 3) & \text{if } a \leq t \leq b \\ 1 & \text{if } b < t \leq 1 \end{cases} \tag{6.2}$$

Conventional volume rendering DVR uses the front-to-back alpha blending which employs a physically motivated absorption-plus-emission optical model, computing the accumulated opacity α_i^* and color $c_i^* = (r_i^*, g_i^*, b_i^*)$ at step i with regular increment $\Delta s = \|P_i - P_{i-1}\|$ along the viewing ray V as follows

$$\begin{cases} c_i^* & = c_{i-1}^* + (1 - \alpha_{i-1}^*)\alpha_i c_i \\ \alpha_i^* & = \alpha_{i-1}^* + (1 - \alpha_{i-1}^*)\alpha_i \end{cases} \tag{6.3}$$

where α_i and $c_i = (r_i, g_i, b_i)$ are the opacity and color respectively at P_i, both derived from the transfer function.

The light properties make it possible to enhance objects for a variety of effects. Before accumulating, generally we use the traditional Phong model to perform a shading effect by modifying the color c_i to be the shaded color $c_{i,\text{shaded}}$ as below

$$c_{i,\text{shaded}} = \lambda_i c_i \tag{6.4}$$

where

$$\lambda_i = k_a + \sum_{\text{lights}} k_d(\hat{L} \cdot \hat{g}_i) + k_s(\left|\hat{H} \cdot \hat{g}_i\right|)^{k_e} \tag{6.5}$$

k_a, k_d and k_s are the ambient, diffuse and specular lighting coefficients. $\hat{L}$ is the normalized light vector, $\hat{H}$ is the normalized half-way vector, and k_e is the shininess constant.

After shading, we denote the opacity at P_i by $\alpha_{i,\text{shaded}}$, which basically follows the same variation as color in Eq. 6.4. Then we substitute c_i with $c_{i,\text{shaded}}$ and α_i with $\alpha_{i,\text{shaded}}$ into Eq. 6.3 performing the normal accumulation process.

6.3.2 Probe Shapes

Medical datasets can capture quite heterogeneous parts of the human body, varying in a wide range of physical dimensions. For a satisfactory exploration of diverse volume datasets we propose to use an exploration tool based on the superquadrics family functions. These functions provide us a flexible and relative simple solution for modelling rounded or sharp corners for a variety of basis geometric shapes (See Fig. 6.1).

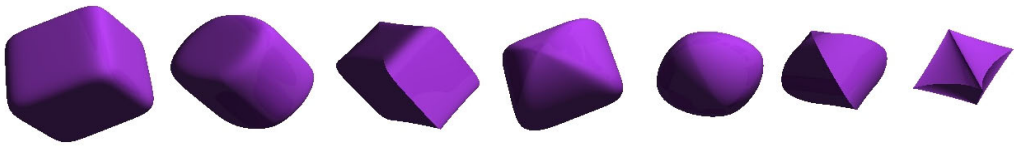


Figure 6.1: **Superquadrics shapes.** Examples of shapes generated from the superquadrics family functions

We use a geometric distance based on the pq-norm to define the superquadric. Since it provides us with a distance function, we can use it to define a probe center. The pq-norm for a sample P_i with x_i, y_i, z_i coordinates is defined by

$$\|P_i\|^{pq} = \|(\|(x_i, y_i)\|^p, z_i)\|^q \tag{6.6}$$

where

$$\|(x_i, y_i)\|^p = (|x_i|^p + |y_i|^p)^{\frac{1}{p}} \tag{6.7}$$

is p-norm, which is the generalization of the Euclidean metric ($p = 2$). As shown in Fig. 6.2, various consistent focus shapes can be obtained by using different p and q values.

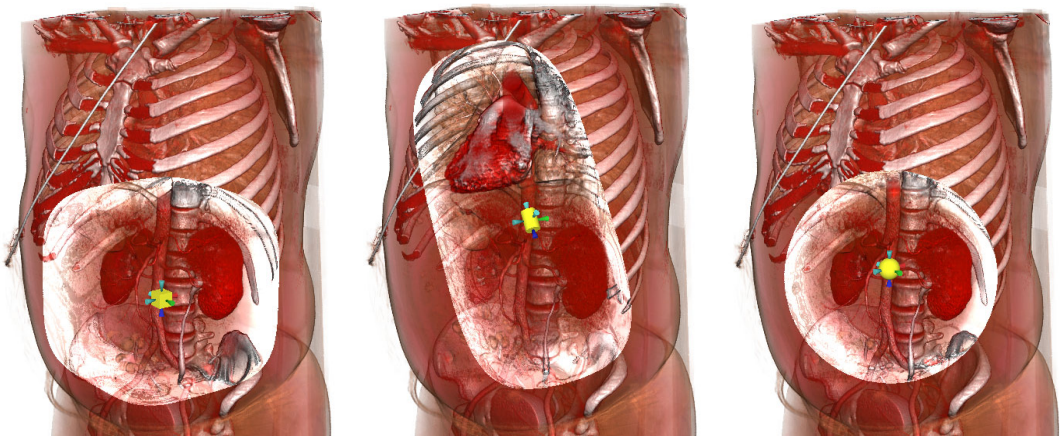


Figure 6.2: **Examples of probe shapes.** We have tested various probe shapes, including rounded cubic ($p = 4$, $q = 4$), cylindrical ($p = 2$, $q = 4$) and spherical ($p = 2$, $q = 2$).

6.3.3 Distance-based Merging of Rendering Styles

In our focal probe model we propose to employ different rendering styles corresponding with the focus and the context region. One idea could be to use a particular threshold to differentiate between the different zones within the probe, but this decision can lead to sharp transitions with a continuity loss in the resulting rendering. Instead, we propose to merge both rendering styles using a distance-based function d_i having a plateau for the focus region determined by $\rho \quad (0 \leq \rho \leq 1)$, which can be interactively adjusted by the end user of the system.

$$d_i = \begin{cases} 1 & \text{if } 0 \leq \|P_i\|^{pq} \leq \rho \\ 1 - g_\beta(\frac{\|P_i\|^{pq} - \rho}{1-\rho}) & \text{if } \rho < \|P_i\|^{pq} \leq 1 \end{cases} \tag{6.8}$$

where

$$g_\beta(t) = \frac{t}{e^\beta(1-t) + t} \tag{6.9}$$

is the schlick rational function [Schl 94]. We use the β parameter, normally being $\beta < 0$, for having a smoother or sharper decreasing value of the context region.

We use different rendering styles for the focus and context regions which will be introduced later. Let ($c_{i,\text{focus}}$,$\alpha_{i,\text{focus}}$) and ($c_{i,\text{context}}$,$\alpha_{i,\text{context}}$) be the color and opacity at P_i defined by these

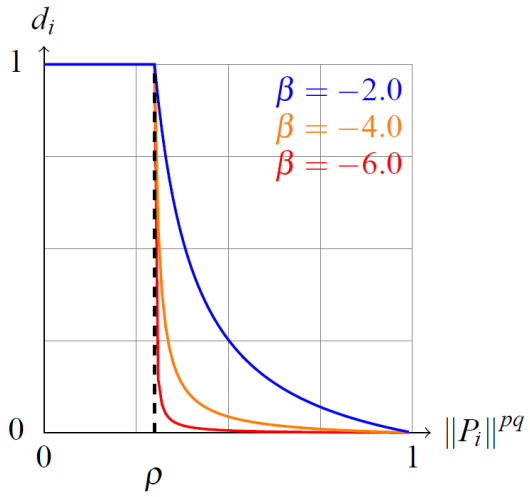


Figure 6.3: **Distance based function.** The pq-distance function employed in the focal probe model can be varied by changing the value of the β parameter.

rendering styles, our proposed blending strategy is performed by using d_i as focus fraction.

$$\begin{cases} c_{i,\text{blend}} = d_i c_{i,\text{focus}} + (1 - d_i) c_{i,\text{context}} \\ \alpha_{i,\text{blend}} = d_i \alpha_{i,\text{focus}} + (1 - d_i) \alpha_{i,\text{context}} \end{cases} \tag{6.10}$$

After blending (See Eq. 6.10), c_i is substituted by $c_{i,\text{blend}}$ and α_i by $\alpha_{i,\text{blend}}$ into Eq. 6.3 performing the normal accumulation in the rendering pipeline.

6.3.4 Focus Model

Next, we propose a rendering style which performs better shape depiction for the focus region based on traditional DVR and non-photorealistic techniques (See Fig. 6.4). In particular, we perform a silhouette darkening effect based on the following detector

$$s_i = w_i * \text{smoothstep}(1 - |\hat{g}_i \cdot \hat{V}|, s_l, s_h) \tag{6.11}$$

where s_l and s_h control the silhouette sharpness. The focus color $c_{i,\text{focus}}$ and opacity $\alpha_{i,\text{focus}}$ at P_i are defined by

$$\begin{cases} c_{i,\text{focus}} = (1 - s_i) r_i c_i \\ \alpha_{i,\text{focus}} = r_i \alpha_i \end{cases} \tag{6.12}$$

where r_i is a shading factor implementing an illustrative effect explained next and defined by Eq. 6.14. We combine the previous color darkening processing with an illustrative relief shading effect. Principles in relief shading [Rusi 06] advise to omit shadows and specular effects, exaggerate the height of ridges and valleys in the object shape and use a particular lighting effect which appears to originate as from the top of the image. Furthermore, these principles suggest to locally adjust light direction in order to homogeneously light the interesting part of the volume. Finally, to support multi-scale toon shading effects, it is recommended to blend between different smoothed versions of the volume.

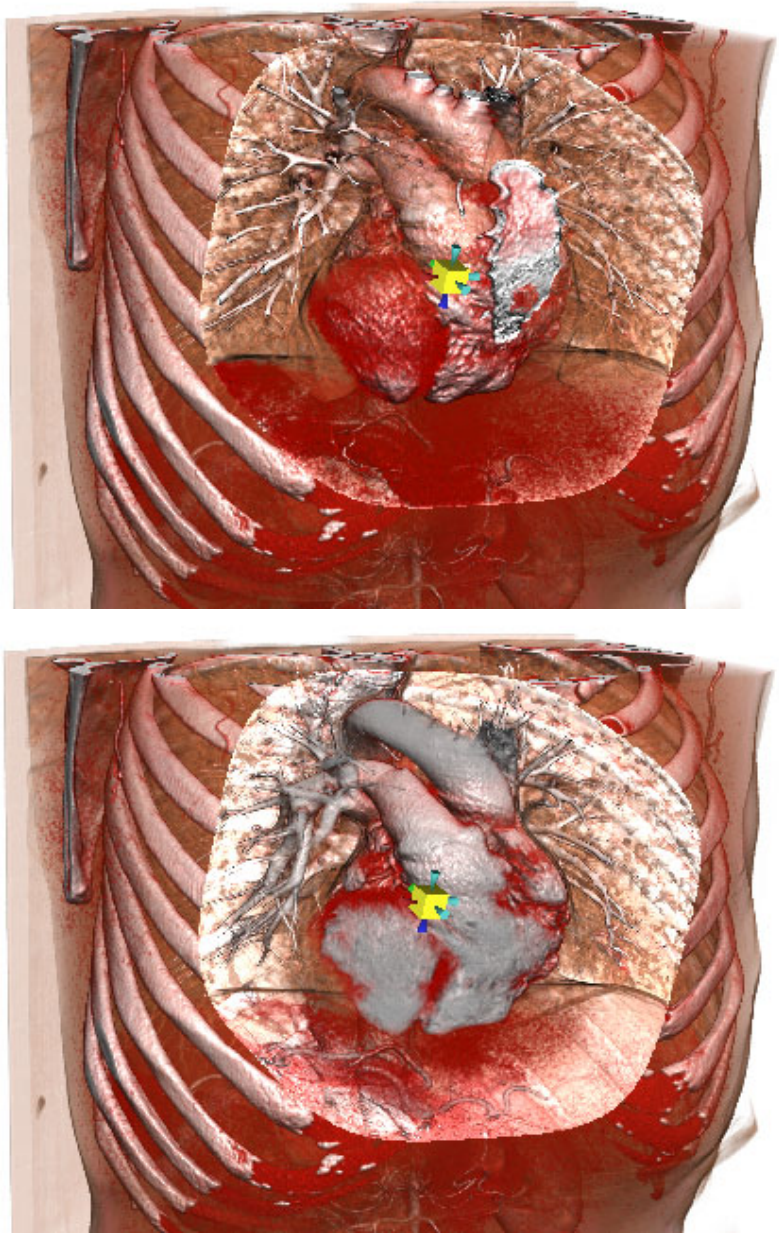


Figure 6.4: **Comparison between a probe with and without focus.** On the image on top we provide an example of a normal probe without focus. Notice how the sternum bone hides part of the heart. Besides, flesh and lung tissues create a cluttering effect as they cover part of neighbour structures. On the image below, we show the same situation using a context-preserving focal probe. In this case, heart is right focused, no cluttering is created around it. Furthermore, context information is preserved and previously hidden vessel structures are now better identified.

Applying inside our system Rusinkiewicz et al.'s techniques based on relief shading, requires a special preprocessing to produce smoothed versions of the volume dataset for multi-scale based effects. In our model we will apply similar basis concepts in order for detail enhancement without necessity of preprocessing. We compute on the fly a local version $\hat{n_i}$ of the smoothed normal, accessing samples located at positions $x_{i\pm1}$, $y_{i\pm1}$ and $z_{i\pm1}$ for a sample at P_i with x_i, y_i, z_i coordinates. Next, we make a local light adjustment at each sample, computing the new light position L_i^* at P_i based on the global light position L, and the smoothed

normal $\hat{n}_i$ at P_i. For this purpose, we create a light which is perpendicular to the smoothed normal $\hat{n}_i$, exaggerating the shading effect for ridges and valleys in our volume (See Eq. 6.13).

$$L_i^* = L - \hat{n}_i(\hat{n}_i \cdot L) \tag{6.13}$$

Furthermore, as shown in Eq. 6.14, we use the original shading $\hat{L} \cdot \hat{g}_i$ as a basis for the lighting of the diffuse component, thus to avoid the appearance of excessively dark zones in the final rendering. Next, we sum the contribution of the P_i local light $\hat{L_i^*} \cdot \hat{g}_i$. Finally, we have altered the local lighting to produce a toon shading effect, just clamping $\hat{L_i^*} \cdot \hat{g}_i$ between 0 and 1 and multiplying the result by the toon shading factor a.

$$r_i = k_a + k_d(\hat{L} \cdot \hat{g}_i + a * \text{clamp}_{[0..1]}(\hat{L_i^*} \cdot \hat{g}_i)) \tag{6.14}$$

6.3.5 Context Model

The rendering style proposed for the context region is based on view-dependent transparency and silhouette detection. The context color $c_{i,\text{context}}$ and opacity $\alpha_{i,\text{context}}$ at P_i are defined by

$$\begin{cases} c_{i,\text{context}} = \lambda_i c_i \\ \alpha_{i,\text{context}} = h(\theta) s_i \lambda_i \alpha_i \end{cases} \tag{6.15}$$

where

$$h(\theta) = \begin{cases} 0 & \text{if } 0 \leq \theta \leq \varphi \\ 1 - \frac{cos\theta - cos\varphi}{cos\phi - cos\varphi} & \text{if } \varphi < \theta \leq \phi \end{cases} \tag{6.16}$$

Shown by the left most image in Fig. 6.5, θ is the angle formed by the O vector connecting the eye position with the probe center and the viewing direction V, ϕ is the angle defining the maximum angle for rays intersecting the probe, and finally φ the angle defining rays passing through the focus region. The shading factor λ_i has been previously defined by Eq. 6.5. The $\alpha_{i,\text{context}}$ parameter introduces a view-dependent effect, giving more opacity to voxels in the outer part of the probe and increasing the transparency as samples are closer to the probe center.

The use of volumetric probes entails the problem of possible occlusion due to the voxels situated between the eye position and the probe. This is also valid for voxels in the context region, which can impede the correct visualization of our interest region. We illustrate this problem in the right image of Fig. 6.5. Therefore, a reasonable strategy seems to cancel the accumulated color just before the first intersection of the viewing ray with the probe, corresponding to the red cone in the right of Fig. 6.5. Samples in the context region behind the probe focus are ignored, and following samples in the volume are blended with the background contributing less to the final accumulated color along the ray.

6.3.6 Visualization Results

In this section, we report on the results obtained when performing exploration of medical datasets by using our focal probe model. We have tested the proposed technique with a variety of volumetric medical datasets. We discuss the results obtained with the inspection

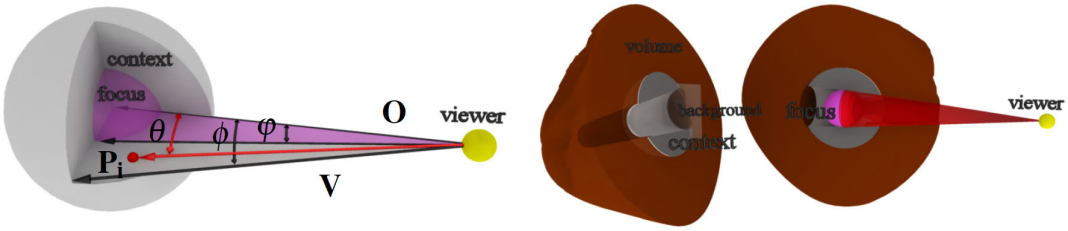


Figure 6.5: **Context definition for focal probes** In the left image, for a sample P_i in the context region, we define the θ angle which decides how much the context will be displayed. In the right, voxel clipping is performed in order to produce a non cluttered render image.

of a $512 \times 512 \times 1559$ whole body contrast and a $512 \times 512 \times 743$ thorax study, both CT [1] with $16bit/sample$.

Regarding the focal probe model, we employ the following parameter configuration. We use $\beta = -2.0$ for having a smooth transition between the focus and context information (See Fig.6.3). To separate the focus and context regions, we adjust the ρ parameter in the interval $[0.4, 0.7]$. For gradient filtering we compute the w_i interpolation function with $g_l = 0.015$ and $g_h = 0.95$. Related with the non-photorealistic shading, we control the silhouette detector s_i using $s_l = 0.7$ and $s_h = 0.95$ for a fine silhouette enhancement without creating cluttering owing to low gradient values. We define the toon shading factor $a = 3.0$ to enhance lighting differences between ridges and valleys in the focus region.

As described in Fig. 6.3 the behaviour of our focal probe model establishes a center of attention, fitting in the region around of the probe center. Transparency is modulated by the d_i distance based function and its effect results in more transparency for voxels in the context region. In Fig. 6.6 we can appreciate the effect of an interactive incursion of a focal probe within the whole human body CT. At the beginning, the probe acts just as a clipping sphere and as soon as the center of the probe is inside the volume, the focus effect becomes visible, as well as the context-preserving effect showing contour shapes. When exploring parts of the volume which are too deep, voxel clipping becomes active.

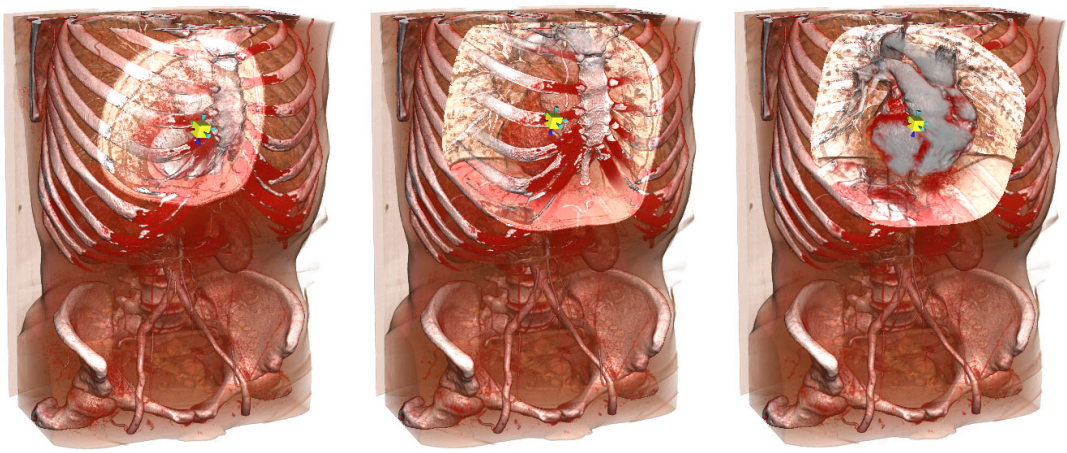


Figure 6.6: **Incursion sequence of a focal probe model inside a medical dataset.** From left to right we increase the focal probe penetration stopping when the heart is right focused.

[1] Source: Geneva University Hospital, Radiology Department

Using the silhouette darkening effect we can depict shape borders using a darker color and enhancing the shape over its background. In Fig. 6.7 we show a series of snapshots of our volume renderer varying the parameter s_h of Eq. 6.11, which indicates the threshold value above which a sample is considered a border shape.

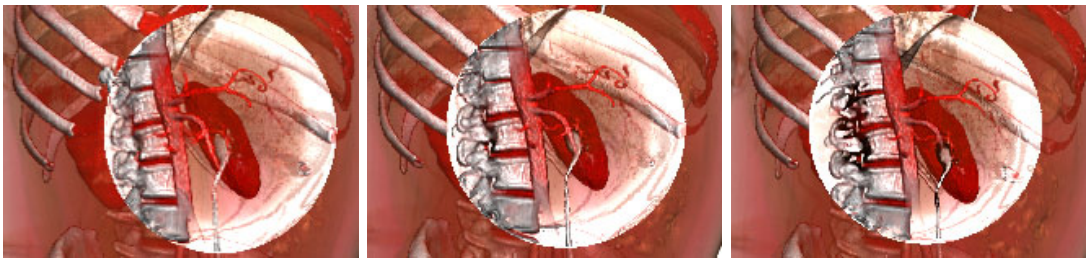


Figure 6.7: **Silhouette darkening effect.** From left to right, no silhouette darkening $s_h = 1$, silhouette darkening with $s_h = 0.8$ and $s_h = 0.4$.

The relief shading technique, described in Sec. 6.3, allows the user to appreciate additional details within the focus region, exaggerating the lighting effect by using a perpendicular light for each voxel (See Fig. 6.8). In our approach we rely on a local version of the smoothed normals, and for this reason, the enhanced details are limited to local details, normally corresponding to high frequencies in the volume.

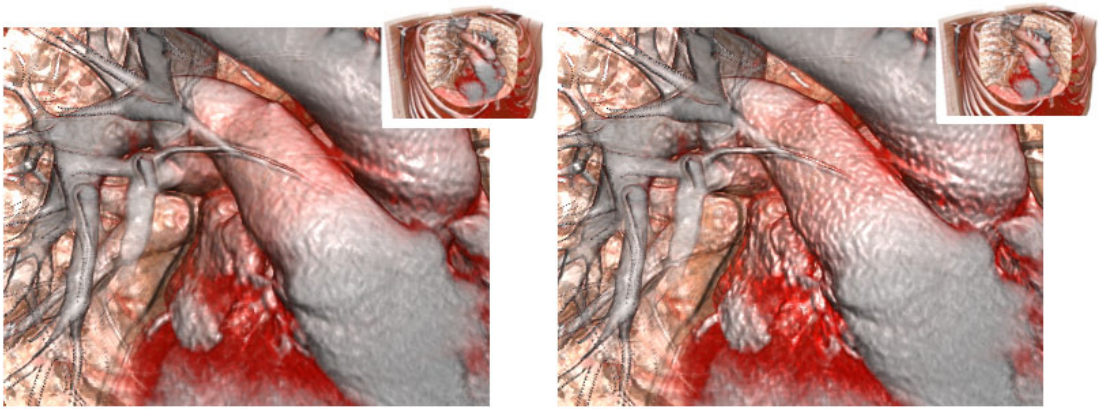


Figure 6.8: **The relief shading effect.** On the left we show a probe focus without relief shading whereas on the right we show the same viewpoint with relief shading for the focus probe. Notice how the high-frequency details are highlighted.

6.4 View-dependent Illustrative Techniques for the Light-field Display

The view-dependent characteristics of the display can be exploited to develop specialized interactive illustrative techniques designed to improve spatial understanding. With such techniques, simple head motions can reveal new aspects of the inspected data. In the following, we illustrate three examples of techniques that can be implemented in our framework by defining appropriate enhancers associated to tools manipulated with a 3D cursor. These techniques can be also freely mixed and matched within a single interactive application.

6.4.1 Clip-plane with View-dependent Context

Frequently, the use of clipping planes is the only way to uncover otherwise hidden details of a volumetric dataset or to visualize non-orthogonal slices of the objects. Clip planes, however, provide most of their information when viewers are facing them, but offer little insight in most other situations. In particular, when view direction is parallel to the plane, no information is provided to the viewer (besides the removal of a portion of the object). Thus, we propose a formulation that supports traditional cut away visualization, when our view direction is orthogonal to the clip plane, while offering more helpful contextual information in other situations (see Fig. 6.9). We compute the distance from the plane $\delta_i = \mathbf{n} * x_i + p$. If the distance is positive, we simple modify the opacity of samples by multiplying it by a view-dependent correction factor $\mu_i = \text{smoothstep}(1 - \max(0, -\mathbf{n} \cdot \mathbf{v}), f_\text{l}, f_\text{h})$, where the smoothstep function is a cubic polynomial that smoothly transitions from zero to one as the view-dependent factor varies between f_l and f_h. If, otherwise, $-\delta_\text{thickness} < \delta_i < 0$, we smoothly vary the opacity of the plane and shading parameters from the original ones at $\delta_i = -\delta_\text{thickness}$ to full opacity and ambient plus emission shading at $\delta_i = 0$, emphasizing the tissue sliced by the plane. The other samples on the back of the plane are left unchanged.

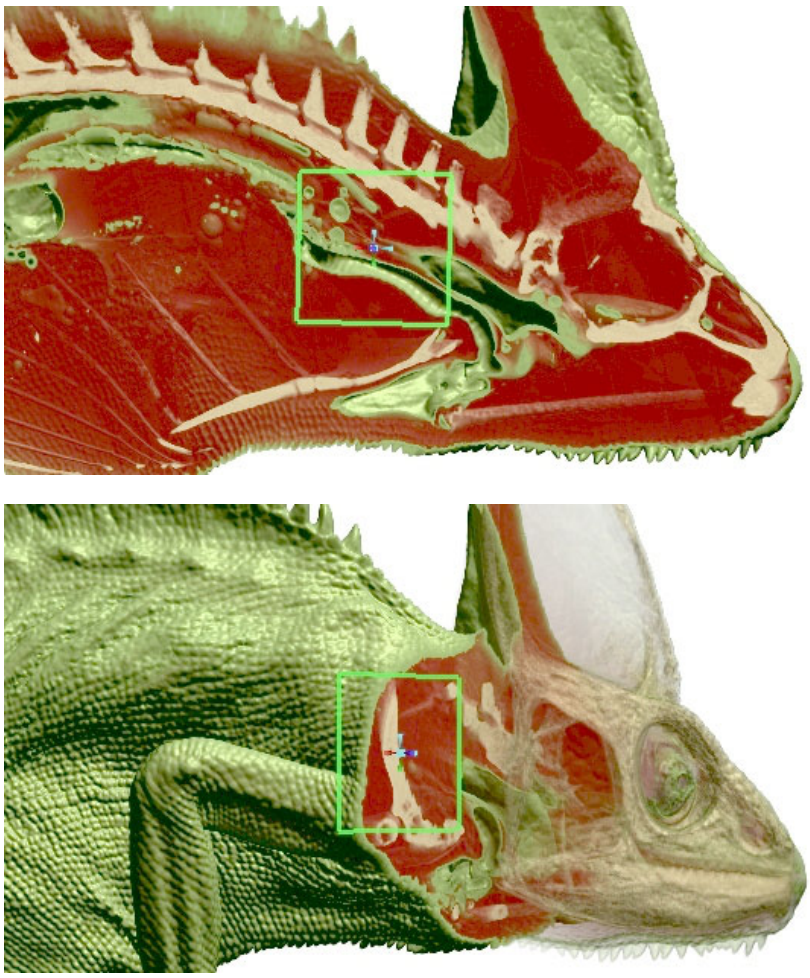


Figure 6.9: **Clip plane with view-dependent context.** As the view becomes less and less orthogonal to the plane, more and more contextual information appears.

Figure 6.10: **Screenshot of the view-dependent clip plane effect.** Notice that depending on the orientation of the cut more contextual information becomes available.

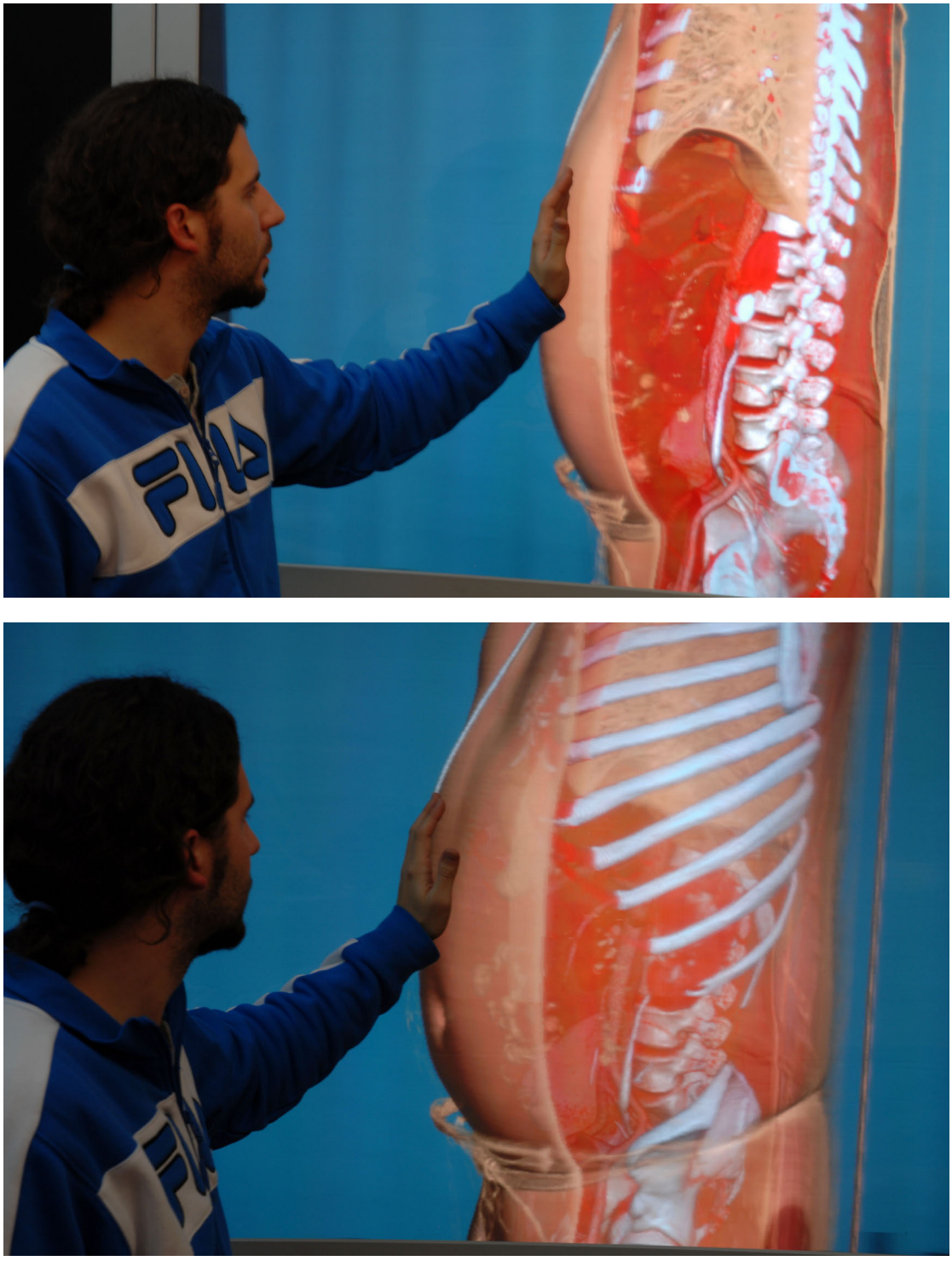


Figure 6.11: **View-dependent clip plane effect with medical data on the 3D display.** The cut in the sagital plane, on the top image, allows to see the organization of the inner parts of the body, whereas when the view becomes less orthogonal to the plane, like in the image below, more information about the surface of the body appears.

6.4.2 Context-preserving Probe

Our context-preserving probe provides a means of interactively inspecting the interior of a volumetric dataset in a feature-driven way which retains context information (see Fig. 6.12). Our spherical probe is positioned in space, and has an associated enhancer, which modifies sample opacity and color by multiplying them with two correction factors. Both correction factors have a dependency on two weighting functions, ω_{d}, which is a quadratic polynomial equal to one on the plane passing through the probe center and oriented towards the viewer and zero at a distance from this plane equal to the probe radius, and ω_{a}, computed similarly to the view-dependent factor of our clip plane and is equal to one when the probe axis is aligned with the view direction and quickly goes to zero for other directions. The opacity correction factor $\mu_i^{(\alpha)}$ and the intensity correction factor $\mu_i^{(i)}$ are computed by:

$$\mu_i^{(\alpha)} = \begin{cases} \mathrm{lerp}(0, \eta_i, \omega_{\mathrm{d}}) & d < d_{max} \\ \mathrm{lerp}(\eta_i, 1, 1-\omega_{\mathrm{d}}) & d \geq d_{max} \end{cases} \tag{6.17}$$

$$\mu_i^{(i)} = \begin{cases} 1-\eta_i & d < d_{max} \\ \mathrm{lerp}(1-\eta_i, 1, 1-\omega_{\mathrm{d}}) & d \geq d_{max} \end{cases} \tag{6.18}$$

where $\eta_i = \omega_{\mathrm{a}}\mathrm{smoothstep}(\iota_i \frac{\|\nabla s_i\|}{\nabla_{\max}}, g_l, g_h)(1 - \frac{(\nabla s_i \cdot \mathbf{v})}{\|\nabla s_i\|})^4$.

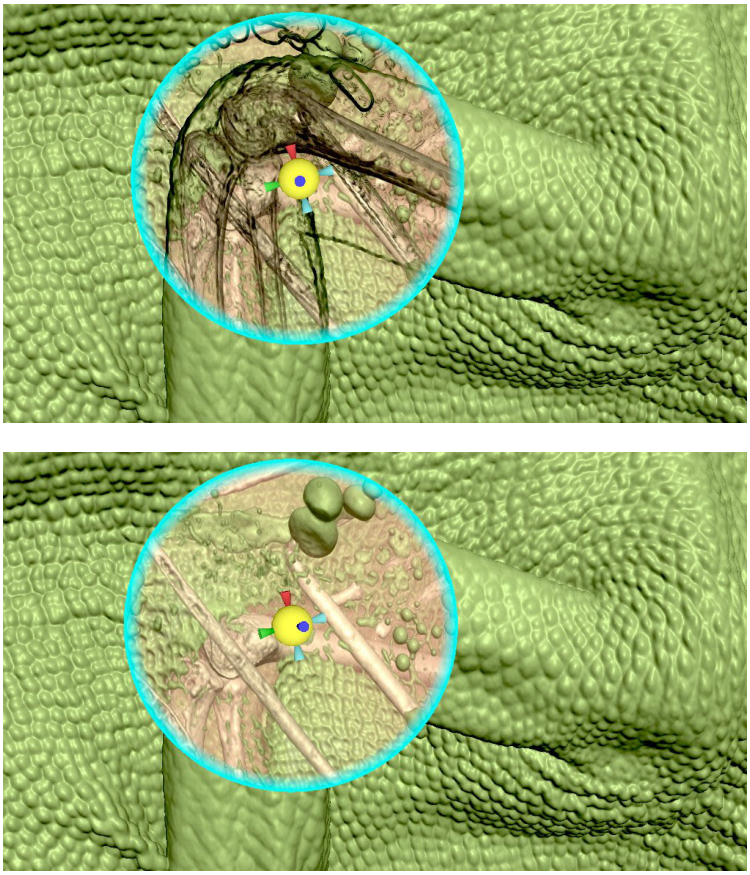


Figure 6.12: **The context-preserving probe.** Material becomes less and less opaque with increasing tool penetration. Contextual edges are visible only if looking straight at the tool (see the image on top), and clutter due to contextual information can be removed by a simple head motion (see the image below).

By scaling opacity by a function of gradient norm, we emphasize edges within contextual information. Since the gradient is a bad predictor for the surface normal orientation in nearly homogeneous regions due to the increased influence of noise, we filter its magnitude through the smoothstep function, which smoothly transitions from zero to one as the (rescaled) gradient varies between g_l and g_h. Scaling gradient norm by the importance ι_i allows us to have edges within less important tissues fade out quickly. η_i is proportional to the alignment of both the tool axis and the plane defined by the local gradient with the view direction. Rather flat surfaces that are oriented towards the viewer will thus fade out in the context region. At the same time, edges will only be visible if looking straight at the tool, and clutter due to contextual information can be removed by simple head motions. The distance-based modulation forces everything to transparent when entering the probe, and gradually increases the opacity modulation factor to one on the probe back. Thus, the material within the probe becomes less and less opaque with increasing tool penetration. The effect of the color correction factor is to perform silhouette darkening in the context region, while gradually blending to the original material in the back of the probe.

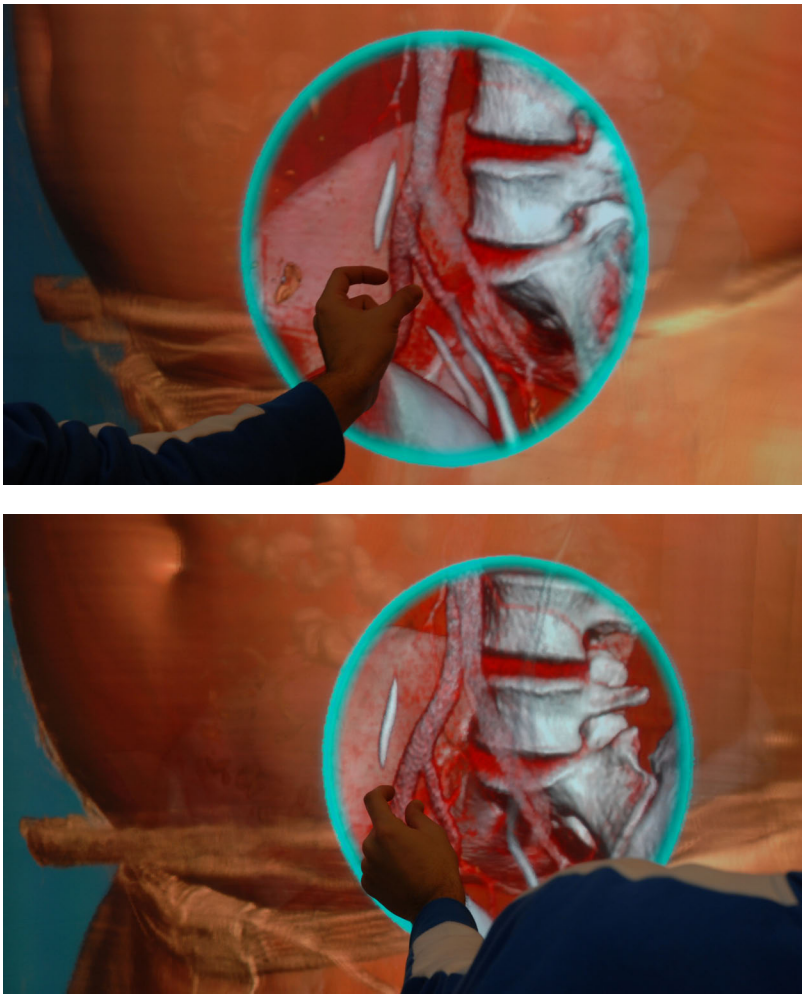


Figure 6.13: **The context-preserving focal probe inspecting medical data on a 3D display.** Sequences in a real test case using the light-field visualization. Notice the appearance of the edges of a thinner vein as context information (on the image below).

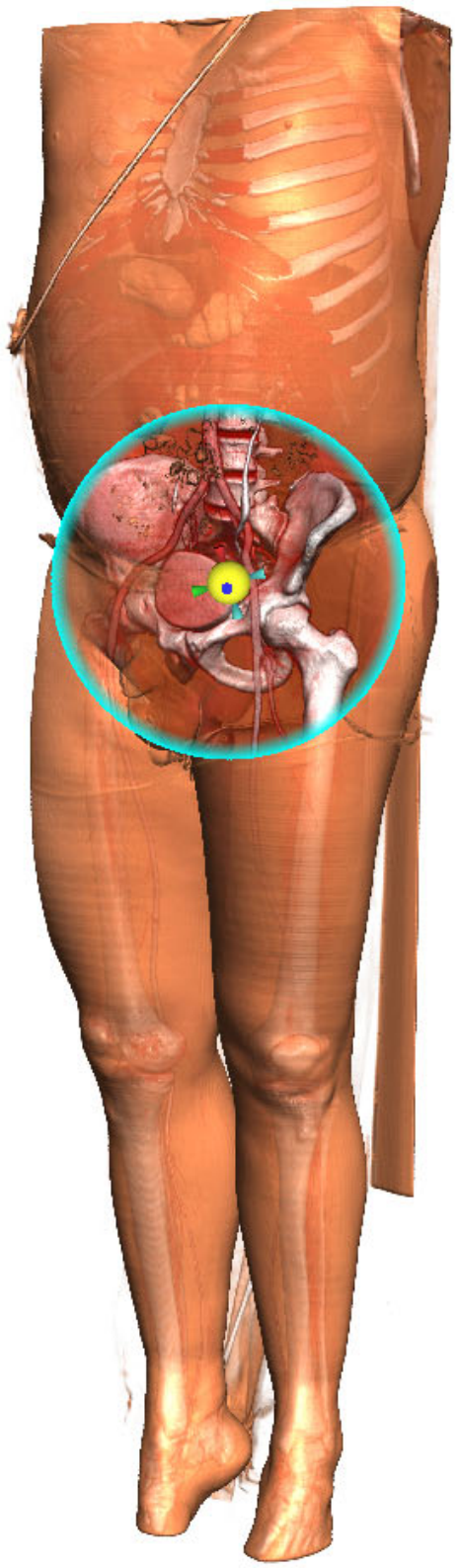


Figure 6.14: **Single frame rendering of a view-dependent probe.** The probe allows to inspect the inner structures of the body and even enhance the silhouette of some finer structures positioned in the border part of the probe.

6.4.3 Band Picker

Our band picker provides a means to enhance the importance of a currently selected band in the transfer function. We assume, as usual, that a band is parametrized by four ordered values, which define a trapezoid function. The values delimiting the current band can be determined by accessing an existing transfer function, or by using a region growing approach similar to [Huan 03]. In addition, the area within which the modification is applied is made view- and tool-dependent by weighting the importance and opacity modification by factors proportional to the alignment between the view direction and the tool main axis (see Fig. 6.15). In our band picker, we set the importance of the samples within the selected band to one, and the importance of other samples to zero. At the same time, we modulate the opacity of the other bands as a function of the view angle. When looking straight at the probe, only the selected band is visible, while when looking at an angle other bands provide contextual information.

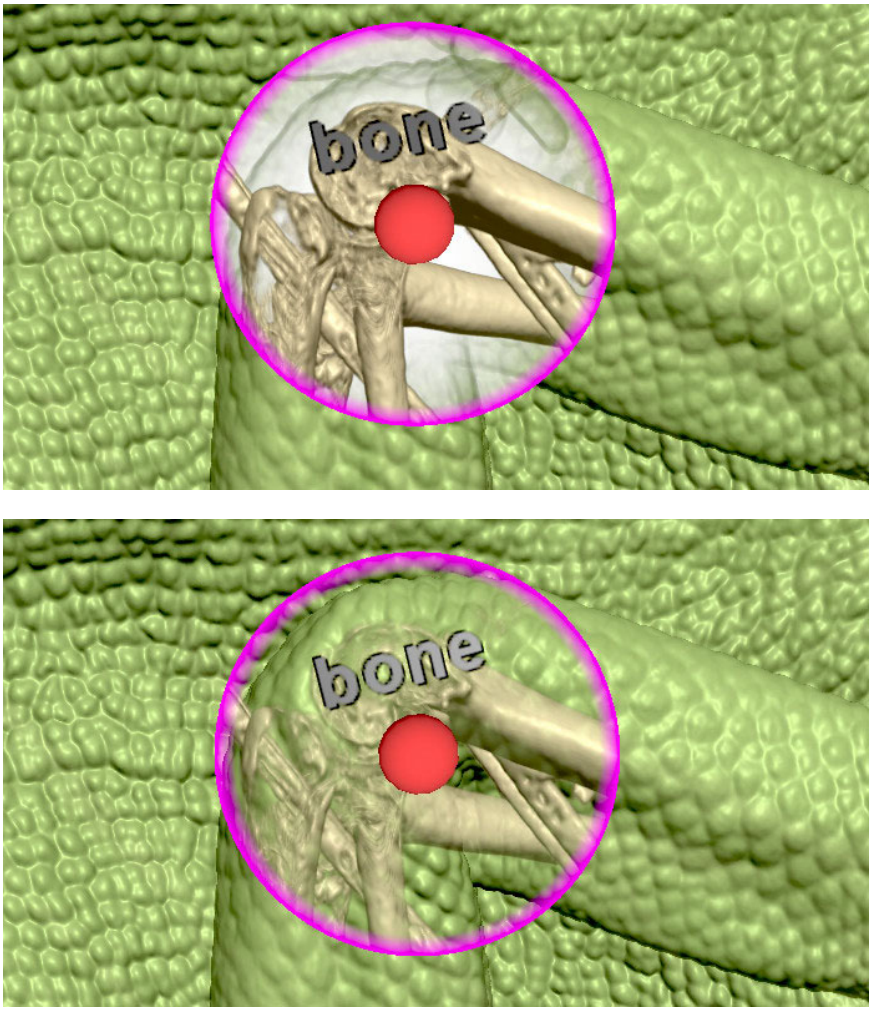


Figure 6.15: **The view-dependent band picker effect.** The tool selectively highlights the selected material. When looking straight at the probe (image on top), only the selected band is visible, while when looking at an angle (on the image below) other bands provide contextual information using MImDA.

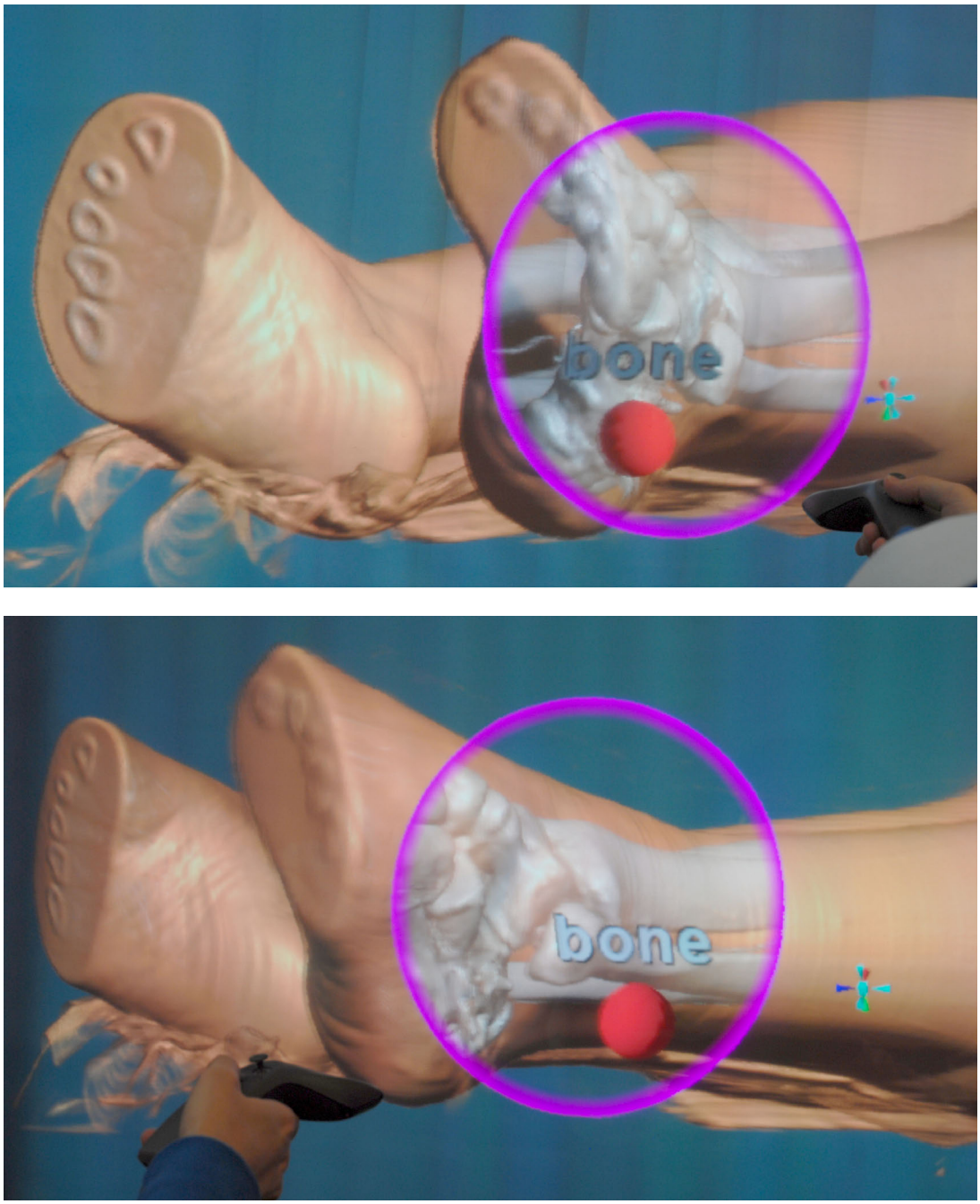


Figure 6.16: **Band picker working with medical data on a 3D display.** The band picker provides view-dependent transfer function information which can be naturally exploited when using a light-field display.

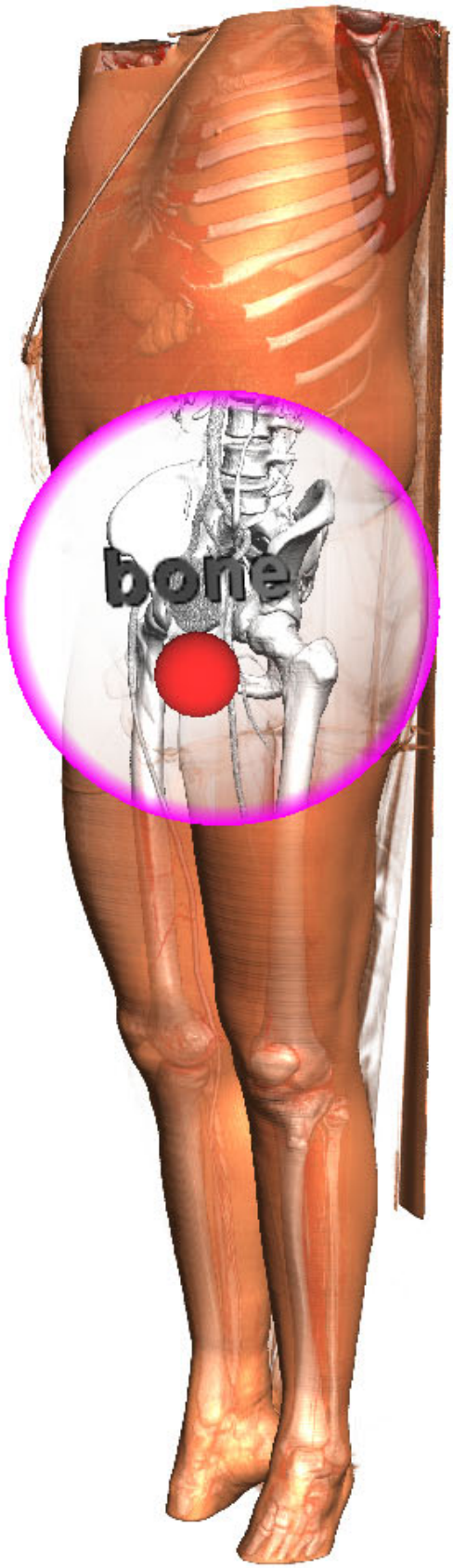


Figure 6.17: **Effect of employing the view-dependent band picker.** During a full-body CT inspection the band picker allows to enhance materials which normally would be hidden.

6.5 Conclusion

In this chapter, we reported on a set of illustrative and non-photorealistic rendering (NPR) techniques that complemented the work presented in the chapters 2, 3 and 5. In the field of context-preserving volume rendering we proposed an illustrative technique to focus the attention of the users in a region of interest while preserving the information in context.

We have described an alternative focal probe model for interactive exploration of medical volumetric datasets. The key feature of our approach relied on the enhancement of natural filtering mechanisms of humans to separate and abstract information. Our model was based on the development of different rendering styles for the focus and context information. The superquadrics family functions were used to get consistent focus shapes. Furthermore, we proposed an adaptive voxel shading, based on illustrative NPR techniques, in order to enhance shape depiction in the focus region. In addition, we used silhouette detection to convey better structure cues in the focus region and to preserve the context information in the outer part. Finally, we proposed a strategy to smoothly blend both rendering styles.

A particular clipping strategy was performed for voxels in front of the probe and behind it, in order to solve the occlusion problem and to avoid the creation of a noisy background.

In the area of view-dependent effects we introduced a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of a light-field display. This set of view-dependent tools needed to be integrated with our GPGPU-based ray casting engine and its new accumulation scheme for importance-driven volume rendering (MImDA), both explained in detail in chapter 3. The possibilities of these techniques were demonstrated by the interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display driven by a cluster of PCs.

6.6 Bibliographical Notes

Most of the contents of this chapter regarding the user-driven exploration of regions of interest in volumetric datasets were extracted from the paper [Luo 09], where we presented our *context-preserving focal probes* model. For a complete description of the prototype system integrating all the presented illustrative techniques, we refer the reader to our article [Igle 10], where we presented, among other things, a new accumulation scheme for importance-driven volume rendering and a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of a light-field display.

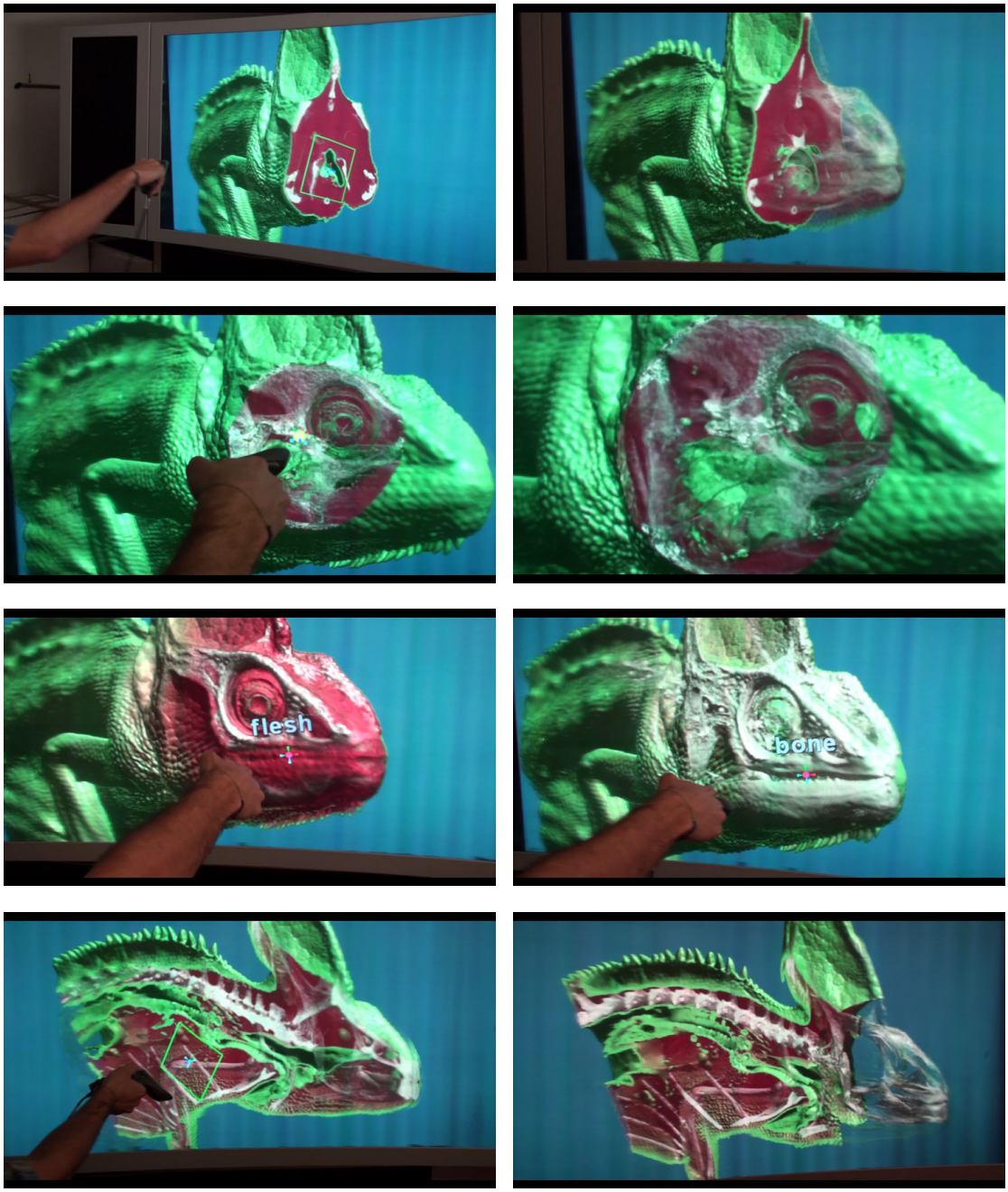


Figure 6.18: **Interactive volume inspection on a light-field display.** These images, show different screenshots during different interactive exploration sessions.

Chapter 7

Summary and Conclusions

This thesis has introduced scalable methods for rendering volumes of potentially unlimited size on modern GPU architectures, as well as methods to improve their understanding through illustrative techniques and image delivery on light field displays. This final chapter summarizes the results obtained.

7.1 Achievements

Visual analysis by means of interactive visualization and inspection of spatial information and data embedded in three-dimensions is a particularly efficient approach to gain insight into the structure and implications of large datasets. In this thesis, we have tackled the problem of improving visualization of rectilinear scalar volumes, i.e., scalar functions sampled on a 3D grid, which arise in many engineering and scientific areas. The research work has led to the following advances with respect to the state-of-the-art:

- The introduction of a novel single-pass ray casting framework for interactive out-of-core rendering of massive volumetric models. The method is GPU-accelerated and has demonstrated the capability of managing multi-gigavoxel datasets [Gobb 08]. The key insight of the method is to use specialized multi-resolution structures, separating visibility-aware level-of-detail selection from the actual rendering using co-operative algorithms.

- The generalization of the previous ray casting framework using the possibilities offered by modern GPGPU architectures [Agus 08c]. The method supports a more flexible ray traversal (e.g., changes in the direction of the ray propagation or different accumulation strategies) and proposes an improved solution for incorporating visibility feedback.

- The development of a volume representation technique [Agus 10b] suitable for cases where the volumes represent physical objects with well defined boundaries separating different materials, giving rise to models with quasi-impulsive gradient fields. In this representation, we replace blocks of N^3 voxels by one single voxel that is split by a feature plane into two regions with constant values. We also show how to convert a standard mono-resolution representation into an out-of-core multi-resolution structure, both for labeled and continuous scalar volumes.

- The introduction of an adaptive technique for the interactive rendering of volumetric models on projector-based multi-user light-field displays. The method achieves interactive performance and provides rapid visual understanding of complex volumetric datasets even when using depth-oblivious compositing techniques [Agus 08c, Agus 08a, JAIg 08].

- The development of a new interactive visualization framework which enables multiple naked-eye users to perceive detailed multi-gigavoxel volumetric models as floating in space, responsive to their actions, and delivering different information in different areas of the workspace [Agus 09, Igle 10]. The main contributions include a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of the display, as well as an out-of-core CUDA-based ray casting engine with a number of improvements over current GPU-accelerated volume ray-casters. The possibilities of the system have been demonstrated by the multi-user interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display driven by a cluster of PCs.

- The evaluation of volume rendering on light-field displays and its relevance for medical training and virtual examinations. Initial results demonstrate increased efficiency in tasks requiring spatial understanding. The development of preliminary psycho-physical tests which demonstrate that light-field displays and virtual reality improve understanding in common medical tasks [Agus 08b]. The continuation of such perceptual experiments to evaluate the depth discrimination capabilities of the light-field display technology with respect to two-view (stereo) and discrete multi-view designs [Agus 10a]. The evaluation employs a large scale multi-projector 3D display offering continuous horizontal parallax in a room size workspace. Two tests are considered in the context of depth oblivious rendering techniques: a layout discrimination task, and a path tracing task.

Out of the scope of this thesis, but always connected with the goal of improving volumetric data analysis tasks, additional results were achieved:

- The adaptation of an edge-directed optimization-based method for volumetric data super-sampling. The method faces the problem of partial volume effect by up-scaling the volumetric data, subdividing voxels in smaller parts and performing an optimization step keeping constant the energy of each original subdivided voxel while enhancing edge continuity [Giac 10b, Giac 10a].

- Motivated by problems of image segmentation in the medical field, we worked on a GPU framework based on explicit discrete deformable models, implemented on the NVIDIA CUDA architecture, aimed at the segmentation of volumetric images. The framework supports the segmentation of different volumetric structures in parallel, as well as interaction during the segmentation process and real-time visualization of the intermediate results [Schm 10, Schm 11].

7.2 Conclusions

A virtual environment based on a light-field display and a real-time multi-resolution volume rendering engine was realized. Such environment enables multiple naked-eye users to explore multi-gigavoxel volumetric models. The system was responsive to multiple user actions, showing models as floating in space and delivering different information in different areas of the workspace. This prototype can be considered the main tangible result of this work.

The system was integrated with standard medical image archives and extended the capabilities of current radiology workstations by supporting real-time rendering of volumetric information, such as CT, MRI, and PET scans, of potentially unlimited size on light-field displays. An out-of-core flexible ray casting engine that took advantage of current GPGPU architectures made possible to include a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of the display. The possibilities of the system were demonstrated by the multi-user interactive exploration of 64-GVoxel datasets on a 35-MPixel light-field display driven by a cluster of PCs.

The evaluation of volume rendering on light-field displays proved its relevance for medical training and virtual examinations. Initial results demonstrated an increased efficiency in tasks requiring spatial understanding. The development of preliminary psycho-physical tests demonstrated that virtual environments based on light-field displays improved the understanding in common medical tasks. Two tests were considered in the context of depth-oblivious rendering techniques: a layout discrimination task, and a path tracing task. Results confirmed that continuous multiview technology is able to elicit depth cues more efficiently with respect to standard stereo systems, providing clear advantages in typical analysis tasks like network structures understanding. Furthermore, our results indicated that depth-perception capabilities are closely related to the number of views provided by multiview systems.

Furthermore, we proposed different solutions for particular problems that can easily arise in volume rendering. First, we proposed an alternative volume representation primitive (*split-voxel*) suitable for cases where the volumes represented physical objects with well defined boundaries separating different materials, giving rise to models with quasi-impulsive gradient fields. Second, and in order to improve the analysis of medical data we proposed an edge-directed optimization-based method for volumetric data super-sampling. The method faced the problem of partial volume effect by up-scaling the volumetric data.

The potential of the *split-voxel* representation has not yet been adequately explored. To this end, we plan to investigate other compositing strategies, in order to find more meaningful visualization metaphors, such as layer extraction, or non-photorealistic illustrative methods. Moreover, when dealing with massive models, we plan to further evaluate the compression capabilities of the *split-voxel* primitive.

One general limitation of our multi-resolution volume rendering engine is that, just before rendering, it needs to perform decompression of the data contained in the volume bricks. This behaviour limits the quantity of information that can be cached in the GPU memory and therefore compromises the quality and detail of the multi-level representation maintained out-of-core. For this purpose, further work on lossless volume compression and direct rendering from compressed data is a worthwhile work to be done in the future.

7.3 Bibliographical Notes

Most of the scientific results obtained during this Ph.D. work also appeared in related publications. Specifically, paper [Gobb 08] presented the single-pass GPU ray-casting framework for interactive out-of-core rendering of massive volumetric datasets. The article [Igle 10] presented an adaptive out-of-core rendering engine based on CUDA that contained a number of improvements over previous GPU volume renderers. The paper [Agus 10b], presented the *split-voxel* primitive for discontinuity-preserving voxel representation of volumetric data encoding scalar data together with edge detection information.

The major part of the light-field content presented in this thesis was based on paper [Agus 08c], where we presented a GPU-accelerated volume ray-casting system interactively driving a multi-user light-field display. The content of the latest validation results came from reference [Agus 10a], where we presented a set of perceptual experiments completed to evaluate the depth discrimination capabilities with respect to two-view (stereo) and discrete multi-view designs. Previous results on the validation and more information about rendering applications on light-field displays can be found on [Agus 08a, Agus 08b, Agus 09], where we reported on a prototype system for medical data visualization and demonstrated the usefulness of the generated depth cues and the improved performance in understanding complex spatial structures with respect to standard techniques. Additional information regarding the scalability of the technique is available in [Igle 10], where we demonstrated the possibilities of the approach by the multi-user interactive exploration of massive volume datasets on a 35-MPixel light-field display driven by a cluster of PCs. More general considerations about human-computer interaction and elements affecting these kinds of virtual reality systems can be found in [JAIg 08].

Regarding the user-driven exploration of regions of interest in volumetric datasets, major part of the material were extracted from the paper [Luo 09], where we presented our *context-preserving focal probes* model. For a complete description of the prototype system integrating all the presented illustrative techniques, we refer the reader to our article [Igle 10], where we presented, among other things, the new accumulation scheme for importance-driven volume rendering and a set of specialized interactive illustrative techniques able to provide different contextual information in different areas of a light-field display.

Bibliography

[Adel 94] S. Adelson and C. Hansen. "Fast Stereoscopic Images with Ray-Traced Volume Rendering". In: *Symposium on Volume Visualization*, pp. 3–10, 1994.

[Agus 08a] M. Agus, A. Giachetti, E. Gobbetti, **J. A. Iglesias Guitián**, and F. Marton. "Towards advanced volumetric display of the human musculoskeletal system". In: *Eurographics Italian Chapter Conference*, Eurographics Association, Conference held in Salerno, Italy, June 2008.

[Agus 08b] M. Agus, A. Giachetti, E. Gobbetti, **J. A. Iglesias Guitián**, J. Nilsson, G. Pintore, and G. Zanetti. "Implementation and evaluation of an interactive volume visualization system on a lightfield display". In: *First 3D Physiological Human Workshop*, December 2008. Conference Abstract.

[Agus 08c] M. Agus, E. Gobbetti, **J. A. Iglesias Guitián**, F. Marton, and G. Pintore. "GPU Accelerated Direct Volume Rendering on an Interactive Light Field Display". *Computer Graphics Forum*, Vol. 27, No. 3, pp. 231–240, 2008. Proc. Eurographics 2008.

[Agus 09] M. Agus, F. Bettio, A. Giachetti, E. Gobbetti, **J. A. Iglesias Guitián**, F. Marton, J. Nilsson, and G. Pintore. "An interactive 3D medical visualization system based on a light field display". *The Visual Computer*, Vol. 25, No. 9, pp. 883–893, 2009.

[Agus 10a] M. Agus, E. Gobbetti, **J. A. Iglesias Guitián**, and F. Marton. "Evaluating layout discrimination capabilities of continuous and discrete automultiscopic displays". In: *Proc. Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, 2010. Paper 100, Electronic proceedings.

[Agus 10b] M. Agus, E. Gobbetti, **J. A. Iglesias Guitián**, and F. Marton. "Split-Voxel: A Simple Discontinuity-Preserving Voxel Representation for Volume Rendering". In: *Proc. Volume Graphics*, pp. 21–28, 2010.

[Balo 05] T. Balogh, T. Forgacs, T. Agocs, O. Balet, E. Bouvier, F. Bettio, E. Gobbetti, and G. Zanetti. "A Scalable Hardware and Software System for the Holographic Display of Interactive Graphics Applications". In: *Eurographics Short Papers Proceedings*, pp. 109–112, 2005.

[Bitt 04] J. Bittner, M. Wimmer, H. Piringer, and W. Purgathofer. "Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful.". *Computer Graphics Forum*, Vol. 23, No. 3, pp. 615–624, 2004.

[Boad 01] I. Boada, I. Navazo, and R. Scopigno. "Multiresolution volume visualization with a texture-based octree". *The Visual Computer*, Vol. 17, No. 3, pp. 185–197, 2001.

[Bouc 09] C. Boucheny, G.-P. Bonneau, J. Droulez, G. Thibault, and S. Ploix. "A Perceptive Evaluation of Volume Rendering Techniques". *ACM Transactions on Applied Perception*, Vol. 5, No. 4, pp. 23:1–23:24, Jan. 2009.

[Brow 05] M. Brown, A. Majumder, and R. Yang. "Camera-Based Calibration Techniques for Seamless Multiprojector Displays". *IEEE Trans. Vis. Comput. Graph*, Vol. 11, No. 2, pp. 193–206, 2005.

[Bruc 05] S. Bruckner and M. E. Groller. "VolumeShop: An Interactive System for Direct Volume Illustration". *Visualization Conference, IEEE*, Vol. 0, p. 85, 2005.

[Bruc 06a] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. "Illustrative context-preserving exploration of volume data". *IEEE Trans.Vis.Comput. Graph*, Vol. 12(6), pp. 1559–1569, 2006.

[Bruc 06b] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. "Illustrative Context-Preserving Exploration of Volume Data". *IEEE Trans. Vis. Comput. Graph*, Vol. 12, No. 6, pp. 1559–1569, 2006.

[Bruc 07a] S. Bruckner and M. E. Gröller. "Style Transfer Functions for Illustrative Volume Rendering". *Computer Graphics Forum*, Vol. 26, No. 3, pp. 715–724, 2007.

[Bruc 07b] S. Bruckner and M. E. Gröller. "Style Transfer Functions for Illustrative Volume Rendering". *Computer Graphics Forum*, Vol. 26, No. 3, pp. 715–724, Sep. 2007.

[Bruc 08] S. Bruckner. *Interactive Illustrative Volume Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 3 2008.

[Bruc 09] S. Bruckner and M. E. Gröller. "Instant Volume Visualization using Maximum Intensity Difference Accumulation". *Computer Graphics Forum*, Vol. 28, No. 3, p. , 2009.

[Cign 05] P. Cignoni, R. Scopigno, and M. Tarini. "A simple Normal Enhancement technique for Interactive Non-photorealistic Renderings". *Computer & Graphics*, Vol. 29, No. 1, pp. 125–133, 2005.

[Ciuc 10] I. Ciuciu, H. Kang, R. Meersman, J. Schmid, N. Magnenat-Thalmann, **J. A. Iglesias Guitián**, and E. Gobbetti. "Collaborative Semantic Content Management: an Ongoing Case Study for Imaging Applications". In: *Proc. 11th European Conference on Knowledge Management*, pp. 257–267, Conference held in Famalicao, Portugal, September 2010.

[Cohe 04] M. Cohen and K. Brodlie. "Focus and Context for Volume Visualization". In: *TPCG '04: Proceedings of the Theory and Practice of Computer Graphics 2004 (TPCG'04)*, pp. 32–39, IEEE Computer Society, Washington, DC, USA, 2004.

[Coss 07] O. Cossairt, J. Napoli, S. Hill, R. Dorval, and G. Favalora. "Occlusion-capable multiview volumetric three-dimensional display". *Applied Optics*, Vol. 46, No. 8, pp. 1244–1250, March 2007.

[Cras 09] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. "GigaVoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Rendering". In: *Proc. I3D*, pp. 15–22, 2009.

[Csbf 01] B. Csbfalvi, L. Mroz, H. Hauser, A. König, and M. E. Gröller. "Fast visualization of object contours by non-photorealistic volume rendering". *Computer Graphics Forum*, Vol. 20(3), pp. 452–460, 2001.

[DeFa 89] T. A. DeFanti, M. D. Brown, and B. H. McCormick. "Visualization: Expanding Scientific and Engineering Research Opportunities". *Computer*, Vol. 22, No. 8, pp. 12–25, 1989.

[Dodg 00] N. A. Dodgson, J. R. Moore, S. R. Lang, G. Martin, and P. Canepa. "Time-sequential multi-projector autostereoscopic 3D display". *J. Soc. for Information Display*, Vol. 8, No. 2, pp. 169–176, 2000.

[Dodg 02] N. Dodgson. "Analysis of the viewing zone of multi-view autostereoscopic displays". In: *SPIE Symposium on Stereoscopic Displays and Applications XIII*, pp. 254–265, 2002.

[Dodg 05] N. A. Dodgson. "Autostereoscopic 3D Display". *Computer*, Vol. 38, No. 8, pp. 31–36, 2005.

[Dodg 96] N. A. Dodgson. "Analysis of the viewing zone of the Cambridge autostereoscopic display". *Applied Optics: Optical Technology & Biomedical Optics*, Vol. 35, No. 10, pp. 1705–1710, 1996.

[Dono 99] D. Donoho. "Wedgelets: nearly minimax estimation of edges". *Ann. Statist.*, Vol. 27, No. 3, pp. 859–897, 1999.

[Eber 00] D. Ebert and P. Rheingans. "Volume illustration: non-photorealistic rendering of volume models". In: *VIS '00: Proceedings of the conference on Visualization '00*, pp. 195–202, IEEE Computer Society Press, Los Alamitos, CA, USA, 2000.

[Enge 01] K. Engel, M. Kraus, and T. Ertl. "High-quality pre-integrated volume rendering using hardware-accelerated pixel shading". In: *Proc. Graphics Hardware*, pp. 9–16, 2001.

[Enge 06] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-time Volume Graphics*. AK-Peters, 2006.

[Fava 01] G. Favalora, R. Dorval, D. Hall, and J. Napoli. "Volumetric three-dimensional display system with rasterization hardware". In: *Proc. SPIE*, pp. 227–235, 2001.

[Fuji 85] A. Fujimoto and K. Iwata. "Accelerated Ray Tracing". In: *CG Tokio*, 1985.

[Ghos 05] A. Ghosh, M. Trentacoste, and W. Heidrich. "Volume Rendering for High Dynamic Range Displays". In: *Eurographics/IEEE VGTC Workshop on Volume Graphics*, pp. 91–98, 2005.

[Giac 10a] A. Giachetti, **J. A. Iglesias Guitián**, and E. Gobbetti. "Edge adaptive and energy preserving volume upscaling for high quality volume rendering". In: *Eurographics Italian Chapter Conference*, Eurographics Association, Conference held in Genoa, Italy, November 2010.

[Giac 10b] A. Giachetti, **J. A. Iglesias Guitián**, and E. Gobbetti. "An Energy Preserving Upscaling Technique for Enhanced Volume Rendering of Medical Data". In: *Proc. 3DAnatomicalHuman Summer School*, 2010.

[Gobb 08] E. Gobbetti, F. Marton, and **J. A**. **Iglesias Guitián**. "A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets". *The Visual Computer*, Vol. 24, No. 7-9, pp. 797–806, 2008. Proc. CGI 2008.

[Govi 03] N. K. Govindaraju, A. Sud, S.-E. Yoon, and D. Manocha. "Interactive visibility culling in complex environments using occlusion-switches". In: *2003 ACM Symposium on Interactive 3D Graphics*, pp. 103–112, Apr. 2003.

[Guth 04] S. Guthe and W. Strasser. "Advanced techniques for high quality multiresolution volume rendering". *Computers & Graphics*, Vol. 28, pp. 51–58, February 2004.

[Hadw 03] M. Hadwiger, C. Berger, and H. Hauser. "High-Quality Two-Level Volume Rendering of Segmented Data Sets on Consumer Graphics Hardware". In: *Proc. Visualization*, pp. 301–308, 2003.

[Hadw 05] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. H. Gross. "Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces". *Comput. Graph. Forum*, Vol. 24, No. 3, pp. 303–312, 2005.

[Hall 91] M. W. Halle, S. A. Benton, M. A. Klug, and J. S. Underkoffler. "Ultragram: a generalized holographic stereogram". In: *Proc. SPIE*, pp. 142–155, July 1991.

[Hall 98] M. Halle. "Multiple viewpoint rendering". In: *Proc. SIGGRAPH*, pp. 243–254, 1998.

[Haus 01] H. Hauser, L. Mroz, G.-I. Bischiand, and M. E. Gröller. "Two-level volume rendering". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 7(3), pp. 242–252, 2001.

[Havr 98] V. Havran, J. Bittner, and J. Sára. "Ray Tracing with Rope Trees". In: L. S. Kalos, Ed., *14th Spring Conference on Computer Graphics*, pp. 130–140, 1998.

[He 96] T. He and A. Kaufman. "Fast stereo volume rendering". In: *Proc. Visualization*, pp. 49–ff., 1996.

[Herm 00] I. Herman, G. Melançon, and M. Marshall. "Graph visualization and navigation in information visualization: A survey". *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 6, No. 1, pp. 24 –43, jan-mar 2000.

[Hern 06] L. A. Hernández, J. D. Blanco, **José A. Iglesias**, J. Taibo, A. Seoane, A. Jaspe, and R. López. "El Museo Vacío: Uso de una instalación transitable de Realidad Virtual para la experimentación espacial de una unidad habitacional en un asentamiento preromano". In: *Proceedings of the 10th Iberoamerican Congress of Digital Graphics*, SIGRADI, Sociedad Iberoamericana de Gráfica Digital, Conference held in Santiago de Chile, Chile, 2006.

[Hern 07] L. A. Hernández, J. Taibo, D. Blanco, **José A. Iglesias**, A. Seoane, A. Jaspe, and R. López. "Physically Walking in Digital Spaces - A Virtual Reality Installation for Exploration of Historical Heritage". *International Journal of Architectural Computing (IJAC)*, Vol. 5, pp. 487–506, 2007.

[Hong 05] W. Hong, F. Qiu, and A. Kaufman. "GPU-based Object-Order Ray-Casting for Large Datasets". In: *Eurographics / IEEE VGTC Workshop on Volume Graphics*, pp. 177–186, June 2005.

[Hou 06] X. Hou, L.-Y. Wei, H.-Y. Shum, and B. Guo. "Real-time multi-perspective rendering on graphics hardware". In: *Proc. 17th Eurographics Workshop on Rendering*, pp. 93–102, 2006.

[Huan 03] R. Huang and K.-L. Ma. "RGVis: region growing based techniques for volume visualization". In: *Proc. Pacific Conf. Comput. Graph. and Applic*, pp. 355–363, 2003.

[Hueb 03] M. Huebschman, B. Munjuluri, and H. Garner. "Dynamic holographic 3-D image projection". *Optics Express*, Vol. 11, pp. 437–445, 2003.

[Hugh 09] D. Hughes and I. S. Lim. "Kd-Jump: a Path-Preserving Stackless Traversal for Faster Isosurface Raytracing on GPUs". *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 15, No. 6, pp. 1555 –1562, 2009.

[Igle 10] **J. A. Iglesias Guitián**, E. Gobbetti, and F. Marton. "View-dependent exploration of massive volumetric models on large-scale light field displays". *The Visual Computer*, Vol. 26, pp. 1037–1047, 2010.

[JAIg 08] **J. A. Iglesias Guitián** and M. Agus. "Interfacce uomo-macchina nella Realtà Virtuale". In: A. Soro, Ed., *Human Computer Interaction – fondamenti e prospettive*, pp. 289–330, Polimetrica, 2008. In italian.

[Jone 07] A. Jones, I. McDowall, H. Yamada, M. T. Bolas, and P. E. Debevec. "Rendering for an interactive 360 degree light field display". *ACM Trans. Graph*, Vol. 26, No. 3, p. 40, 2007.

[Ju 02] T. Ju, F. Losasso, S. Schaefer, and J. Warren. "Dual contouring of hermite data". *ACM Trans. Graph.*, Vol. 21, No. 3, pp. 339–346, 2002.

[Kaeh 06] R. Kaehler, J. Wise, T. Abel, and H.-C. Hege. "GPU-Assisted Raycasting for Cosmological Adaptive Mesh Refinement Simulations". In: *Eurographics / IEEE VGTC Workshop on Volume Graphics*, pp. 103–110, July 2006.

[Kers 06] M. A. Kersten, A. J. Stewart, N. Troje, , and R. Ellis. "Enhancing Depth Perception in Translucent Volumes". *IEEE Transactions on Visualization and Computer Graphics Journal*, Vol. 12, No. 6, pp. 1117–1123, 2006.

[Kind 03] 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*, pp. 513–520, October 2003.

[Knol 09] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. "Volume Ray Casting with Peak Finding and Differential Sampling". *IEEE Trans. Vis. Comput. Graph.*, 2009.

[Kobb 01] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. "Feature sensitive surface extraction from volume data". In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 57–66, ACM, New York, NY, USA, 2001.

[Koo 99] Y.-M. Koo, C.-H. Lee, and Y.-G. Shin. "Object-order template-based approach for stereoscopic volume rendering". *Journal of Visualization and Computer Animation*, Vol. 10, No. 3, pp. 133–142, 1999.

[Krau 02] M. Kraus and T. Ertl. "Adaptive Texture Maps". In: *Graphics Hardware 2002*, pp. 7–16, Sep. 2002.

[Krau 08] M. Kraus. "Pre-Integrated Volume Rendering for Multi-Dimensional Transfer Functions". In: *Proc. Volume and Point-based Graphics*, pp. 97–104, 2008.

[Krug 03] J. Krüger and R. Westermann. "Acceleration Techniques for GPU-based Volume Rendering". In: *Proc. Visualization*, pp. 287–292, 2003.

[Krug 06] J. Krüger, J. Schneider, and R. Westermann. "ClearView: An Interactive Context Preserving Hotspot Visualization Technique". In: *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2006)*, pp. 941–948, 2006.

[Kye 08] H. Kye, B.-S. Shin, and Y. G. Shin. "Interactive classification for pre-integrated volume rendering of high-precision volume data". *Graph. Models*, Vol. 70, No. 6, pp. 125–132, 2008.

[LaMa 99] E. C. LaMar, B. Hamann, and K. I. Joy. "Multiresolution Techniques for Interactive Texture-based Volume Visualization". In: *IEEE Visualization '99*, pp. 355–362, Oct. 1999.

[Lefe 05] S. Lefebvre, S. Hornus, and F. Neyret. *Octree Textures on the GPU*, pp. 595–613. Addison-Wesley, 2005.

[Li 03] W. Li, K. Mueller, and A. Kaufman. "Empty Space Skipping and Occlusion Clipping for Texture-based Volume Rendering". In: *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, p. 42, IEEE Computer Society, Washington, DC, USA, 2003.

[Liu 09] B. Liu. "Accelerating Volume Raycasting using Proxy Spheres". *Computer Graphics Forum*, Vol. 28, pp. 839–846(8), June 2009.

[Ljun 06] P. Ljung. "Adaptive Sampling in Single Pass, GPU-based Raycasting of Multiresolution Volumes". In: *Eurographics / IEEE VGTC Workshop on Volume Graphics*, pp. 39–46, July 2006.

[Lloy 03] S. Lloyd. "Least squares quantization in PCM". *Information Theory, IEEE Transactions on*, Vol. 28, No. 2, pp. 129–137, January 2003.

[Lum 04] E. Lum, B. Wilson, and K.-L. Ma. "High-Quality Lighting and Efficient Pre-Integration for Volume Rendering". In: *Proceedings of Joint Eurographics-IEEE TVCG Symposium on Visualization*, pp. 25–34, May 2004.

[Luo 09] Y. Luo, **J. A. Iglesias Guitián**, E. Gobbetti, and F. Marton. "Context Preserving Focal Probes for Exploration of Volumetric Medical Datasets". In: *Second 3D Physiological Human Workshop*, November 2009.

[Luo 93] L. Luo, C. Hamitouche, J.-L. Dillenseger, and J.-L. Coatrieux. "A moment-based three-dimensional edge operator". *IEEE Trans Biomed Eng*, Vol. 40, No. 7, pp. 693–703, 1993.

[Lux 09] C. Lux and B. Fröhlich. "GPU-Based Ray Casting of Multiple Multi-resolution Volume Datasets". In: G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. Encarnação, C. Silva, and D. Coming, Eds., *Advances in Visual Computing*, pp. 104–116, Springer Berlin / Heidelberg, 2009.

[Magn 09] N. Magnenat-Thalmann, J. Schmid, H. Delingette, M. Agus, and **J. A. Iglesias Guitián**. *3D Anatomical Modelling and Simulation Concepts*. Eurographics 2009 Course Notes, Eurographics 2009, Munich, August 2009.

[Matu 04] W. Matusik and H. Pfister. "3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes". *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 814–824, Aug. 2004.

[McCo 88] B. H. McCormick. "Visualization in scientific computing". *SIGBIO Newsl.*, Vol. 10, No. 1, pp. 15–21, 1988.

[McKa 00] S. McKay, G. Mair, S. Mason, and K. Revie. "Membrane-mirror based autostereoscopic display for teleoperation and telepresence applications". In: *Proc. SPIE*, pp. 198–207, 2000.

[Mora 04] B. Mora and D. S. Ebert. "Instant Volumetric Understanding with Order-Independent Volume Rendering". *Computer Graphics Forum*, Vol. 23, No. 3, pp. 489–497, 2004.

[Mora 05] B. Mora and D. S. Ebert. "Low-Complexity Maximum Intensity Projection". *ACM Transactions on Graphics*, Vol. 24, No. 4, pp. 1392–1416, October 2005.

[Pant 10] J. Pantaleoni, L. Fascione, M. Hill, and T. Aila. "PantaRay: fast ray-traced occlusion caching of massive scenes". *ACM Trans. Graph.*, Vol. 29, pp. 37:1–37:10, July 2010.

[Pavi 10] D. Pavic and L. Kobbelt. "Two-colored Pixels". *Computer Graphics Forum*, 2010. To appear.

[Popo 07] S. Popov, J. Günther, H.-P. Seidel, and P. Slusallek. "Stackless KD-Tree Traversal for High Performance GPU Ray Tracing". *Computer Graphics Forum*, Vol. 26, No. 3, pp. 415–424, Sep. 2007.

[Port 00] L. Portoni, A. Patak, P. Noirard, J.-C. Grossetie, and C. van Berkel. "Real-time auto-stereoscopic visualization of 3D medical images". In: S. K. Mun, Ed., *Proc. SPIE*, pp. 37–44, Apr. 2000.

[Pras 09] J.-S. Praßni, T. Ropinski, and K. H. Hinrichs. "Efficient Boundary Detection and Transfer Function Generation in Direct Volume Rendering". In: *Proceedings of the 14th International Fall Workshop on Vision, Modeling, and Visualization (VMV09)*, pp. 285–294, nov 2009.

[Rama 04] G. Ramanarayanan, K. Bala, and B. Walter. "Feature-Based Textures". In: *Proc. Rendering Techniques*, pp. 265–274, 2004.

[Rask 98] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. "The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays". In: *Proc. SIGGRAPH*, pp. 179–188, 1998.

[Relk 05] I. Relke and B. Riemann. "Three-dimensional multiview large projection system". In: *Proc. SPIE*, p. , 2005.

[Robe 00] J. W. Roberts and O. Slattery. "Display Characteristics and the impact on Usability for Stereo". In: *Proc. SPIE*, p. 128, 2000.

[Roet 03] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Straßer. "Smart hardware-accelerated volume rendering". In: *Proc. VisSym*, pp. 231–238, 2003.

[Ropi 05] T. Ropinski, F. Steinicke, and K. H. Hinrichs. "Tentative Results in Focus-Based Medical Volume Visualization". In: *Proceedings of the 5th International Symposium on Smart Graphics (SG06)*, pp. 218–221, Springer-Verlag, 2005.

[Rusi 06] S. Rusinkiewicz, M. Burns, and D. DeCarlo. "Exaggerated shading for depicting shape and detail". *ACM Trans. Graph.*, Vol. 25, No. 3, pp. 1199–1205, 2006.

[Scha 05] H. Scharsach. "Advanced GPU raycasting". In: *Proceedings of the 9th Central European Seminar on Computer Graphics*, pp. 69–76, May 2005.

[Schl 94] C. Schlick. "A fast alternative to Phong's specular model". In: *Graphics Gems IV*, pp. 385–387, Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[Schm 10] J. Schmid, **J. A. Iglesias Guitián**, E. Gobbetti, and N. Magnenat-Thalmann. "A GPU framework for parallel segmentation of volumetric images using discrete deformable models". In: *Proc. 3DAnatomicalHuman Summer School*, 2010.

[Schm 11] J. Schmid, **J. A. Iglesias Guitián**, E. Gobbetti, and N. Magnenat-Thalmann. "A GPU framework for parallel segmentation of volumetric images using discrete deformable models". *The Visual Computer*, Vol. 27, No. 2, pp. 85–95, 2011.

[Schr 96] W. Schroeder and B. Lorenson. *Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1996.

[Sen 04] P. Sen. "Silhouette Maps for Improved Texture Magnification". In: *Proc. Graphics Hardware*, pp. 65–74, 2004.

[Sere 06] P. Sereda, A. V. Bartroli, I. W. O. Serlie, and F. A. Gerritsen. "Visualization of boundaries in volumetric data sets using LH histograms". *IEEE Trans. Vis. Comput. Graph.*, Vol. 12, No. 2, pp. 208–218, 2006.

[Sigg 04] C. Sigg and M. Hadwiger. "Fast Third-Order Texture Filtering". In: *GPU Gems*, Morgan-Kaufmann, 2004.

[St H 95] P. St.-Hillaire, M. Lucente, J. Sutter, R. Pappu, C. G. Sparrell, and S. Benton. "Scaling up the MIT holographic video system". In: *Proc. 5th SPIE Symposium on Display Holography*, pp. 374–380, 1995.

[Stan 00] M. Stanley, P. Conway, S. Coomber, J. Jones, D. Scattergood, C. Slinger, B. Bannister, C. Brown, W. Crossland, and A. Travis. "A novel electro-optic modulator system for the production of dynamic images from giga-pixel computer generated holograms". In: *Proc. SPIE*, pp. 13–22, 2000.

[Steg 05] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. "A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting". In: *Eurographics/IEEE VGTC Workshop on Volume Graphics*, pp. 187–195, 2005.

[Tapp 06] A. Tappenbeck, B. Preim, and V. Dicken. "Distance-Based Transfer Function Design: Specification Methods and Applications". In: *Proceedings of SimVis 2006*, 2006.

[Tari 05] M. Tarini and P. Cignoni. "Pinchmaps: textures with customizable discontinuities". *Computer Graphics Forum*, Vol. 24, No. 3, pp. 557–568, Sep. 2005.

[Tumb 04] J. Tumblin and P. Choudhury. "Bixels: Picture Samples with Sharp Embedded Boundaries". In: *Proc. Rendering Techniques*, pp. 255–264, 2004.

[van 96] C. van Berkel, D. Parker, and A. Franklin. "Multiview 3D-LCD". In: *Proc. SPIE*, p. 32, 1996.

[Vapn 95] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[Viol 04] I. Viola, A. Kanitsar, and M. E. Groller. "Importance-Driven Volume Rendering". In: *Proceedings of the conference on Visualization '04*, pp. 139–146, IEEE Computer Society, Washington, DC, USA, 2004.

[Viol 05] I. Viola, A. Kanitsar, and M. E. Groller. "Importance-Driven Feature Enhancement in Volume Visualization". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 11, pp. 408–418, July 2005.

[Viol 06] I. Viola, M. C. Sousa, D. Ebert, B. Preim, B. Gooch, B. Andrews, and C. Tietjen. "Illustrative Visualization for Science and Medicine, Tutorial Notes". Eurographics Tutorials, 2006.

[Voll 06] J. E. Vollrath, T. Schafhitzel, and T. Ertl. "Employing Complex GPU Data Structures for the Interactive Visualization of Adaptive Mesh Refinement Data". In: *Eurographics / IEEE VGTC Workshop on Volume Graphics*, pp. 55–58, July 2006.

[Wan 04] M. Wan, N. Zhang, H. Qu, and A. E. Kaufman. "Interactive Stereoscopic Rendering of Volumetric Environments". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 10, No. 1, pp. 15–28, 2004.

[Ware 08] C. Ware and P. Mitchell. "Visualizing graphs in three dimensions". *ACM Trans. Appl. Percept.*, Vol. 5, No. 1, pp. 1–15, 2008.

[Wich 01a] F. A. Wichmann and N. J. Hill. "The psychometric function: I. Fitting, sampling, and goodness of fit.". *Perception and Psychophysics*, Vol. 63, No. 8, pp. 1293–1313, November 2001.

[Wich 01b] F. A. Wichmann and N. J. Hill. "The psychometric function: II. Bootstrap-based confidence intervals and sampling.". *Perception and Psychophysics*, Vol. 63, No. 8, pp. 1314–1329, November 2001.

[Wilh 92] J. Wilhelms and A. V. Gelder. "Octrees for faster isosurface generation". *ACM Transactions on Graphics*, Vol. 11, No. 3, pp. 201–227, July 1992.

[Will 03] R. Willett and R. Nowak. "Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging". *IEEE Trans. Med. Imag.*, Vol. 22, pp. 332–350, 2003.

[Wood 00] G. J. Woodgate, J. Harrold, A. M. S. Jacobs, R. R. Moseley, and D. Ezra. "Flat-panel autostereoscopic displays: characterisation and enhancement". In: *Proc. SPIE*, p. 153, 2000.

[Yang 06] R. Yang, X. Huang, S. Li, and C. Jaynes. "Toward the Light Field Display: Autostereoscopic Rendering via a Cluster of Projectors". In: *Eurographics Short Papers Proceedings*, 2006.

[Zhou 04] J. Zhou, A. Döring, and K. D. Tönnies. "Distance based enhancement for focal region based volume rendering". In: *Proceedings of Bildverarbeitung für die Medizin'04*, pp. 199–203, 2004.

[Zwic 06] M. Zwicker, W. Matusik, F. Durand, and H. Pfister. "Antialiasing for Automultiscopic 3D Displays". In: *Proc. Eurographics Symposium on Rendering*, pp. 73–82, 2006.

Curriculum Vitae

José Antonio Iglesias Guitián studied Computer Science at the University of A Coruña (UDC) in Spain, where he received his master's degree in February 2006. Since 2004, he started working as a researcher in the Visualization for Engineering Architecture and Urban Design Group (VideaLAB) at the School of Civil Engineering of the UDC, where he worked on several projects related to virtual reality and visualization. His research interests at that time included virtual humans, character animation techniques, 3D modelling and terrain rendering. In 2007, he was awarded with a Marie Curie Research Grant and was hired as an Early Stage Researcher in the Visual Computing group of CRS4 (Center for Advanced Studies, Research and Development in Sardinia). Since then, his research interests turn more towards multi-resolution data representation, time-critical graphics, external memory algorithms, rendering on light-field displays, GPGPU programming, volume rendering and illustrative techniques.

Contact Information

Name	José Antonio Iglesias Guitián
Address	CRS4, POLARIS Edificio 1, 09010 Pula (CA), Italy
Phone	+39 0709250283
E-Mail	jalley@crs4.it

Personal Details

Date of Birth	October 18th, 1980
Place of Birth	Sta. Uxía de Ribeira, Spain
Citizenship	Spanish
Languages	Spanish (native), Galician (native), Italian (fluent), English (fluent)

Education

since Jan. 2008	**Ph.D. Program in Electronics and Computer Engineering**, University of Cagliari, Italy. Working on the dissertation *Real-time GPU-accelerated Out-of-core Rendering and Light Field Visualization for Improved Massive Volumes Understanding* at the Department of Electrical and Electronic Engineering (DIEE). Tutor: Prof. M. Vanzi. Research Advisor: Dr. E. Gobbetti (CRS4).
Sep. 1998 - Feb. 2006	Computer Science Faculty, University of A Coruña (UDC), Spain **Master Studies in Computer Science**. Succesfully completed the masther's thesis *Animación en tiempo real de personajes e integración en un sistema de realidad virtual*. Advisors: Prof. A. Santos del Riego and J. Taibo Pena. Graduation with honors.

Employment History

since May. 2010	**Consultant Researcher** at the Visual Computing group of CRS4.
May. 2007 - May. 2010	**Marie Curie Early Stage Researcher** at the Visual Computing group of CRS4 (Center for Advanced Studies, Research and Development in Sardinia).
Oct. 2004 - Apr. 2007	**Researcher** at the Visualization for Engineering Architecture and Urban Design Group (VideaLAB) at the School of Civil Engineering of the University of A Coruña.

Professional Activities

Eurographics 2009	Tutorial on 3D Anatomical Modelling and Simulation Concepts.
HCI Meetings 2008	Course on Computer Graphics and Virtual Reality (in Italian).

Publications

Journal Articles

J. Schmid, **J. A. Iglesias Guitián**, E. Gobbetti, and N. Magnenat-Thalmann. "A GPU framework for parallel segmentation of volumetric images using discrete deformable models". *The Visual Computer*, Vol. 27, No. 2, pp. 85–95, 2011

J. A. **Iglesias Guitián**, E. Gobbetti, and F. Marton. "View-dependent exploration of massive volumetric models on large-scale light field displays". *The Visual Computer*, Vol. 26, pp. 1037–1047, 2010

M. Agus, F. Bettio, A. Giachetti, E. Gobbetti, **J. A. Iglesias Guitián**, F. Marton, J. Nilsson, and G. Pintore. "An interactive 3D medical visualization system based on a light field display". *The Visual Computer*, Vol. 25, No. 9, pp. 883–893, 2009

M. Agus, E. Gobbetti, **J. A. Iglesias Guitián**, F. Marton, and G. Pintore. "GPU Accelerated Direct Volume Rendering on an Interactive Light Field Display". *Computer Graphics Forum*, Vol. 27, No. 3, pp. 231–240, 2008. Proc. Eurographics 2008

E. Gobbetti, F. Marton, and **J. A**. **Iglesias Guitián**. "A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets". *The Visual Computer*, Vol. 24, No. 7-9, pp. 797–806, 2008. Proc. CGI 2008

L. A. Hernández, J. Taibo, D. Blanco, **José A. Iglesias**, A. Seoane, A. Jaspe, and R. López. "Physically Walking in Digital Spaces - A Virtual Reality Installation for Exploration of Historical Heritage". *International Journal of Architectural Computing (IJAC)*, Vol. 5, pp. 487–506, 2007

Conference Papers (based on full paper peer reviewing)

A. Giachetti, **J. A. Iglesias Guitián**, and E. Gobbetti. "Edge adaptive and energy preserving volume upscaling for high quality volume rendering". In: *Eurographics Italian Chapter Conference*, Eurographics Association, Conference held in Genoa, Italy, November 2010

M. Agus, E. Gobbetti, **J. A. Iglesias Guitián**, and F. Marton. "Split-Voxel: A Simple Discontinuity-Preserving Voxel Representation for Volume Rendering". In: *Proc. Volume Graphics*, pp. 21–28, 2010

M. Agus, E. Gobbetti, **J. A. Iglesias Guitián**, and F. Marton. "Evaluating layout discrimination capabilities of continuous and discrete automultiscopic displays". In: *Proc. Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, 2010. Paper 100, Electronic proceedings

I. Ciuciu, H. Kang, R. Meersman, J. Schmid, N. Magnenat-Thalmann, **J. A. Iglesias Guitián**, and E. Gobbetti. "Collaborative Semantic Content Management: an Ongoing Case Study for Imaging Applications". In: *Proc. 11th European Conference on Knowledge Management*, pp. 257–267, Conference held in Famalicao, Portugal, September 2010

Y. Luo, **J. A. Iglesias Guitián**, E. Gobbetti, and F. Marton. "Context Preserving Focal Probes for Exploration of Volumetric Medical Datasets". In: *Second 3D Physiological Human Workshop*, November 2009

N. Magnenat-Thalmann, J. Schmid, H. Delingette, M. Agus, and **J. A**. **Iglesias Guitián**. *3D Anatomical Modelling and Simulation Concepts*. Eurographics 2009 Course Notes, Eurographics 2009, Munich, August 2009

M. Agus, A. Giachetti, E. Gobbetti, **J. A. Iglesias Guitián**, and F. Marton. "Towards advanced volumetric display of the human musculoskeletal system". In: *Eurographics Italian Chapter Conference*, Eurographics Association, Conference held in Salerno, Italy, June 2008

L. A. Hernández, J. D. Blanco, **José A. Iglesias**, J. Taibo, A. Seoane, A. Jaspe, and R. López. "El Museo Vacío: Uso de una instalación transitable de Realidad Virtual para la experimentación espacial de una unidad habitacional en un asentamiento preromano". In: *Proceedings of the 10th Iberoamerican Congress of Digital Graphics*, SIGRADI, Sociedad Iberoamericana de Gráfica Digital, Conference held in Santiago de Chile, Chile, 2006

Conference Papers (based on abstract refereeing)

A. Giachetti, **J. A. Iglesias Guitián**, and E. Gobbetti. "An Energy Preserving Upscaling Technique for Enhanced Volume Rendering of Medical Data". In: *Proc. 3DAnatomicalHuman Summer School*, 2010

J. Schmid, **J. A. Iglesias Guitián**, E. Gobbetti, and N. Magnenat-Thalmann. "A GPU framework for parallel segmentation of volumetric images using discrete deformable models". In: *Proc. 3DAnatomicalHuman Summer School*, 2010

M. Agus, A. Giachetti, E. Gobbetti, **J. A. Iglesias Guitián**, J. Nilsson, G. Pintore, and G. Zanetti. "Implementation and evaluation of an interactive volume visualization system on a lightfield display". In: *First 3D Physiological Human Workshop*, December 2008. Conference Abstract

Book chapters

J. A. Iglesias Guitián and M. Agus. "Interfacce uomo-macchina nella Realtà Virtuale". In: A. Soro, Ed., *Human Computer Interaction – fondamenti e prospettive*, pp. 289–330, Polimetrica, 2008. In italian