1-1-2012

Context-Preserving Volume Image Exploration Using A 3D Painting Metaphor

Lev Faynshteyn
Ryerson University

CONTEXT-PRESERVING VOLUME IMAGE EXPLORATION USING A 3D PAINTING METAPHOR

by

Lev Faynshteyn, BSc, South-Russian State Technical University, Novocherkassk, Russia, 2001

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2012

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

CONTEXT-PRESERVING VOLUME IMAGE EXPLORATION USING A 3D PAINTING METAPHOR

Lev Faynshteyn

MSc, Computer Science, Ryerson University, 2012

ABSTRACT

This thesis combines a 3D interaction model with a *Maximum Intensity Difference Accumulation* (MIDA) volume visualization algorithm to create a technique for exploring volumetric datasets. The interaction model is based on a 3D *Painting* metaphor where a user selects a *Region of Interest* (ROI) by "painting" a 3D envelope enclosing features of interest. The result is an exploration technique that is intuitive to use and easy to learn even for non-expert users. The painting based model and the MIDA algorithm also provide visualization flexibility by allowing for different combinations of volumetric exploration operations. In addition, the various algorithms comprising the exploration technique have been implemented to take full advantage of parallel computational capabilities of modern *Graphics Processing Units* (GPUs), thus providing real-time interaction and high-quality visualisation. Finally, the contributions of the thesis are validated by a series of experiments and a user study.

ACKNOWLEDGEMENTS

I want to thank everybody who has helped me on this journey. My parents and extended family abroad, who have always supported me in every single one of my undertakings and who invariably in our weekly Skype chat sessions would half-jokingly ask me to come and visit them, even though they know I cannot until I am fully done with the work you are now reading. My friends, who inspired and gave me ideas over countless beer sessions. My girlfriend, who had to deal with a serious lack of attention, especially in the later stages of the project, but still stuck by me. My supervisor Dr. Tim McInerney, who always gave me valuable support and guidance and conversations with whom I enjoyed immensely. And finally, even though it might sound a bit vain, I want to thank myself for finally doing it. It had been a long time coming, but I finally did it!

TABLE OF CONTENTS

Chapter 1: Introduction 1

1.1 Contributions 3

1.2 Thesis Outline 7

Chapter 2: Literature Survey 8

2.1 Mapping From a Data Field to a Visual Field 8

2.1.1 One-Dimensional and Derivative-based Transfer Functions 10

2.1.2 Curvature or Shape Based Transfer Functions 13

2.1.3 Size-based Transfer Functions 15

2.1.4 Statistics-based Transfer Functions 16

2.1.5 Semantics-based Transfer Functions 18

2.1.6 Clustering-based Transfer Functions 19

2.1.7 Non-Transfer Function-based Mapping Methods 21

2.2 Illustrative Volume Visualization 23

2.2.1 Low-Level Abstraction Techniques 24

2.2.2 High-Level Abstraction Techniques 29

2.3 Volume Space Navigation 38

Chapter 3: Methodology and Implementation 42

3.1 ImageVis3D as an Implementation Platform 44

3.2 Introduction to Volume Rendering 45

3.2.1 Ray Casting 47

3.3 Methodology and Implementation Details 49

3.3.1 User Interface and Camera Control 49

3.3.2 Painting 51

3.3.3 GUI Controls 59

3.3.4 Maximum Intensity Difference Accumulation 62

3.3.5 Front- to-Back Depth Peeling 67

3.3.6 Envelope Definition with GPU Accelerated Metaballs 72

3.3.7 Surface Painting Mode Implementation 76

Chapter 4: Evaluation and Results 79

4.1 Exploration Flexibility 79

4.1.1 3D Painting Mode 80

4.1.2 2D Painting Mode 83

4.2 User Study 85

4.3 Performance 88

4.4 Extensibility 92

Chapter 5: Conclusion and Future work 94

5.1 Conclusion 94

5.2 Future work 95

LIST OF ACRONYMS

1. MIDA - Maximum Intensity Difference Accumulation
2. ROI - Region of Interest
3. GPU - Graphics Processing Unit
4. TF - Transfer Function
5. VR - Volume Rendering
6. DVR - Direct Volume Rendering
7. FtBDP - Front-to-Back Depth Peeling
8. GUI - Graphic User Interface
9. FPS - Frames per Second
10. MRI - Magnetic Resonance Imaging
11. CT - Computer Tomography
12. PET - Positron Emission Tomography
13. DICOM - Digital Imaging and Communications in Medicine
14. RGBA - Red Green Blue Alpha
15. MIP - Maximum Intensity Projection
16. LOD - Level of Detail
17. SDK - Software Development Kit
18. API - Application Programming Interface
19. ESS - Empty Space Skipping

LIST OF FIGURES

Figure 1: (a) A 1D TF specifying color and opacity values for the scull and skin and (b) the resulting volume image [3]. 2

Figure 2: Using multiple 2D projections of the volume to aid in the process of navigation and ROI selection in 3DSlicer [4]. 3

Figure 3: A series of 2D DICOM MRI slices is *stacked up* into a 3D volume. 9

Figure 4: An example of a 1D TF and the resulting image of a hand dataset in ImageVis3D [2]. 10

Figure 5: A 2D histogram of density values and gradient magnitudes. Some regions and boundaries are selected and assigned different colors. The resulting generated image is on the right [10]. 11

Figure 6: (a) Selecting different materials by coloring the diagonals on a histogram [12]. (b) An example of working with an LH histogram based TF [13].. 13

Figure 7: Non-photorealistic rendering using curvature based transfer functions [16]. 14

Figure 8: A process of classification of the blood vessels [17]. In the initial step the curve-skeleton is built. Then the user decides how to merge the detected shape primitives shown in different colors. 15

Figure 9: Comparison of the images produced using traditional (a) 1D and (b) 2D histogram based TFs with size-based TFs, (c) mapping size to only color and (d) to both color and opacity [19]. 16

Figure 10: Top row: images produces using a traditional 1D TF in VTK, a 2D lighting and gradient based TF and a multi-dimensional TF respectively. Bottom row: images generated using texture-based TF approach [20]. 17

Figure 11: Comparison of (a) 1D and (b) 2D TFs and (c) an LH histogram with (d) a statistical TF [22]. All TF types are used to try to classify the same brain tissue. 18

Figure 12: Visibility of different structures within a volume is controlled by the user by using a single slider to switch between instances of TFs [23]. 19

Figure 13: Comparison of single modal visualizations (two imaged on the left) to multimodal visualization (two images on the right) [24]. 19

Figure 14: Visualizations of the engine and the carp are produced by painting over LH clustered histograms [26]. 21

Figure 15: Images generated using (a) DSMIP [27] and (b) LMIP [28] visualization techniques. 22

Figure 16: Seamless transition between DVR, MIDA and MIP [5]. 23

Figure 17: (a) A pencil hand drawing of a staghorn beetle and (b) a VR generated image of a staghorn beetle using a similar illustrative style [29]. In the VR generated image the shape of the various parts of the beetle is emphasized by using a contouring technique [29]......................... 24

Figure 18: Lit sphere shading [30]. A 3D sphere represents all possible normal direction variations for any visualized object. The actual normal directions of the currently visualized object are encoded in the form of a sphere image. During the shading process the nx, y points (corresponding to the currently evaluated normal value) on the sphere image are mapped to the TF values.. 26

Figure 19: Visualizations using (a) a multi-dimensional style TF and (b) contouring [30]. 27

Figure 20: (a) Examples of using the view direction dependant stylized transparency and contouring techniques [30]. (b) Visualizations using the directional and omni-directional haloing effect respectively [30]. 28

Figure 21: A photograph of a plastinated anatomic model from Gunter von Hagens' *Body-worlds* exhibition (a) and a similar exploded-view image generated using VR (b) [29]......................... 29

Figure 22: The context in the image is achieved by making the parts of the body that occlude the features of interest transparent, while reducing transparency for the rest of the body [32]. 30

Figure 23: An illustration of the human heart as a combination of three layers (three images on the left: line drawing, DVR muscle layer, isosurface veins layer) with a rotation operation applied (right) [33]. 31

Figure 24: (a) Magic Lantern visualization using two different TFs [8]. (b) Magic Lens using masking and compositing technique [33]. (c) Magic Volume Lens using a magnification effect [34]. 31

Figure 25: "Zoom in" effect in action. The context is provided by using rectangular and circular ROIs, that are super-sampled using B-spline subdivision and enlarged [35]. 32

Figure 26: Images produced MIDA and distance-based opacity modulated DVR using different superquadratic envelope shapes [36]. 33

Figure 27: An interactive exploded-view illustration with two virtual hinges constraining the positioning of the exploded parts [37]. 34

Figure 28: Exploded views with different portioning settings applied [38]. 34

Figure 29: (a) An example of the spherical carving operation of the volume [39]. (b) A geometry based 3D deformable mesh is used to define the ROI and exclude all other parts from the visualization process [40]. (c) A section of the skin layer is cut-away after painting on the surface of the head and neck (c) [41]. 35

Figure 30: A free-from mask is painted on the surface on the skull. The rays are cast through the mask and any intersected parts of the volume are peeled away [42]. 36

Figure 31: (a) The "*leafing*" technique in action [43] and (b) the "*pliers*" operation applied to the internal organs of the frog while the skin is being retracted [44]. 37

Figure 32: An example of (a) a retracting deformation and (b) a bending deformation [45]. 38

Figure 33: A simulation of a whiplash injury using constrained deformations [46]. 38

Figure 34: Various rotational and translational widgets from 3DS Max, Blender, XSI, Houdini, Mode and Maya [49]. All share the same underlying principle. 39

Figure 35: The 3D shape of the interior surface of the blood vessels is used to constrain the possible camera travelling paths [53]. 40

Figure 36: A series of screenshots taken during the volume navigation process using the context-aware volume navigation technique [57]. 41

Figure 37: Voxels representing a discretized volumetric object [6]. 47

Figure 38: Rays cast through a volume are sampled between starting (f_n) and ending (l_n) positions [6]. 48

Figure 39: User interface with the GUI controls (on the left) and the visualization area (on the right). 50

Figure 40: A 2D envelope defined in the space of the screen as three 2D stripes. The shape and position of the envelope stays the same regardless of the orientations of the visualized volumetric object. 51

Figure 41: A 3D envelope defined in the space of the target volumetric object. The position and orientation of the envelope change according to the position and orientation of the visualized object. 52

Figure 42: Envelope painting in 3D depicted as a sequence of steps of: a) previewing the results of the visualization by moving the brush tip to the desired location; b) placing a new brush stroke at the brush tip's current location. 53

Figure 43: The painting plane always stays parallel to the screen plane, regardless of the orientation of the volumetric object in the 3D scene space. The current depth of the brush tip determines the depth of the painting plane, which can be adjusted within limits specified by the near and far clipping planes. When the user moves the mouse cursor on the screen plane its current position is projected onto the painting plane, thus defining the current 3D coordinates of the brush tip. New brush strokes can then be placed at the current 3D position of the brush tip on the painting plane, thus allowing the user to extend the envelope as needed. 56

Figure 44: (a) A negative brush tip erasing a part of the previously painted envelope. (b) A cylindrical brush tip self-aligned with the cross-section of a finger and depth-adjusted to show the whole digit of the finger. (c) A flattened envelope defined using a cylindrical brush tip and surface-painted following the curvature of the skull. 58

Figure 45: Visualizations of the palm using different visible value ranges and base colors for parts of the volume inside and outside the envelope. 59

Figure 46: A green 3D envelope visualized in an opaque, semitransparent and transparent manner.. 60

Figure 47: (a) A small-sized brush tip used to define the envelope tracing the outline of the veins on the surface of the palm. (b) A large brush tip used to quickly envelope a big part of the palm. (c) A negative brush tip used to “push away” the previously painted envelope.......................... 60

Figure 48: A transparent outlined version of a 2D envelope vs. a solid semitransparent version.61

Figure 49: A ray traversing a volumetric space, while encountered voxels are mapped to color and opacity values and blended with the previously accumulated results. 63

Figure 50: A comparison between (a) typical DVR and (b) MIDA ray profiles. In the case of MIDA the accumulated opacity value (red line) is modulated by the amount of the difference between the current and the previous local data value maximum (grey line). The blue line represents the accumulated color intensity of the resulting projected pixel on the screen. 64

Figure 51: In the absence of the brightness compensation (a) a visualization of a hand using a wide MIDA sub-range has more definition, but is too dim. (b) Under the same conditions a much narrower MIDA sub-range produces a less detailed but noticeably brighter image. 66

Figure 52: A visualization of the same volumetric dataset (a) without and (b) with brightness compensation applied.. 66

Figure 53: A series of images displaying peeling of the fragments using DP algorithm [64]...... 68

Figure 54: DP stripping away layers with each successive pass. The first pass peels away the front-most (leftmost) fragments, with hidden fragments shown in thin black lines. In the subsequent passes the already peeled away fragments are shown in light grey lines.................. 69

Figure 55: (a) A normal map without filtering applied. (b) A normal map with the 1x9 horizontal Gaussian kernel applied. (c) A normal map with the 9x1 vertical Gaussian kernel applied. 78

Figure 56: The results of visualizing different volumetric datasets ((a) a head , (b) a palm and (c) two feet) while providing the desired level of contextual information....................................... 81

Figure 57: Carving and peeling are used to expose internal features and to remove the occluding parts in volumetric datasets. 83

Figure 58: (a) An orange visualized without an envelope applied. (b) A 3D envelope is painted enclosing a part of the orange. (c) A 2D envelope is used to see all the way through the orange to better reveal internal structures. 84

Figure 59: Examples of pre-rendered images from each trial group: (a) an example of a 2D painting trial, (b) an example of a 3D carving trial and (c) an example of 3D context-preserving trial. 86

Figure 60: The results of the user study questionnaire based on a 7-point Likert scale. 87

Figure 61: Average FPS measured under different visualization conditions. 90

Figure 62: A head dataset visualized at (from left to right) 100%, 65% and 30% sampling rate. 91

Figure 63: A head dataset visualized at different LODs (the highest on the left to the lowest on the right). 92

Figure 64: (a) The 2D painting mode with 1D and 2D TF rendering modes. (a) An example of the 3D surface painting over the surface defined by a 1D TF. (c) A combination of MIDA and 2D VR techniques using a free-hand painted 3D envelope. 93

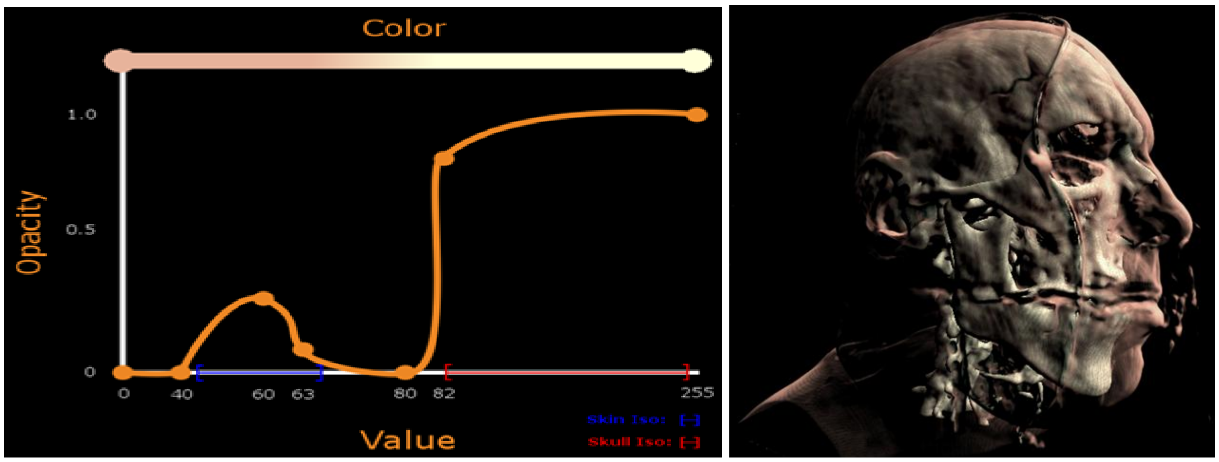
Chapter 1: Introduction

The visualization of volumetric data is a rapidly evolving area in the field of computer graphics, especially in the context of medical applications. The advent of modern *Graphics Processing Units* (GPUs) along with the development of various computational and visualization techniques has resulted in previously unattainable levels of user interactivity and realism. Medical professionals can harness this power to explore and manipulate volumetric images in order to get a better understanding of the underlying data and processes, often in ways that would be impossible to replicate in the real world.

Nevertheless, when it comes to providing the user with easily understandable and predictable ways of specifying how the data is to be visualized, there is still a lot of room for improvement. Traditionally this *mapping* from a scalar value field of a volumetric dataset to a visual field of pixel colors on a screen was and, in the vast majority of cases, still is specified by the user with *Transfer Functions* (TFs) (Figure 1). Despite the significant amount of research that has been done in this area [1], specifying a TF that achieves the desired visual effect still often remains a non-trivial task, even for expert users.

This problem of finding a perfect TF to achieve the desired visualization can also be compounded by the necessity of having different areas of the volume visualized using different sets of parameters in order to preserve the visual context. This secondary goal presents a challenge in its own right as navigation in the 3D space of a volume and specification of different *Regions of Interest* (ROIs) in a 3D dataset is often a complex operation. Problems such as ROI depth specification and depth perception as well as volumetric feature separation are among the most prominent issues. The traditional approach to the ROI specification and feature separation problems is to the use multiple 2D data views along with a variety of 2D ROI editing capabilities. The multiple 2D views, however, often complicate the user interaction model and make it difficult for the user to understand the spatial relationship between the various 2D and 3D views (Figure 2).

In an attempt to address the aforementioned challenges this thesis presents a prototype volumetric data visualization and exploration system, implemented as a series of extensions to an open-source *Volume Rendering*[1] (VR) (Section 3.2) program called ImageVis3D [2]. The extensions add new visualisation modes and a new user interaction model that will be covered in detail in the following chapters.



(a) (b)

Figure 1: (a) A 1D TF specifying color and opacity[2] values for the scull and skin and (b) the resulting volume image [3].

[1] In computer graphics in general, rendering refers to the process of generating a 2D image from models in a virtual scene. In the case of the 3D computer graphics rendering is typically achieved by projecting 3D models in a scene onto a 2D surface of a screen. In its turn, volume rendering is a set of techniques used to generate 2D projections of discretely sampled 3D datasets.

[2] In computer graphics opacity describes the level of a material's impenetrability to light.

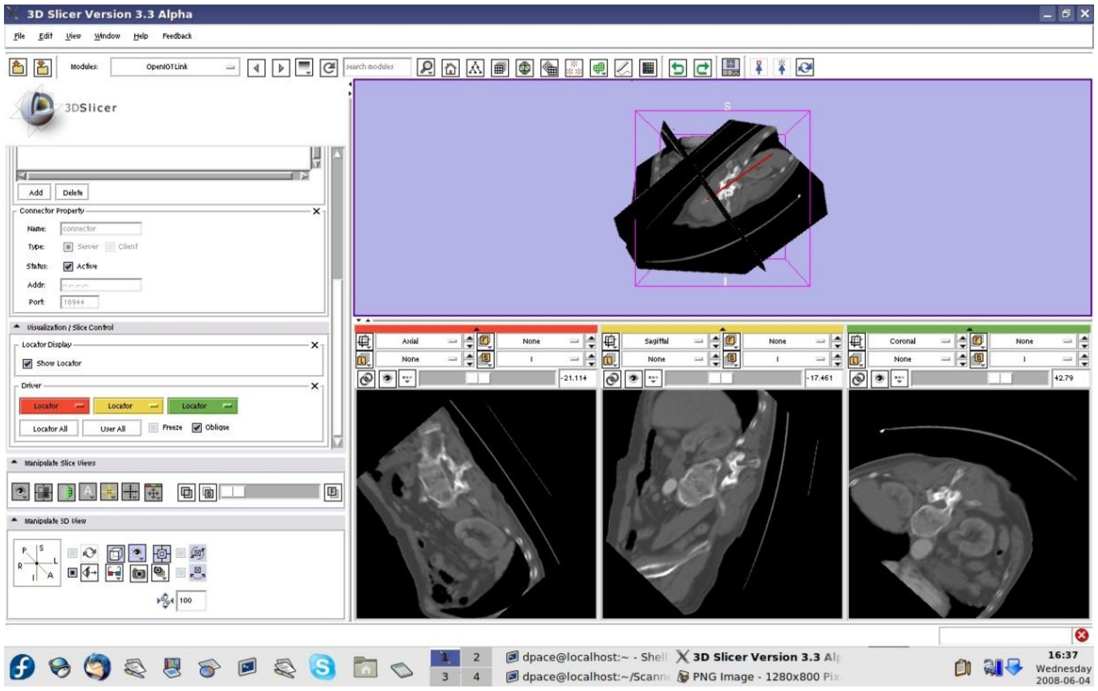


Figure 2: Using multiple 2D projections of the volume to aid in the process of navigation and ROI selection in 3DSlicer [4].

1.1 Contributions

The primary goal of this thesis was to provide an alternative approach to volumetric dataset exploration that would combine qualities such as a fast learning curve, intuitiveness, ease of control and flexibility[3]. A user study (Section 4.2) was conducted in order to evaluate the first three qualities. A series of experiments presented in Section 4.1 demonstrates the flexibility of the proposed approach by generating various context-preserving visualizations of several volumetric datasets. The ability of the system to provide visually appealing and contextually meaningful visualizations at interactive frame-rates throughout the whole interaction and exploration user experience was also paid a lot of attention. Section 4.3 is dedicated to the performance evaluation of the system under different visualization conditions. Finally, designing

[3] Flexibility here means providing several ways of visualizing features of interest, i.e. ROIs.

and implementing a future-ready visualization model should consider the question of extensibility[4] as well. Section 4.4 demonstrates the extensibility potential of the proposed approach by integrating two additional visualization modes into the system.

To meet all of the aforementioned goals an interaction model based on a painting metaphor was conceived and implemented. The painting interaction model allows a user, with just a few *brush strokes*, to define an ROI by painting a 2D or 3D envelope in the space of a volumetric scene to achieve the desired visual effect and contextual representation of volumetric information. A number of algorithmic, visual and user interaction techniques have been explored and implemented as contributing parts to the final solution of volumetric dataset exploration. The following subsections break down the contributions that resulted from this work.

An Efficient Implementation of the Maximum Intensity Difference Accumulation Visualization Mode

A *Maximum Intensity Difference Accumulation* (MIDA) visualization technique was proposed by Bruckner et al. [5] as a means of visualizing volumetric datasets without the necessity to specify TFs (Section 2.1). It takes a conventional *Direct Volume Rendering* (DVR) [6] (Sections 3.2, 3.3.4) approach and modifies it to exploit inherent data characteristics of the underlying volumetric dataset by modulating the accumulated opacity of the image in accordance with changes in the data values. To control what features of a volumetric dataset should have the most prominence in the resulting image, only a range defined by the minimum and maximum data intensity values needs to be specified.

[4] Extensibility here means the ability to extend the visualization capabilities of the system by adding alternative visualization modes.

A 2D/3D Painting Model

In order to provide a non-expert user with an intuitive way of specifying an ROI, a 2D and 3D interaction model based on a painting metaphor was conceived and implemented. This model allows a user to paint and edit a 2D or 3D envelope (i.e. a contour and a closed surface respectively) either in screen space or volume object space respectively, and utilizes Metaballs[5] [7] (Section 3.3.6) as the underling mechanism for implicit specification of an envelope's shape. MIDA or any other visualization technique can be specified for parts of the volume that fall inside and outside an envelope to achieve complex and visually compelling results.

User Interactions within a Single 3D View

As a mechanism to visualize a 3D envelope an approach based on combining polygon-based and volumetric geometry rendering has been implemented. An implementation of the *Front-to-Back Depth Peeling* (FtBDP) (Section 3.3.5) technique was used to allow for seamless integration of both types of rendering, which in turn provides enough visual cues for a user to be able to perform all of the manipulations on the volumetric data in a single 3D view. This means that a common practice of providing additional axially aligned 2D volume data projections aiding in the exploration process can be abandoned in favor of an uninterrupted and fluid user experience.

Different Types of Volume Exploration within a Single Generic Paradigm

As will be shown in Chapter 3:, a combination of the 2D and 3D painting and the MIDA technique allows for many types of volume data exploration and editing operations (such as 3D

[5] Metaballs are organic looking *n*-dimentional objects, whose surface is implicitly defined by a chosen (e.g. Gaussian) field function.

contextual views, 3D volume “*carving*”, or “*Magic Lantern*”[6] [8] like behaviour, etc.) within a single user interaction model.

Fully GPU Accelerated Visualization Pipeline

One of the goals of the project was to provide the user with an interactive and semi-immersive environment, where visual feedback to user interactions with a *Graphic User Interface* (GUI) would be instantly reflected in the resulting image on the screen. To achieve this goal heavy emphasis was placed on utilizing GPU parallel computational capabilities to alleviate computational burden on the CPU. As a result, all stages of the visualization pipeline have been implemented using GPU-based “*shader*” programs [9] written in *OpenGL Graphics Shading Language*. This implementation achieves interactive frame-rates for a wide array of volumetric datasets of varying sizes on the current generation of GPUs found in a modern desktop computer and will also take full advantage of additional computational power of forthcoming generations of GPUs.

Extensible Visualization Model

MIDA is used as the default visualization mode to allow for visualization of volumetric datasets with a minimum of parameter specification. This mode is, however, limited in terms of visual results that can be achieved, and thus the architecture and the code structure of the extensions were designed and implemented in such a way as to allow for this technique to be easily substituted by any other DVR (3.2) method.

---

[6] Magic Lantern or Magic Lens is a metaphor commonly used to describe methods of volumetric visualization where a user guides some kind of a virtual optical device (e.g. a flash light or a lens) to highlight internal structures in a certain region of an object.

1.2 Thesis Outline

Chapter 2 explores the multitude of existing solutions to the problems of volumetric data visualization and exploration. Various TF types and their parameter specification techniques, ROI definition approaches and user interaction models will be covered.

Chapter 3 begins by explaining the reasons behind choosing ImageVis3D [2] as the base platform for the thesis implementation. It then provides a brief introduction to *Volume Rendering* (VR) in general and *Ray Casting* in particular. After that the user interaction model realizing the 2D and 3D painting metaphor is explained, providing the necessary context for the detailed descriptions of various implemented algorithms (MIDA, FtBDP, Metaballs, etc.) that conclude the chapter.

Chapter 4 presents both quantitative and qualitative results to validate the thesis contributions. Quantitative results are centered around measuring the performance of the system (average *Frames per Second* (FPS)) using different datasets of varying sizes. The qualitative part presents results of a user study conducted to evaluate the learning curve, ease of control, and intuitiveness of the implemented visualization techniques and of the user interaction model. Examples of different contextual volume exploration tasks are given to showcase the flexibility of the proposed exploration and visualization model. Finally, the extensibility of the implemented visualization model is validated by integrating two additional visualization modes.

Chapter 5 concludes by summarising the thesis work and discusses possible avenues for future work and improvements.

Chapter 2: Literature Survey

Volumetric dataset exploration is one of the supporting pillars in the field of volume graphics. After all, what is the use of any dataset, no matter how big and detailed, if one cannot identify and analyze features of interest within it? This chapter will present a survey of various techniques and approaches that deal with the problem of volume exploration. To further aid in the comprehension of the following material, the techniques will also be arranged into distinct categories.

It should be noted that this thesis and all of the material presented in it deals with *unsegmented* volumetric datasets exclusively. This is in contrast to *segmented* or *pre-authored* datasets, where a volume has been pre-processed and labelled in such a way as to provide additional information about the objects within. While some of the challenges associated with the visualization and exploration of segmented datasets are similar to those of unsegmented ones, they are by and large a separate topic and are thus not covered in the following material.

2.1 Mapping From a Data Field to a Visual Field

A volumetric dataset in its basic form is nothing more than an array of scalar values that has been obtained by means of scanning a physical phenomenon or an object or modelling/simulating an object or a process. In the medical field volumetric datasets are usually products of a scanning process (e.g. *Magnetic Resonance Imaging* (MRI), *Computer Tomography* (CT), *Positron Emission Tomography* (PET), etc.) and the values are obtained by measuring the intensity of a signal (e.g. X-rays for CT or radio-frequency waves for MRI) in different parts of the object being scanned. Typically scanning is performed in a slice-by-slice fashion, with the final volumetric representation assembled by *stacking up* the resulting 2D slices (Figure 3). The resolution of the scanning process and the size of each element in the final array are usually dictated by the nature of the underlying phenomenon or object and the desired precision of its electronic representation.

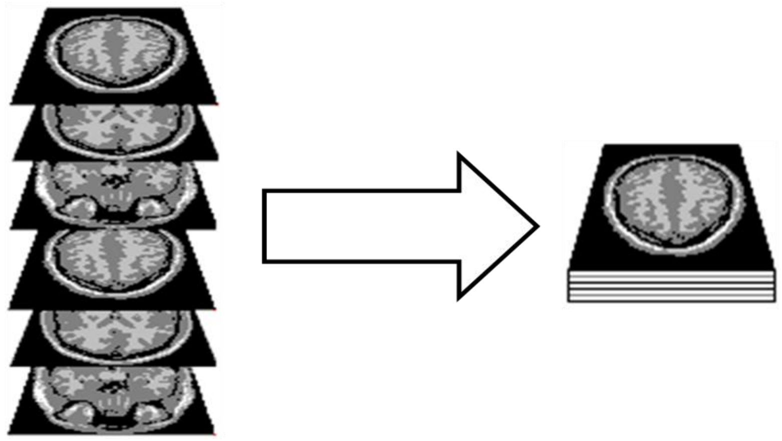


Figure 3: A series of 2D DICOM MRI slices is *stacked up* into a 3D volume.

In order to be able to view and analyze the data in a volumetric dataset in ways comprehensible by humans it is necessary to map values from a dataset's scalar representation to a visual representation. To put it more simply, each scalar value has to have a visual counterpart defined by a color and opacity pair (Section 3.2). This mapping from a scalar to visual field traditionally has been defined by what is known as a *Transfer Function* (TF) (Figure 4). The following subsections will summarize and classify the most commonly used types of TFs, with the emphasis on those that are specifically aimed at facilitating a more meaningful data exploration process. Several other non-TF based approaches that have emerged in an attempt to avoid the often convoluted and unintuitive process of specifying TFs are also described. The classification is based on work presented in [1] while also updating it with the latest research material and references.

2.1.1 One-Dimensional and Derivative-based Transfer Functions

The simplest form of a TF is a 1-dimentional TF. The term *1-dimensional* implies that a simple 1-to-1 or many-to-1 mapping exists from the scalar domain into the visual domain. To allow a user to define a 1D TF within a GUI of a program, typically a 2-dimentional histogram is pre-computed in which one axis corresponds to every scalar value encountered in the dataset and the other axis depicts the frequency of encountering each particular value. In Figure 4 the red, green, blue and white lines on the histogram define how RGBA values corresponding to the mapped scalar values will be generated. The projection of a point from each line onto the X-axis determines the scalar value to be mapped, the projection onto the Y-axis determines the intensity of a mapped R, G, B or A value from 0 to 1, and the projection onto the gradient colored bar at the top (the gradient coloration is defined by a combined color depending on the XY-projected values of all the lines) specifies the opacity of the mapped value.

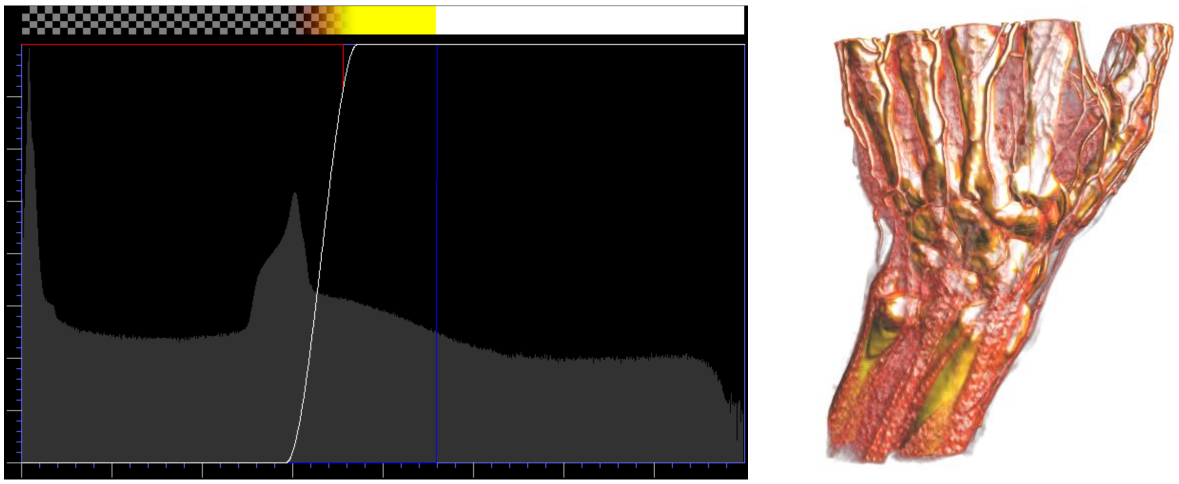


Figure 4: An example of a 1D TF and the resulting image of a hand dataset in ImageVis3D [2].

However, even though for some datasets 1D TFs are sufficient to achieve the desired visual results, more often than not they are afflicted by poor data feature separation (for instance in Figure 4 some arteries cannot be distinguished from the bones due to similar underlying scalar intensity values). This is due to the fact that intensity values alone represent only a small fraction of information contained in any volumetric dataset. To address this problem various multi-dimensional representations of the underlying volumetric data, that are based on utilizing scalar value derivatives of first and higher orders, are used. In the simplest and most commonly used case, only the first derivative is computed for each value inside a volumetric dataset, representing a gradient, or simply a direction of the most prominent change in the values of the dataset at a particular point in space. A histogram with scalar and gradient values arranged along horizontal and vertical axes respectively can then be generated [10], allowing for the visualization of material boundaries in a dataset in the form of arches (Figure 5). The two ends of an arch correspond to the homogenous regions of two different materials within a volume that have a gradient value of 0, while the top of the arch correspond to the border between the materials where the magnitude of the gradient is at its maximum. Usually various graphical widgets are used to allow the user to select regions representing certain materials and boundaries while assigning different color and opacity values to them (Figure 5).

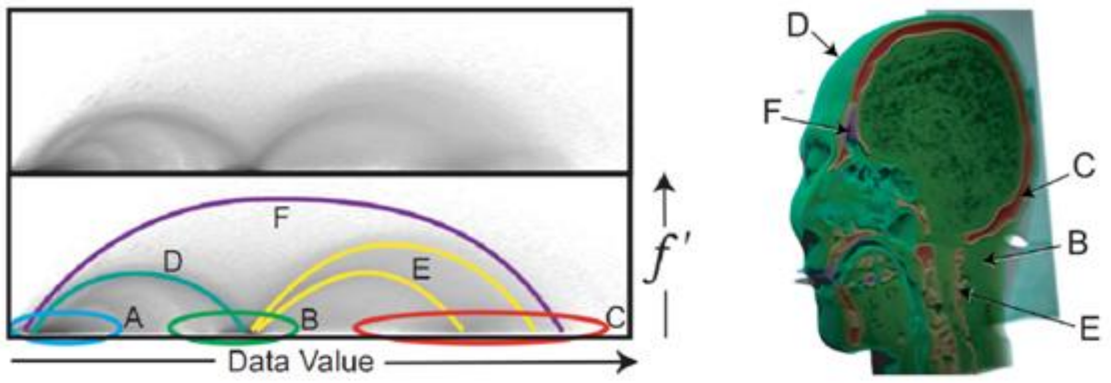
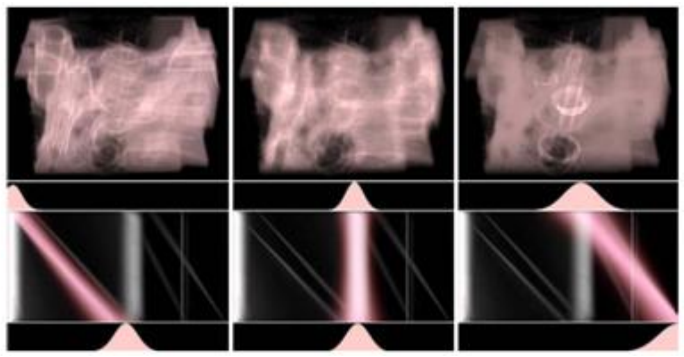


Figure 5: A 2D histogram of density values and gradient magnitudes. Some regions and boundaries are selected and assigned different colors. The resulting generated image is on the right [10].

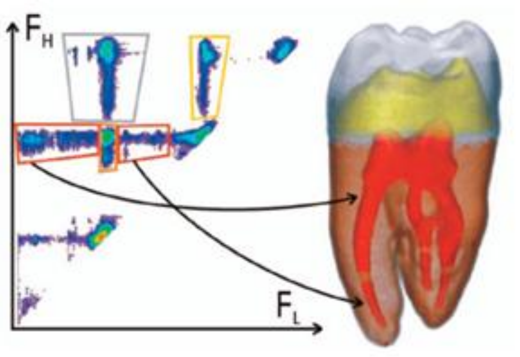
One of the problems with the above method is that material density values and gradient magnitudes can overlap. One possible solution is to add another component (i.e. a second derivative) to the histogram as a third dimension representing gradient direction. However, this additional dimension introduces a user interaction problem, as specification of ROIs on a 3D histogram becomes progressively more difficult and time consuming.

In addition these methods do not address such problems as noise in the data, partial volume effects and false material border detection due to biasing[7] [11], all of which can make selection of the features of interest virtually impossible using a traditional 2D histogram. One of the solutions to these problems is presented in [12], where the authors use two density values from the opposite sides of the gradient border to represent the data in a form of *diagonals*. A TF thus can be created by coloring sections of the diagonals (Figure 6(a)). Another popular approach is proposed in [13]. It uses the idea of so called *LH Histograms* that show lower and higher intensities of the materials that form the boundaries (Figure 6 (b)). This approach performs better in terms of noise, bias, and partial volume effects and has been used in a number of medical volume rendering programs including Voreen [14] and OsiriX [15].

[7] Applied to volumetric datasets in general, biasing is manifested as nonuniformities in the intensity values of an obtained volumetric dataset caused by some high-field MRI scanners. This in turn can cause the generation of the false material borders in a histogram.



(a)



(b)

Figure 6: (a) Selecting different materials by coloring the diagonals on a histogram [12]. (b) An example of working with an LH histogram based TF [13].

However, even with all of the above improvements, selecting features on a 2D histogram can still be difficult. This is due to the fact that often it is hard to estimate what the final image will look like by selecting or coloring arbitrary regions on a histogram. In addition, especially in the case of unsegmented datasets, it is not easy to establish a correlation between the features on a histogram and the actual structures in the volume. Thus, numerous other approaches have been proposed, that attempt to alleviate these issues, with varying degree of success.

2.1.2 Curvature or Shape Based Transfer Functions

These methods are based on the idea of using geometrical features of the structures within a volume to aid in their classification. In [16] the curvature of structures is used as a basis for designing a TF and also as a means to produce non-photorealistic[8] images by highlighting the contours of the structures (Figure 7). First, a set of *isosurfaces*[9] (Section 3.3.6) within a volume is specified. Then an algebraic framework is used to calculate the curvature of the isosurfaces

[8] In computer graphics non-photorealistic rendering is an area that focuses on a variety of expressive styles (e.g. painting, drawing, technical illustration) to produce the resulting images.
[9] An isosurface is a 3D surface that represents points of a constant value (known as an *iso threshold* value) within the space of a volume.

using *differential invariants*. The drawback of this approach is that in noisier datasets (such as the ones usually obtained by means of an MRI scanning process) it suffers from spurious surface curvature variations. The authors try to combat this problem by smoothing the data values, while sacrificing accuracy and resolution.

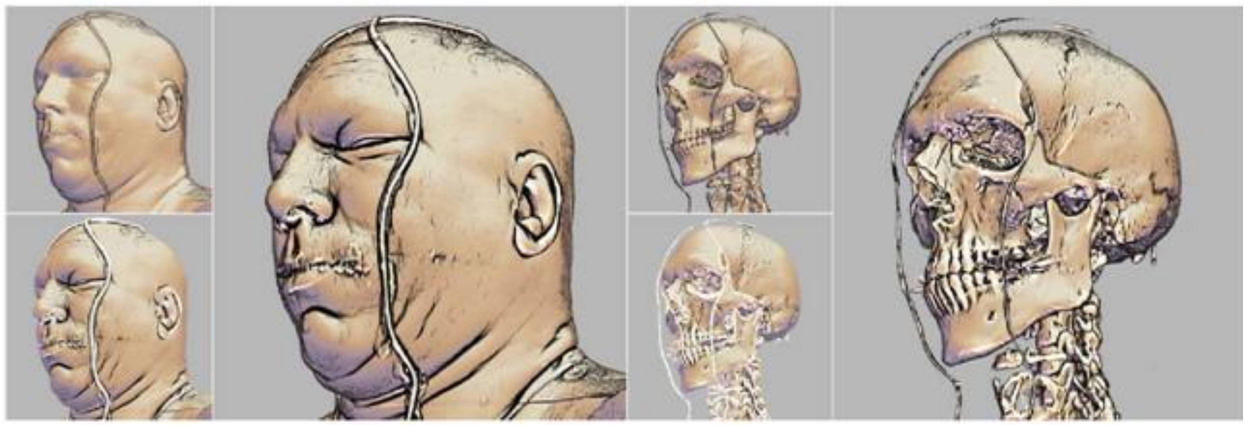


Figure 7: Non-photorealistic rendering using curvature based transfer functions [16].

In [17] the authors propose a method of classification based of the shape of volumetric features (Figure 8). They use a rough pre-segmentation process as the initial step for selecting an ROI within a volume. Pre-segmentation is done by specifying data intensity thresholds (a process called *windowing*), thus effectively defining isosurfaces (Section 3.3.6) approximating the shapes of the objects inside the volume. Then a so called *curve-skeleton* of each volume structure within the region is examined in order to try to fit one of the three predefined shape descriptors (*longitudinal, surface-like* and *blobby*) to them. The structures are then merged in a process supervised by the user. Additional classes of shape descriptors can be defined if necessary. The advantage of this method is that the user deals only with shape-classified volumetric features and avoids the difficulties of interpreting histogram data. The obvious disadvantage is the need to properly mask the volumetric shapes before they even can be shape-classified. Depending of the structure of the volume in question this can also present challenges.

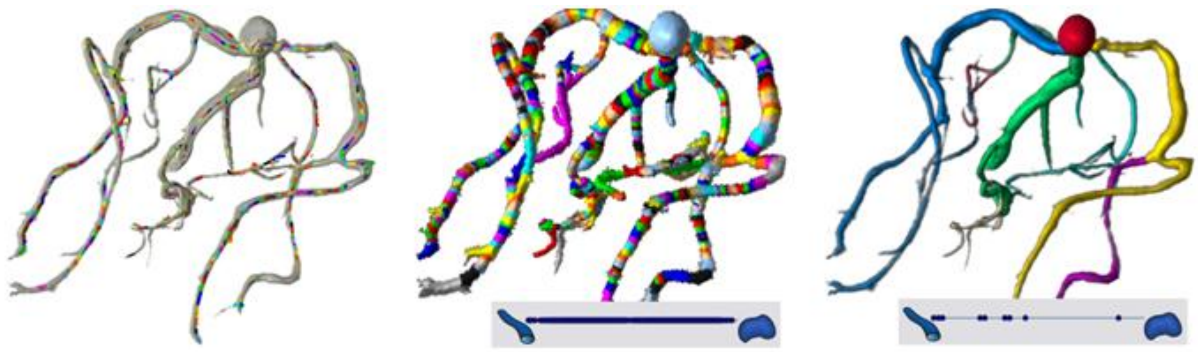


Figure 8: A process of classification of the blood vessels [17]. In the initial step the curve-skeleton is built. Then the user decides how to merge the detected shape primitives shown in different colors.

2.1.3 Size-based Transfer Functions

This group of TF generation methods tries to address the problem of identifying structures within a volume that have similar intensity values but differ in size. This work is mainly based on [18] and [19] and stems from the idea of so called *scale-fields* (which has its origins in computer vision). In the initial step a scale field for a volume is computed and shapes within a user specified size range are detected. As a result of this step a set of discrete points representing the most prominent scales in the volume is generated. These points are then interpolated and combined with the traditional histogram representation (Figure 9(c,d)).

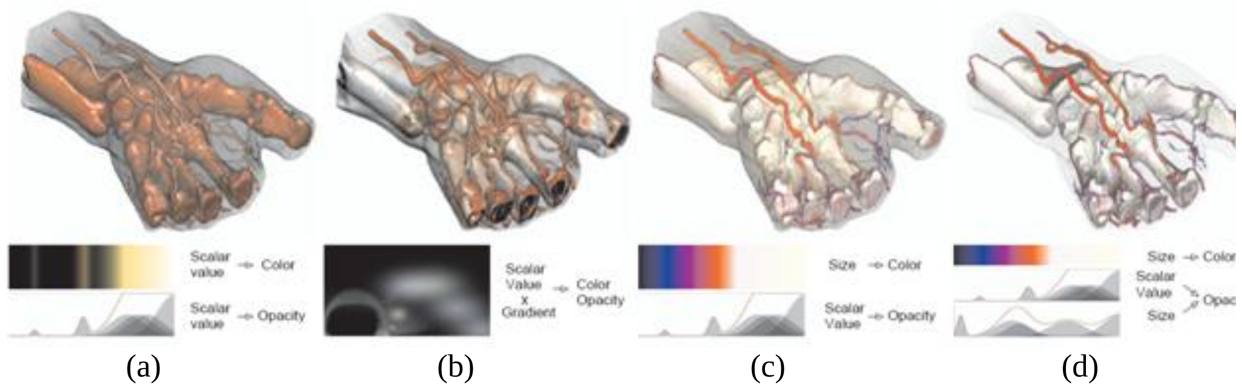


Figure 9: Comparison of the images produced using traditional (a) 1D and (b) 2D histogram based TFs with size-based TFs, (c) mapping size to only color and (d) to both color and opacity [19].

One of the major limitations of this approach is that the computation of a scale field is a very expensive operation. For a data set containing 512^3 values the computation time can exceed 10 seconds even on a rather powerful GPU[10]. A compromise obviously has to be chosen between the need to have interactive frame-rates and the size and/or resolution of the analyzed dataset.

2.1.4 Statistics-based Transfer Functions

These methods are based on capturing different structural and geometrical properties of volumetric datasets using various first-, second- and higher-order statistics and their combinations. Again, as in the case of size-based TFs, they are aimed at discriminating structures even when they have similar intensity values. In [20] the authors propose an approach based on analysing local textural properties of the volume (Figure 10). They capture both geometrical and structural properties by using *histogram statistics*, *run-length* and *co-occurrence matrices*.

[10] This figure is based on the test results with nVidia GeForce 8800 GTX. The latest generations of GPUs will most likely reduce this time by several orders.

For this technique to keep the data size and the computation times under control the dataset is first divided into overlapping sub-volumes, which are then analyzed separately. The textural properties that can be analyzed and captured include occurrence of the predefined intensity values, first order (e.g. *variance, skewness, kurtosis,* etc.) and second-order statistics (i.e. likelihood of observing two different intensity values at a certain distance). Overall there are 20 metrics that can be pre-computed and combined into a TF. Naturally the fact that this method works on the fixed size data extraction regions precludes it from differentiating between homogenous and inhomogeneous regions in a dataset.

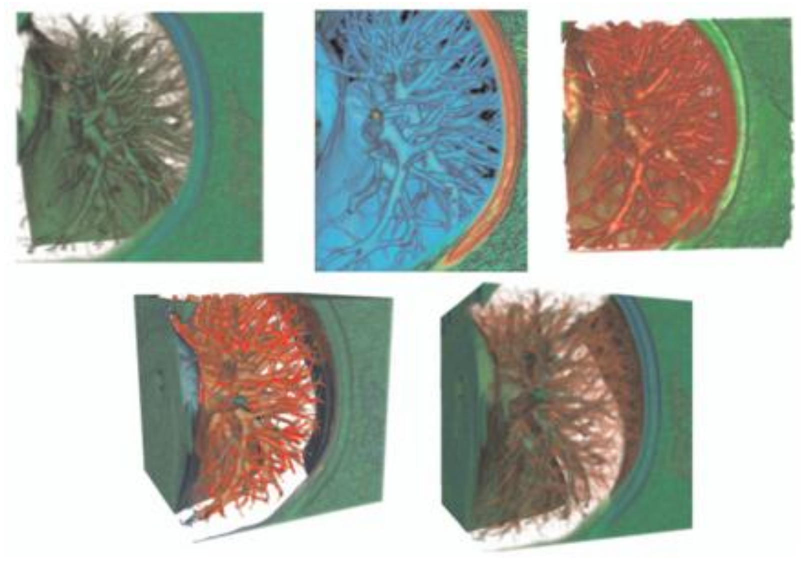


Figure 10: Top row: images produces using a traditional 1D TF in VTK, a 2D lighting and gradient based TF and a multi-dimensional TF respectively. Bottom row: images generated using texture-based TF approach [20].

Another team of researches attempt to address the problem of feature separation in [21] and [22], with the latter offering a semi-automatic selection of *neighborhoods* for extraction of statistical properties (Figure 11). They also demonstrate a novel approach of integrating statistical information into the design process of a TF, which makes selection of the features of interest much more intuitive. However, this technique also suffers from very high memory consumption in the pre-processing steps (triple that of the original data) and poor performance for some types of noise present in the data (when the noise distribution is very different from an assumed Gaussian white noise).

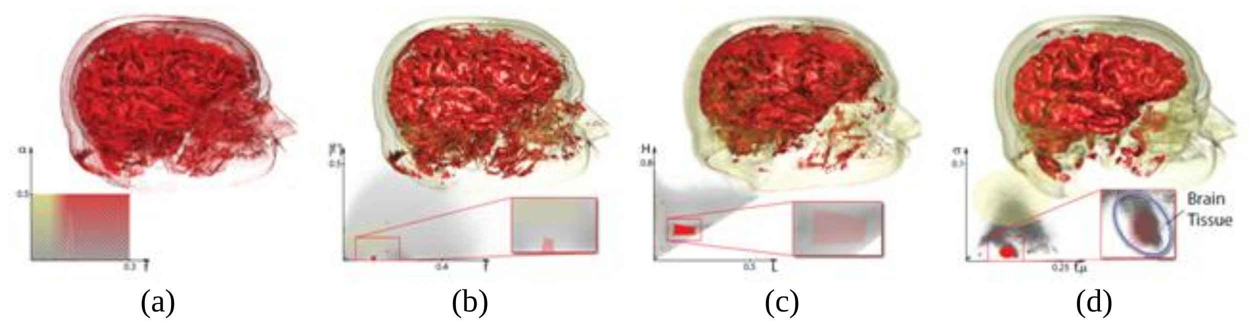


Figure 11: Comparison of (a) 1D and (b) 2D TFs and (c) an LH histogram with (d) a statistical TF [22]. All TF types are used to try to classify the same brain tissue.

2.1.5 Semantics-based Transfer Functions

Approaches in this category try to address the issue of the complexity of adjusting parameters of TFs and reduce the unpredictability of the visual results associated with this process. In [23] the authors create a semantic model which maps various semantic parameters (e.g. *sharpness*, *visibility*, *contrast*, etc.) to instances of TFs based on user requirements (Figure 12). This in turn allows non-expert users to intuitively visualize volumetric data without technical knowledge of the visual parameters involved in the visualization process. The algorithm is centered around the notion of a so called *principal component analysis* (PCA), which is used to determine the vector of maximum TF variance in its given parameter space.

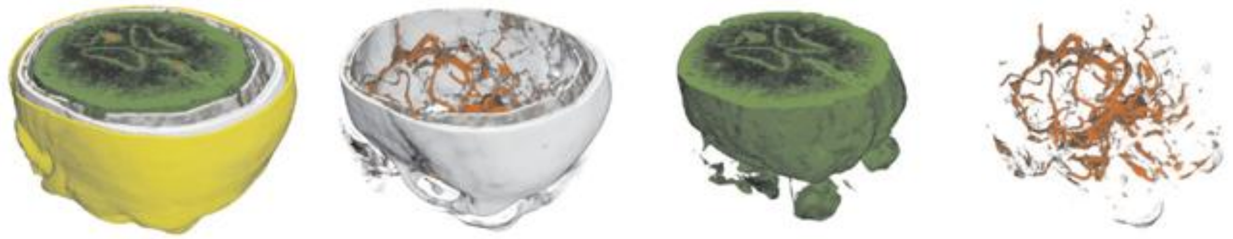


Figure 12: Visibility of different structures within a volume is controlled by the user by using a single slider to switch between instances of TFs [23].

In [24] additional derived quantities for evaluating the joint information of multiple modalities are provided, which further enhance the possibilities of separating different tissues while providing for easier understanding of a TF space (Figure 13).

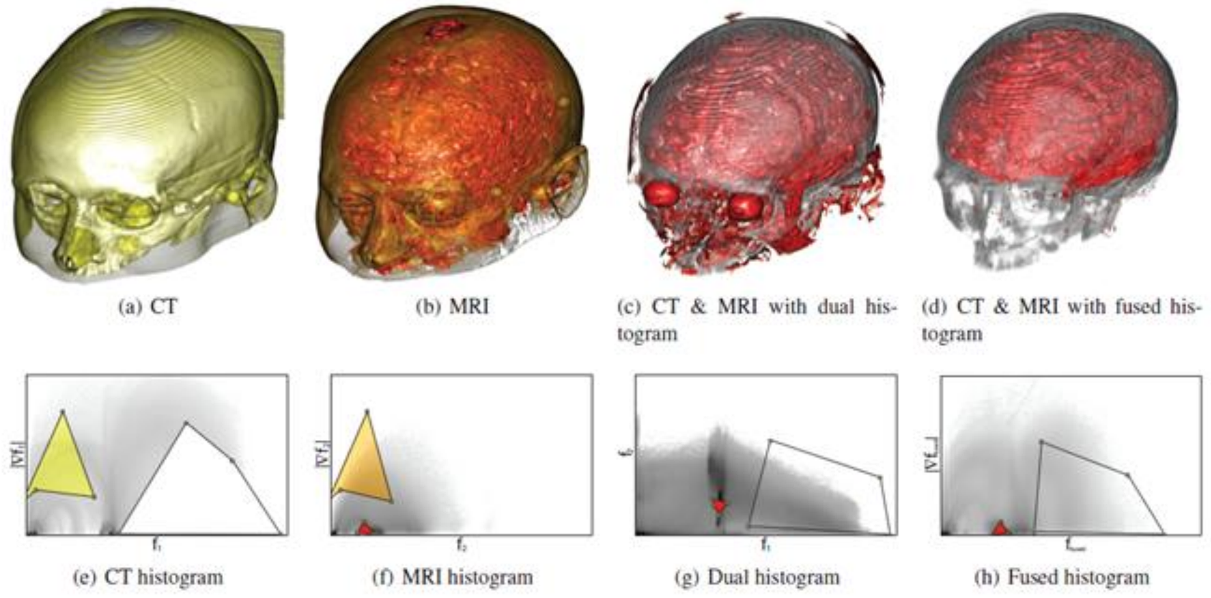


Figure 13: Comparison of single modal visualizations (two imaged on the left) to multimodal visualization (two images on the right) [24].

2.1.6 Clustering-based Transfer Functions

The idea behind these methods is to combine or *cluster* certain features of a volume based on pre-defined criteria. In [25] features are clustered in accordance with volume intensity values, gradient magnitude, second directional derivative and neighboring values. The proposed solution allows the user to not only specify the number of clusters to be used, but also to combine or split them when there are too many or too few of them. To achieve interactive frame-rates the algorithms are only applied to subsets of randomly chosen values within a volume (the remaining values are classified based on the minimum distance between the feature value and mean vectors of clusters). The results of this pre-processing step are then converted into a 2D TF based on density values and gradient magnitudes.

In [26] the authors build on their previous work that dealt with LH histograms. They propose a method that generates clusters by analyzing the *LH feature space* and evaluating the spatial connectivity of the clusters. Filtering is applied to eliminate false positives, which usually occur due to noise present in the data. Different criteria (such as *distance*, *separation* and *direction of a cluster elongation*) are then used to group the clusters together. Using *Bayesian decision theory* the clusters are interpreted as bivariate 2D *Probability Density Functions* (PDF), which allows plotting them in a form that is easy for the user to work with (Figure 14). This method allows for a real-time interaction with the cluster hierarchy and also provides the ability to adjust the grouping of the clusters on the fly, thus creating different visual results.

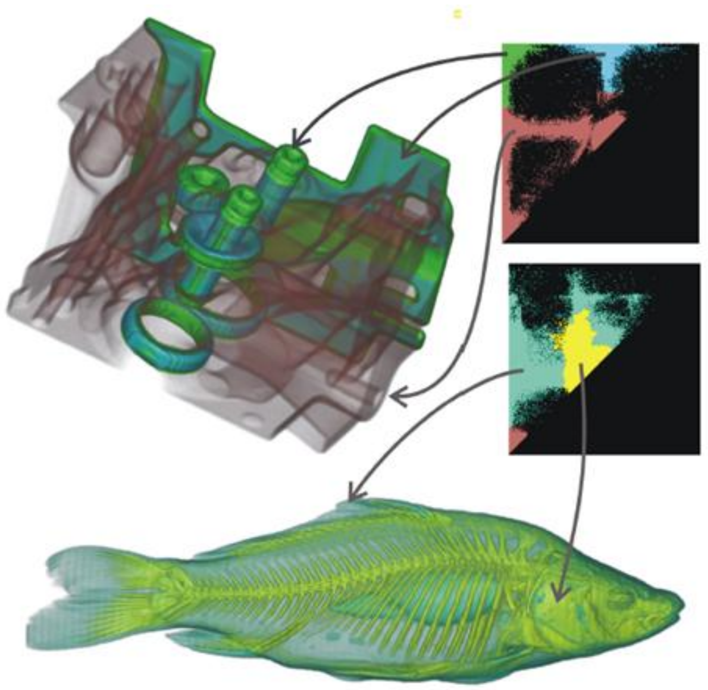


Figure 14: Visualizations of the engine and the carp are produced by painting over LH clustered histograms [26].

2.1.7 Non-Transfer Function-based Mapping Methods

This set of visualization techniques is based on the idea of mapping volume intensity values and their derivatives directly to color/opacity pixel intensities. The simplest and most commonly used approach is called *Maximum Intensity Projection* (MIP). MIP works by finding a global maximum for every ray traversing a volume (Section 3.2) and linearly mapping these values to a visual range. The fact that only a single value for each ray is mapped to a visual field means that MIP does not provide any depth cues, which can lead to ambiguities while interpreting the resulting image. Several approaches addressing this drawback have been proposed. In the *Depth-Shaded Maximum Intensity Projection* (DSMIP) [27] technique the intensity of each sampled value within a volume is modulated by its depth and the results are written to the final image only if the value of the currently processed pixel is smaller than the newly modulated value

(Figure 15(a)). This in turn means that any values located far away from the current viewing point have less of a chance of being projected onto the final image, which can result in obscuring of some of the volumetric features, even if they have high intensity values. In [28] a *Local Maximum Intensity Projection* (LMIP) technique is introduced, which allows the user to interactively control the threshold parameter. The values are evaluated in a front-to-back order along the travelling path of a ray cast into a volume, and the first found local maximum above the user-defined threshold is projected onto the final image (Figure 15(b)). In case no such value is found, a projection of the global maximum is used instead. This approach provides more flexibility by giving the user a degree of control over the visualization process, but also suffers from occlusion problems, since it does not support translucent materials.

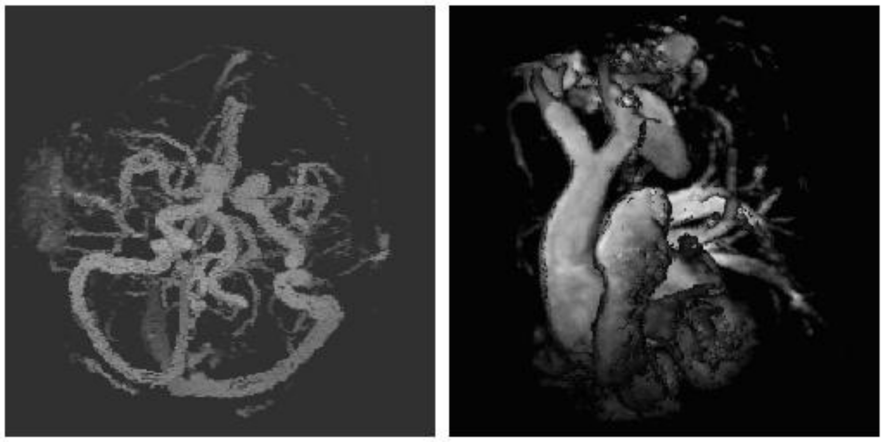

(a) (b)

Figure 15: Images generated using (a) DSMIP [27] and (b) LMIP [28] visualization techniques.

Recently an approach presented in [5] attempts to combine characteristics of both DVR and MIP by modifying a monotonically growing opacity function typically associated with DVR. This is done by modulating any previously accumulated color/opacity results along the path of a ray by the amount of the positive difference between the newly evaluated value and the previous one. In other words the intensity of any previously accumulated results is reduced by the difference between the current and the previous value, but only if the current value is larger than the previous one. It also allows for a smooth transition between DVR and MIP in terms of visual results by adjusting a parameter that controls the level of modulation of any previously accumulated results (Figure 16). This allows the user to choose the best possible representation of the data in the resulting images. The algorithm presented by the authors has been chosen and implemented as the default visualization technique to be used in conjunction with the painting metaphor presented in this thesis, and will be described in detail in Section 3.3.4.

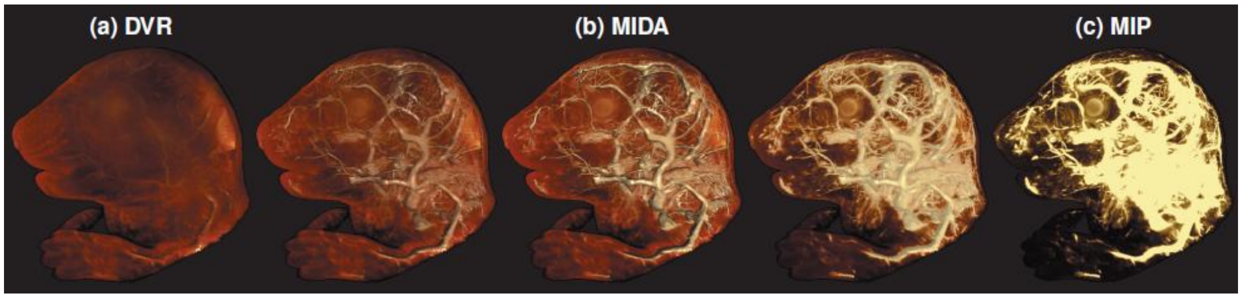


Figure 16: Seamless transition between DVR, MIDA and MIP [5].

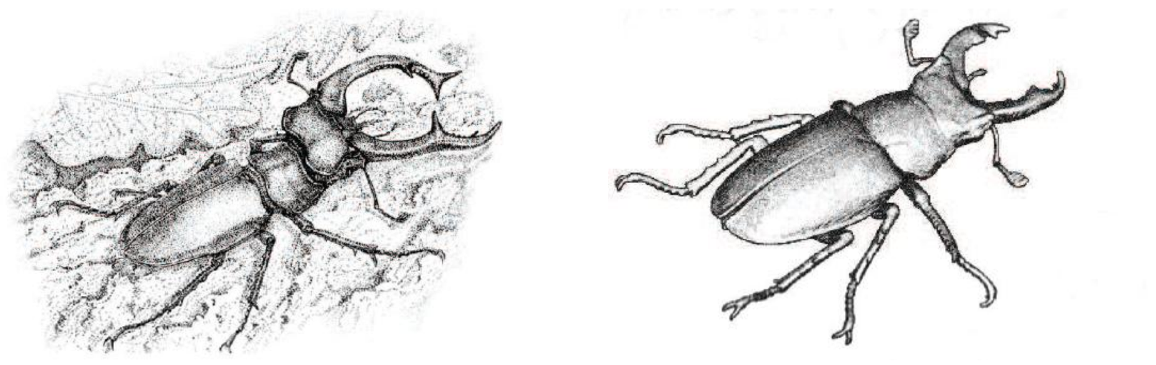
2.2 Illustrative Volume Visualization

For many years one of the prevailing objectives among researchers and developers involved in the field of VR was to achieve an ever higher degree of visual realism of generated images (by using realistic lighting, shading, refractions, etc. [6]). However as the field evolved and matured it became clear that often a more stylized and sometimes even cartoon-like rendition of the volumetric data can be more beneficial in terms of both highlighting the features of interest and ease of information comprehension. Thus a branch of VR that deals with illustrative visualization techniques was born. The following material will present some of the most prominent works in this field while separating them into two major categories based on the classification proposed in [29].

Depending on the kinds of modification they perform and the scope of volumetric features they work on, all of the VR illustrative visualization techniques can be broken down into *low-level* and *high-level* abstraction techniques.

2.2.1 Low-Level Abstraction Techniques

Low-level abstraction techniques change the way volumetric objects are visualized by highlighting important features or de-emphasizing less relevant ones. This is achieved by presenting the information in a more stylized way (e.g. line drawings), while drawing attention to the features of interest by using more prominent illustrative visualization techniques (such as contouring, shading, haloing, etc.) (Figure 17).



(a) (b)

Figure 17: (a) A pencil hand drawing of a staghorn beetle and (b) a VR generated image of a staghorn beetle using a similar illustrative style [29]. In the VR generated image the shape of the various parts of the beetle is emphasized by using a contouring technique [29].

2.2.1.1 Stylized Shading

This is a family of techniques that enhances features of importance while deemphasizing background information using non-photorealistic shading techniques. This can be done in a number of ways. One of the approaches is to modify how lighting in the vicinity of the features of interest is applied during the TF value mapping process to achieve the desired artistic effect. Some of the more popular methods in this category include *Lighting Maps* [30], which are basically 2D functions representing light interaction with the surface of an object based on the direction of light and surface normal vectors[11] of a visualized object, and a so called *Lit Sphere Shading* [30] approach. In the latter case the idea is to capture all of the possible color variations of an object as a function of a surface normal direction. This is done by using the concept of a *sphere*, where an image of a sphere is used to capture all possible variations of normal directions of an object relative to the current viewing direction (normal vectors parallel to the viewing direction are mapped to the center of the sphere and normal vectors orthogonal to the viewing direction are mapped to the outside rim of the sphere) (Figure 18). Thus the sphere image is indexed by the normal values of a volumetric object and is used to map the normal values to the TF values during the shading process.

[11] In computer graphics normal vectors are used to define the direction in which the surface of an object at any given point is facing.

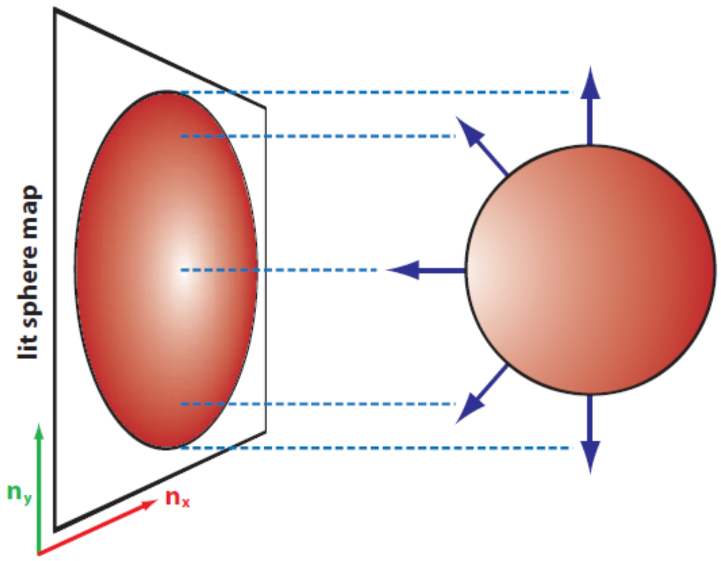


Figure 18: Lit sphere shading [30]. A 3D sphere represents all possible normal direction variations for any visualized object. The actual normal directions of the currently visualized object are encoded in the form of a sphere image. During the shading process the $n(x, y)$ points (corresponding to the currently evaluated normal value) on the sphere image are mapped to the TF values.

2.2.1.2 Style Transfer Functions

This approach extends the domain of TFs, that are normally used to map scalar values to color and opacity values only, to include shading information as well (i.e. normal values are now mapped too). One example of this approach was presented in the already mentioned paper [16], where surface curvature information is used to create non-photorealistic renderings with emphasized silhouettes. Another example is the work presented in [30] where a spherical model for normal-based light attributes mapping is combined with multi-dimensional style TFs (Figure 19).

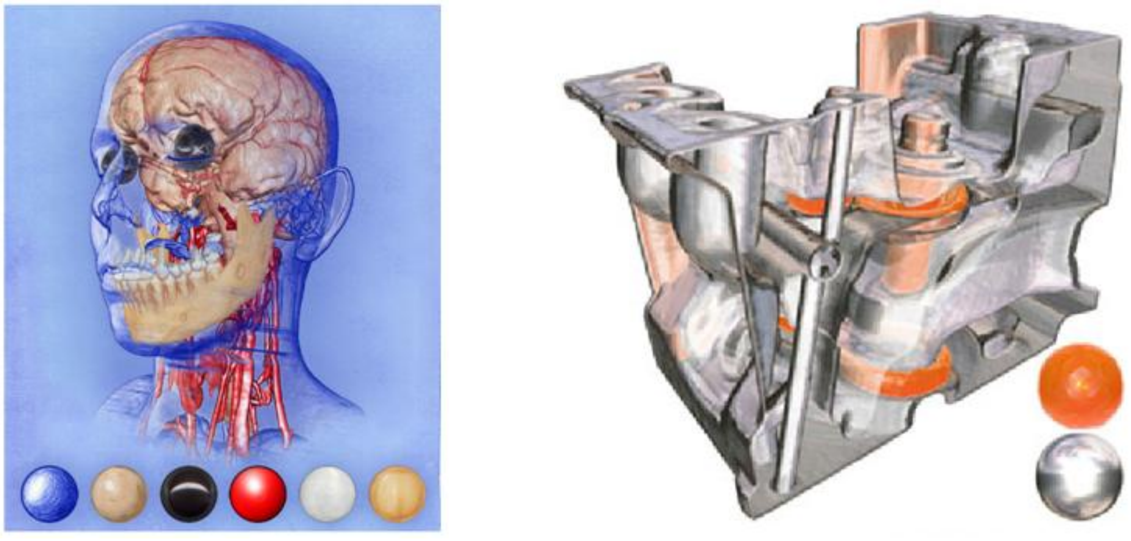

(a) (b)

Figure 19: Visualizations using (a) a multi-dimensional style TF and (b) contouring [30].

2.2.1.3 Stylized Contouring and Transparency

Contouring often helps to resolve the ambiguities when visualizing semi-transparent objects. This is done by emphasizing the transition between front- and back-facing surfaces. A traditional technique most often employed in VR, that uses a dot product[12] of the viewing vector and a surface normal, has a drawback of producing contours of unpredictable thickness. An improved method proposed in [16] regulates the contours thickness by restricting its value based on the viewing direction. A somewhat modified version of this method is presented in [30] (Figure 20(a)).

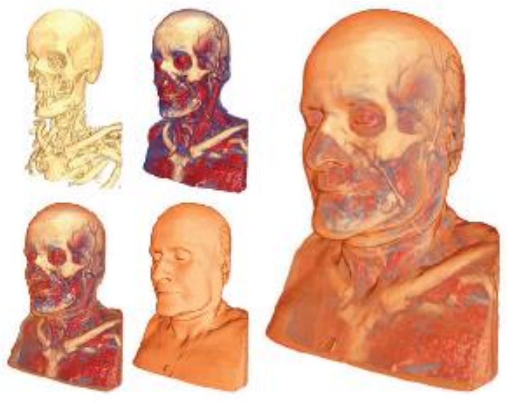
All of the previously mentioned illustrative visualization techniques can be enhanced by adding an option to make certain features semi- or completely transparent. A normal direction can also be brought into the equation to allow for view-dependent opacity effects [30] (Figure 20(a)).

[12] In linear algebra dot product of two vectors is a sum of products of their corresponding components. In computer graphics a dot product is used to determine how "similar" two vectors are (the larger the dot product the more similar the vectors are, i.e. they point in a similar direction).

2.2.1.4 Volumetric Halos

This approach can be helpful when there are a lot of fine and overlapping features in a volumetric dataset. Since human eyes are very sensitive to sharp changes in contrast, halos drawn around objects can be helpful in providing additional visual cues. Two approaches to generating haloes include a technique where halos are generated in a pre-processing step and thus become an integral part of the volume, and methods where halos are generated on the fly during the actual visualization process. Naturally the latter method allows for more flexibility in terms of interactivity and achievable visual results.

In addition, halos can be generated in a directional or omni-directional way, that is they can be visualized while taking the viewing or lighting direction into account, or just extend uniformly in all directions around the object of interest [30] (Figure 20(b)).


(a)

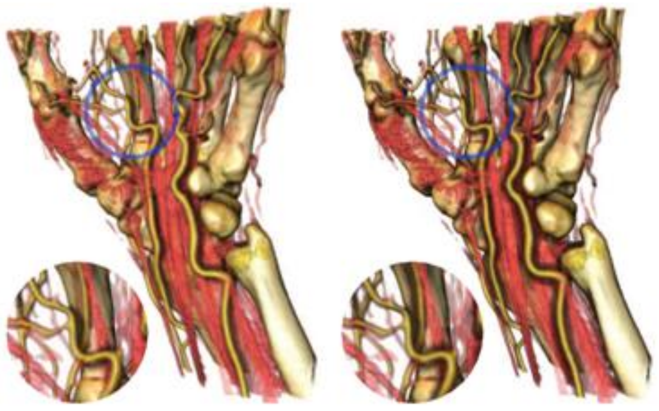

(b)

Figure 20: (a) Examples of using the view direction dependant stylized transparency and contouring techniques [30]. (b) Visualizations using the directional and omni-directional haloing effect respectively [30].

2.2.2 High-Level Abstraction Techniques

In contrast to low-level abstraction techniques, that affect the appearance of all volumetric features in the scene, high-level abstraction techniques try to highlight only certain features in the resulting image by using various techniques borrowed from the field of the traditional medical illustration (e.g. making certain tissues in the illustration transparent, using exploded views, etc.) (Figure 21).

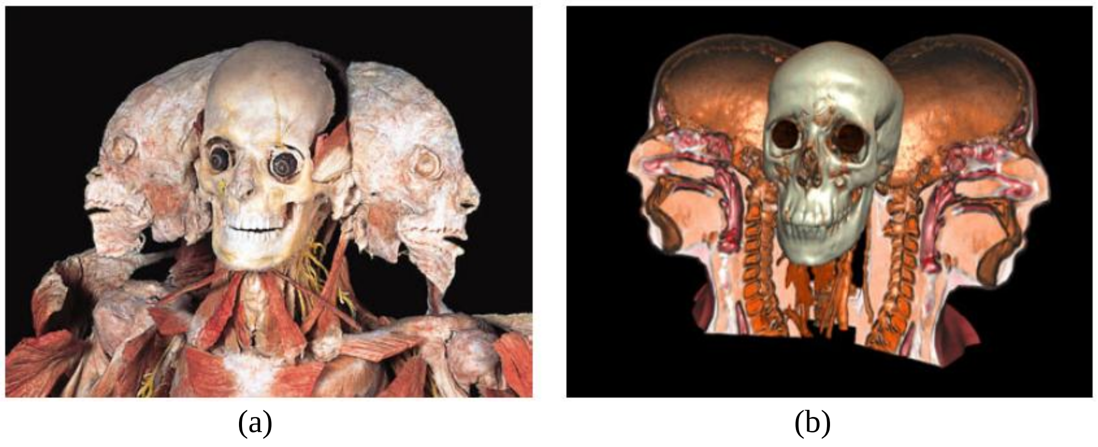


Figure 21: A photograph of a plastinated anatomic model from Gunter von Hagens' *Body-worlds* exhibition (a) and a similar exploded-view image generated using VR (b) [29].

2.2.2.1 Context Preserving Views

One of the most popular illustration techniques is to use transparency to show the interior of the objects while preserving context by reducing transparency closer to their edges [31]. This technique is employed in [32], where the authors use the results of a shading intensity function for opacity modulation (Figure 22). The basic idea is to reduce the opacity for large regions of highly illuminated material that would normally correspond to rather flat surfaces (e.g. skin) while keeping the less brightly lit regions (for example light silhouettes of veins and tendons) visible. Also the distance to the current viewing point is taken into account to reduce opacity attenuation for rays that have already accumulated a lot of opacity and thus limit the number of overlapping transparent objects (as too many of these can actually make perception more difficult).

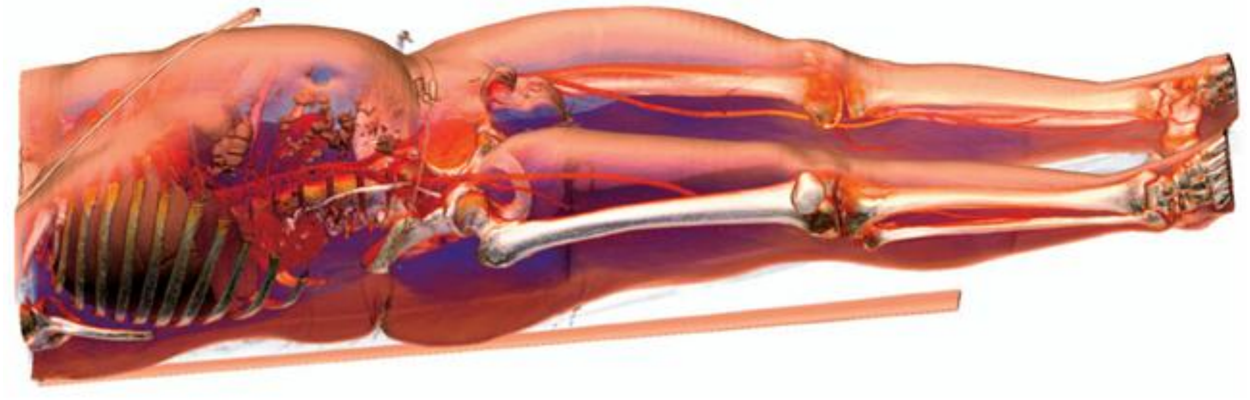


Figure 22: The context in the image is achieved by making the parts of the body that occlude the features of interest transparent, while reducing transparency for the rest of the body [32].

In another paper [33] the authors present a novel VR approach, also borrowing from the field of traditional illustrative techniques, in which they use compositing and masking techniques to draw the user's attention to certain parts of the image while preserving the overall context. The final image is generated from several independently produced layers (such as the line drawing layer corresponding to the contours of the organs and several tissue layers that are generated using isosurface, DVR and even MIP rendering techniques) that are combined in the final stage using a masking operation (Figure 23). Using masking also allows achieving the effect known as a *Magic Lens* (i.e. a method where a user guides some kind of a virtual optical device, such as a flash light or a lens, to highlight internal structures in the certain region of an object) which is another popular and widely used context preserving technique.

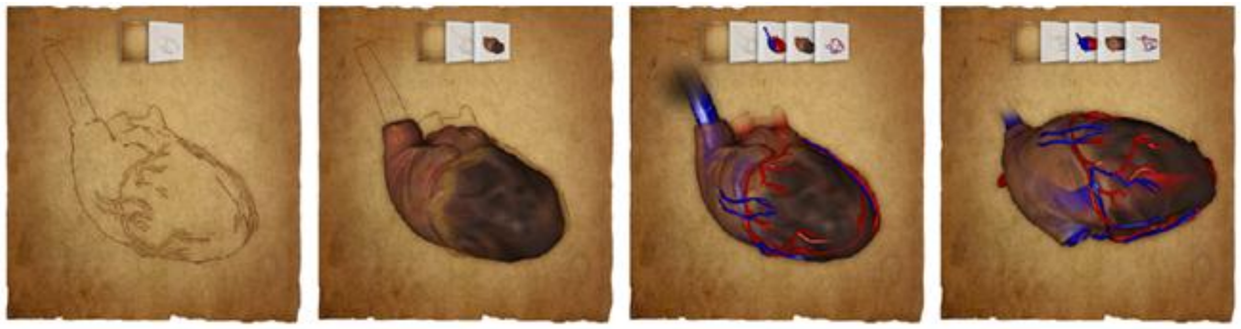


Figure 23: An illustration of the human heart as a combination of three layers (three images on the left: line drawing, DVR muscle layer, isosurface veins layer) with a rotation operation applied (right) [33].

Different variations based on the idea of a magic lens have been proposed by researchers. In the case of DVR a second TF [8] can be used to visualize an ROI in a distinct manner (Figure 24(a)). In [33] multiple layers are first generated using different visualization techniques and then combined using masking in the user defined order (Figure 24(b)). Finally, various standard and custom magnification effects are employed in [34] to enlarge the features of interest within the context of the surrounding volume (Figure 24(c)).

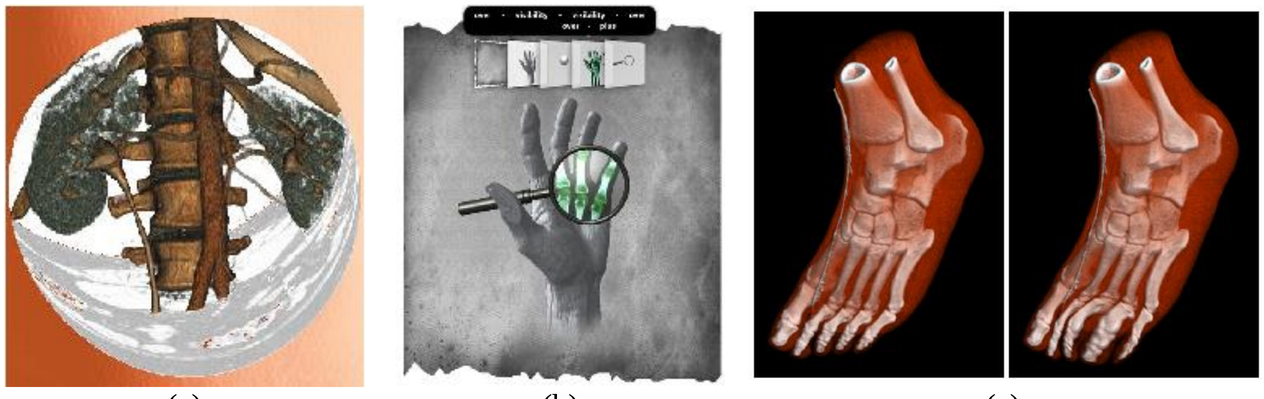

(a) (b) (c)

Figure 24: (a) Magic Lantern visualization using two different TFs [8]. (b) Magic Lens using masking and compositing technique [33]. (c) Magic Volume Lens using a magnification effect [34].

Another approach presented in [35] enhances the feature of interest in the resulting image by super-sampling intensity values of the volume using *B-spline subdivision* and *fast gradient quantization*, which allows for an instant "*zoom in*" effect on the said feature (Figure 25).

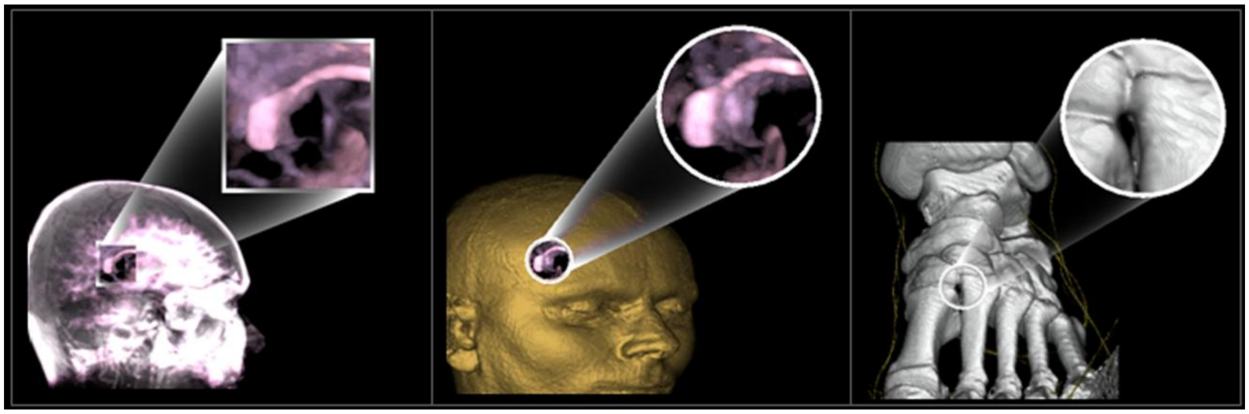


Figure 25: "Zoom in" effect in action. The context is provided by using rectangular and circular ROIs, that are super-sampled using B-spline subdivision and enlarged [35].

A recent technique presented in [36] combines several mathematical and visualization models to allow for a real-time 3D context-preserving exploration of large volumetric datasets. Among them are *octree*[13] based *out-of-core*[14] data management, MIDA based VR for parts of the volume that correspond to an ROI, *distance-based opacity modulated* DVR for regions outside of an ROI, and *superquadratic*[15] 3D envelope shapes (such as a cylinder, a sphere and a rounded cube) for ROI definition (Figure 26).

[13] An octree is a tree data structure in which each parent node has eight child nodes. In computer graphics it is often used to represent objects in a scene in a hierarchical order, allowing for more efficient granular data processing.
[14] In computer graphics out-of-core algorithms are designed to allow for processing of data that is too large to fit in dynamic memory (usually memory dedicated for the GPU usage only) by fetching or streaming the data in chunks according to the demands of the application.
[15] In computer graphics superquadratics or superquadrics are a family of 3D geometric shapes that are described by equations similar to the ones used to describe regular quadratic shapes, but with the squaring operation replaced by the arbitrary powers. By changing the powers in a superquadratic equation the range of shapes can be extended.

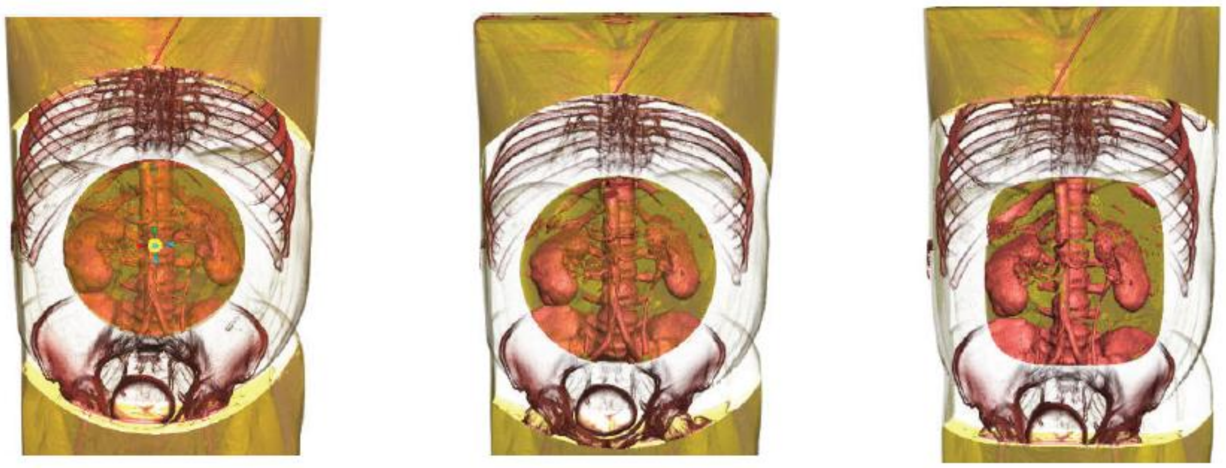


Figure 26: Images produced MIDA and distance-based opacity modulated DVR using different superquadratic envelope shapes [36].

2.2.2.2 Exploded Views

This is another group of methods aimed at solving the problem of occlusion in VR while providing contextual information. The idea is to decompose an object into several parts so that internal structures of interest can become visible. This approach has an additional benefit of being able to reveal additional information such as cross-sectional geometry of an object. In [37] an approach is proposed that is based on the idea of separating a volume into a selection and background parts (the distinction is based on a combination of the selection geometry, the data volume and the specified TF). The user can then specify into how many pieces and along which axes the background potion of the volume should be split. Notions of force and hinge joints are also used to specify how far apart separate pieces of the background should be spaced and along which axes they should be aligned and rotated (Figure 27).

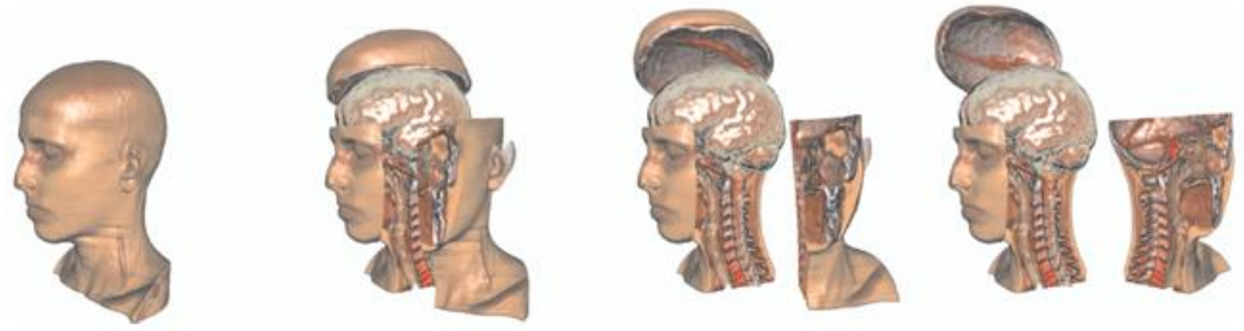


Figure 27: An interactive exploded-view illustration with two virtual hinges constraining the positioning of the exploded parts [37].

In [38] a volume is sliced into slabs along the user specified axis (Figure 28). However, in this approach the manual process of specifying the thickness of slabs is automated through a mechanism that determines the thickness by measuring the level of similarity between partitions. The similarity criterion is evaluated on the basis of *maximum gain of information*.

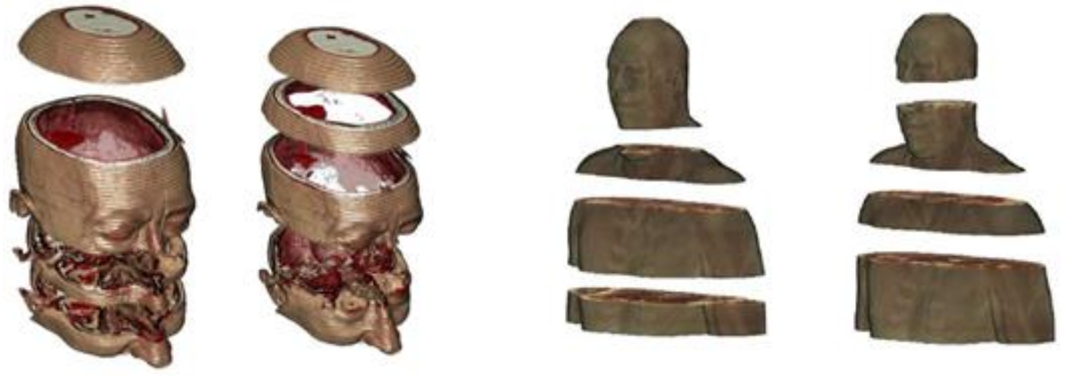


Figure 28: Exploded views with different portioning settings applied [38].

2.2.2.3 Volume Clipping and Sculpting

These methods attempt to reveal the structures inside a volume by physically removing parts of the volume that otherwise would obstruct them. This is achieved by applying *Boolean operations* to *cut* or *carve* away parts of the volume using clipping planes, boxes, spheres and other geometric shapes and in general include any technique that allows displaying a subset of a volumetric dataset bounded by an isosurface. The most prominent works in this field include [39, 40] and [41]. The first paper [39] presents an approach where various geometric primitives and depth test algorithms are used to specify parts of the volume that need to be cut away (Figure 29(a)). In the second approach [40] a mesh which can be flexibly deformed by the user with an adjustable sphere of influence is used (Figure 29(b)). The last paper [41] describes a whole framework called *VolumeShop* that allows for advanced manipulation and illustrative rendering techniques to generate illustrations directly from volumetric data. The principle behind its operations is a multi-volume representation of the scene with ability to individually control each volume. The segmentation of the volumetric data is performed via the means of *surface-constrained 3D painting* (Figure 29(c)). The system also includes segmentation and labeling features that allow for an annotated representation of the data in the resulting images.

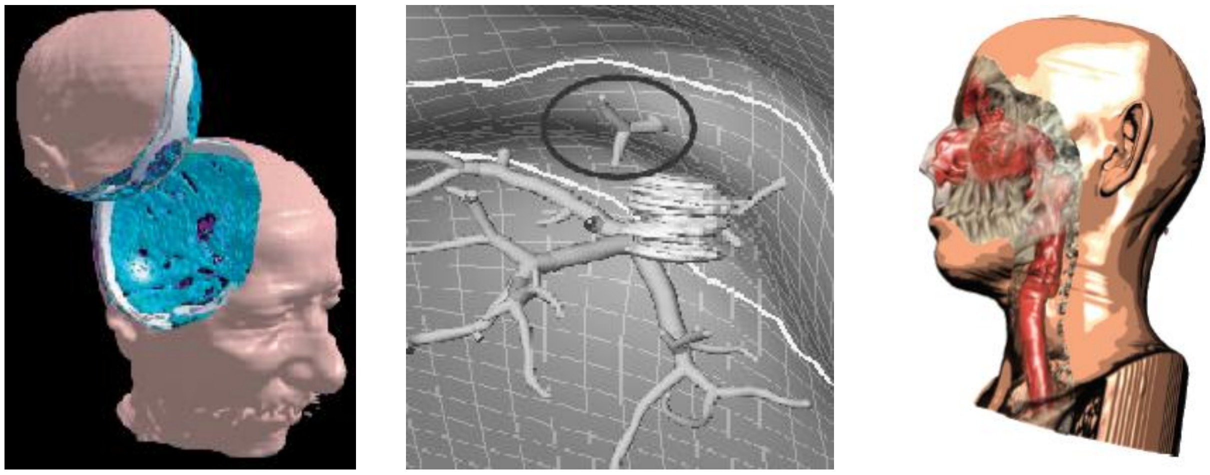

(a) (b) (c)

Figure 29: (a) An example of the spherical carving operation of the volume [39]. (b) A geometry based 3D deformable mesh is used to define the ROI and exclude all other parts from the visualization process [40]. (c) A section of the skin layer is cut-away after painting on the surface of the head and neck (c) [41].

Volume clipping can also be used as a pre-processing step, where the occluding parts of the volume are removed in order to reveal the structures that will be later used to perform other operations on (e.g. segmentation). This approach is used in [42] where the authors use the surface-based sketching operation to mark the region on the surface of the volumetric object that will be clipped or *peeled* away. Then the *Point Radiation* algorithm is used to project rays into the volume within the defined region. The orientation of the rays and the depth of penetration of the surface depend on the current orientation of the masking plane and the pressure level of the input stylus device controlled by the user (the maximum depth of penetration is limited to 12 voxels). Any parts of the volume intersected by the rays are removed to reveal the underlying structures (Figure 30).

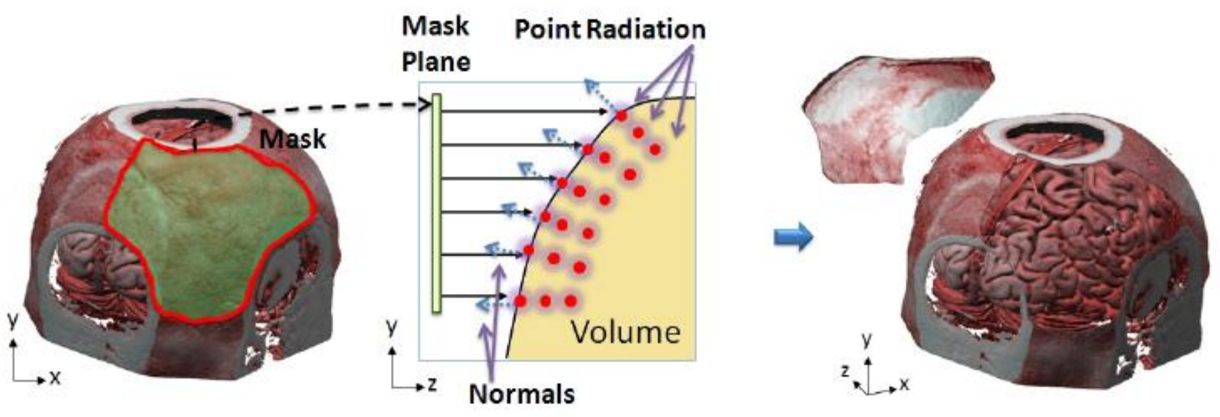
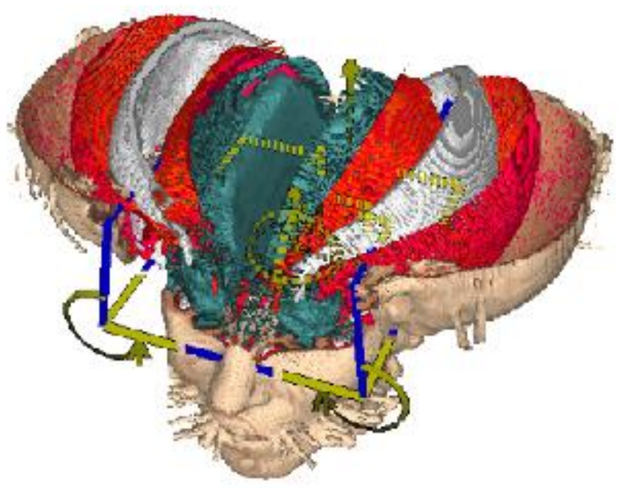


Figure 30: A free-from mask is painted on the surface on the skull. The rays are cast through the mask and any intersected parts of the volume are peeled away [42].

2.2.2.4 Volume Deformation

While making inner structures more visible, volume clipping methods may also result in the loss of context. To address this problem, another approach can be used which merely deforms volumetric data (without actually removing any portions of it) in such a way as to allow the structures inside the volume to become exposed. Operations employed in these methods are usually mimicking actions from real medical practices (e.g. tissue peeling, incisions, various tissue spreading techniques, etc). Two examples of this approach are presented in [43] (Figure 31(a)) and [44] (Figure 31(b)).


(a)

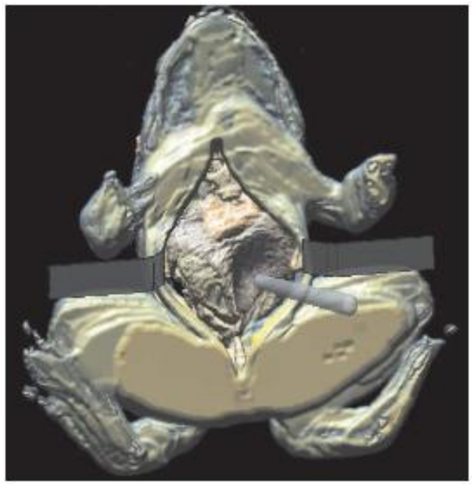

(b)

Figure 31: (a) The "*leafing*" technique in action [43] and (b) the "*pliers*" operation applied to the internal organs of the frog while the skin is being retracted [44].

A more recent paper [45] proposes techniques that allow for more advanced deformations (such as *retratcing*, *multi-spacing*, *highlighting* and *deemphasizing* deformations and others) with an emphasis on illustrative representation of the data (Figure 32). These techniques allow for an extension of the deformation methods into other areas of VR beyond medical applications (such as physics, mechanics and basically any other type of volumetric information visualization).

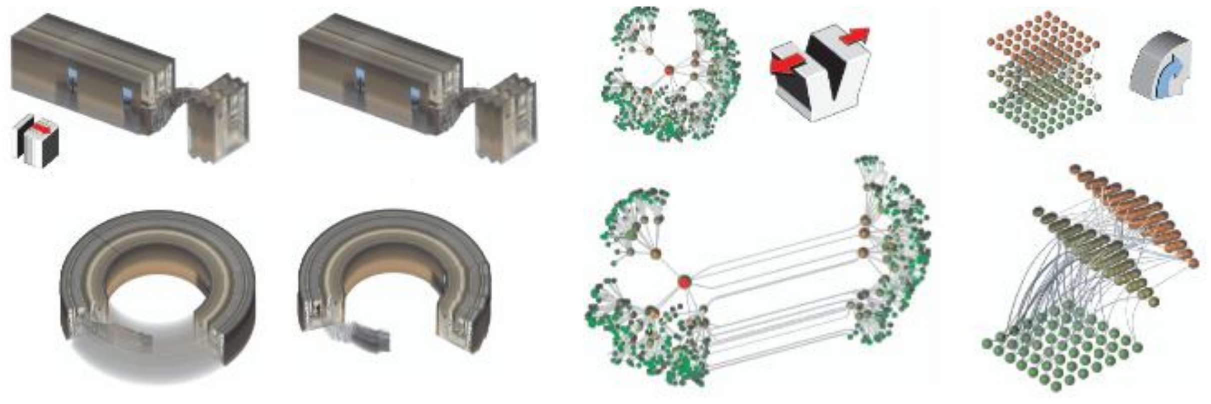

(a) (b)

Figure 32: An example of (a) a retracting deformation and (b) a bending deformation [45].

One of the most recent approaches that deals with deformations in medical datasets is presented in [46]. In it the authors introduce a notion of *constrained displacement fields*, which allow restrictions to be placed in terms of both what parts of the volume are allowed to be deformed and in which ways (by specifying rigidity and degrees of freedom). This avoids such common problems in unrestrained models as self-intersection and collision with features of interest.

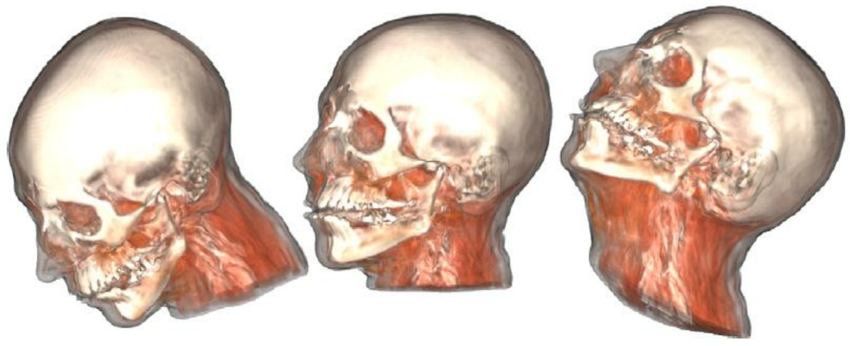


Figure 33: A simulation of a whiplash injury using constrained deformations [46].

2.3 Volume Space Navigation

In order to get a better understanding of any given visualized volumetric dataset it is necessary to provide the user with a means of navigating in a volume space. Furthermore any employed navigational model should be intuitive to use and provide the necessary level of precision.

According to [47] a generalized virtual navigational task can be thought of as consisting of both explorative navigation and directed navigation.

2.3.1.1 Explorative Navigation

In the course of explorative navigation the user is interactively inspecting the data to gather knowledge while being unaided and usually unrestricted in ways of navigation. Most navigational models of this type are based on the concept of a so called *Virtual Trackball*, *Sphere* or *Arcball* [48] and are usually combined with such operations as panning, rotating and zooming to allow for adjustment of an arcball's center of rotation. Examples of this model can be found in numerous 3D modelling suites [49] and volumetric visualization software [2, 4, 14, 15, 50, 51], and have been adapted to both trackball and mouse usage.

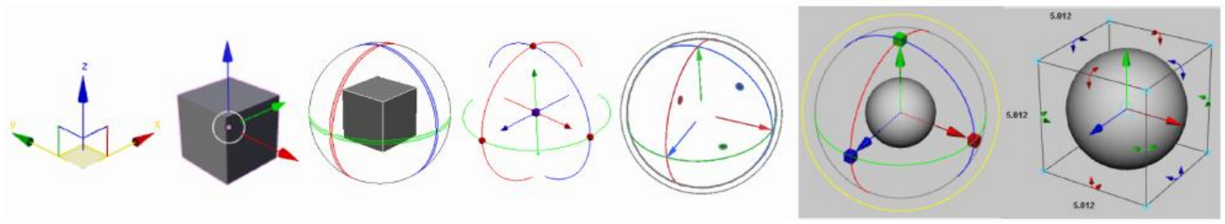


Figure 34: Various rotational and translational widgets from 3DS Max, Blender, XSI, Houdini, Mode and Maya [49]. All share the same underlying principle.

2.3.1.2 Directed Navigation

When it comes to directed navigation, very little research has been done for the field of volume graphics. The vast majority of techniques proposed for directed volumetric navigation are tailored toward specific tasks of virtual endoscopy [52-55] (i.e. navigation in heavily constrained tubular structures) and often include pre-processing steps with pre-defined visualization settings in order to be able to compute such parameters as a centerline or allowed camera travelling path (Figure 35).

Unlike well established navigational models developed for polygonal virtual scenes [56], where geometrical structures are always explicitly defined, visual appearance of features as well as their geometric form in a volumetric scene can be easily changed by adjusting TF parameters, performing carving or peeling operations, or using different visualization modes. Only recently a generic approach that integrates real-time navigation of both the explorative and navigational nature has been published [57]. In it the authors propose and implement a context-aware volume visualization technique in which a 360 degree spherical representation of the scene is updated in real-time and is used to perform instant collision detection under any combination of the visualization parameters (Figure 36). Combined with traditional operations of panning, rotating and zooming this approach allows for seamless transitioning between both navigation modes.

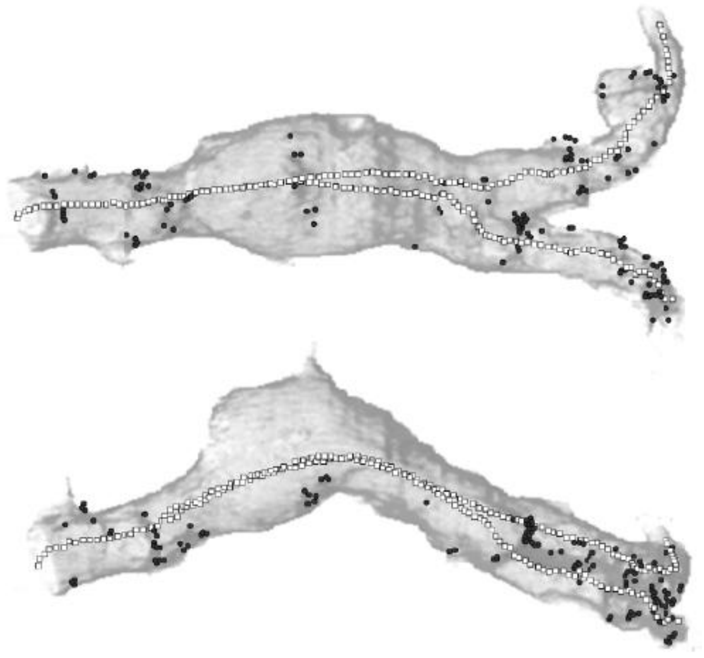


Figure 35: The 3D shape of the interior surface of the blood vessels is used to constrain the possible camera travelling paths [53].

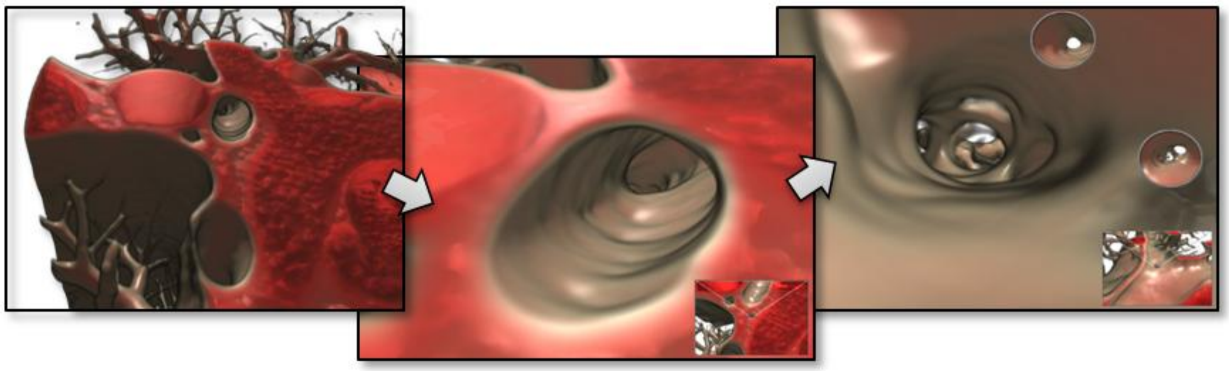


Figure 36: A series of screenshots taken during the volume navigation process using the context-aware volume navigation technique [57].

Chapter 3: Methodology and Implementation

The previous chapter introduced and classified various volume visualization and interaction techniques that aim at providing the user with tools for better volume exploration and understanding. Some of them employ a combinatorial approach by creatively drawing on the knowledge from different areas and fields in order to simplify that task. However, most still suffer from either:

- Assuming a potential user to have at least some background knowledge in the field of VR. The vast majority of the TF specification techniques described in Section 2.1 [1] suffer from this drawback, as they require an understanding of volumetric information representation in the form of various 2D histograms. Similarly, many high-level and low-level abstraction illustrative techniques [29] assume that the user is familiar with the different visualization models (i.e. isosurfaces, DVR, MIP, etc) and rely on their expertise in specifying the shading/lighting parameters, order and modes of combining different pre-rendered layers, orientation and thickness of the slices in the case of an exploded view data representation (in such a way as to achieve the best perception of the features of interest), etc.

- Providing models for interaction and visualization parameter specification that are still not easy to use. Again, many TF specification models are far from ideal in this respect [1]. Operating in a 2D feature space of a histogram is an unintuitive process that can lead to drastically different visual results with only minute changes in the positioning and shape of the TF-defining curves and geometrical shapes. Volume navigation and interaction models are often flawed [52-55] by the inability to easily position a virtual camera in the desired location with the desired orientation in the scene (this is an inherent problem of any navigational model that tries to use a 2D input device, such as a mouse or a trackball, to navigate in a 3D space). A confusing spatial relationship of various objects in the scene is another common problem that stems from the lack of sufficient depth cues or their poor implementation. Finally, depth specification [57] in general is not a straightforward task in the context of VR and its complexity is further compounded by the aforementioned issues.

The work presented in this thesis tries to overcome these domain knowledge and interface complexity issues by proposing and implementing a volume exploration model based on a painting metaphor that strives to be both intuitive and easy to use. This is achieved by focusing on providing the user with an integrated 3D single-view interaction model and a minimum of visualization controls that allow the user to concentrate more on their task of gaining insight into the data rather than encumbering them with the necessity to tweak numerous visualization and interaction related controls.

As a platform for the implementation of the extensions the project uses one of the most popular open source VR rendering software suites called ImageVis3D [2]. The following sections will explain the reasons behind choosing this particular platform, provide an introduction to some of the technical aspects of VR, and detail the user interaction model and underlying algorithms used to implement the proposed visualization and exploration model.

3.1 ImageVis3D as an Implementation Platform

The field of VR is maturing fast and today there are a number of very capable and feature-rich VR programs. Both open source and commercial products are available. Some of the most notable ones include OsiriX [15], Voreen [14], ImageVis3D [2], 3DSlicer [4], Amira [58] and MeVisLab [50].

For the implementation purposes of this thesis, a decision was made not to start "from scratch" but rather to take advantage of some of the already implemented auxiliary functionality (such as dataset loading and saving, graphics pipeline initialization, memory management, etc.) most of the aforementioned products could provide. A subset of open source projects was evaluated, consisting of Voreen, ImageVis3D and 3DSlicer, with ImageVis3D chosen as the final platform for extension. The decision was primarily based on the examination of the source code (ImageVis3D appeared to have the most well thought-out and clean structure) and the list of already implemented features that could potentially aid in the implementation of the envisioned future extensions (ImageVis3D provides a flexible out-of-core memory management system and an automatic *Level of Detail* (LOD)[16] support for any loaded dataset, both of which allow it to perform well even in the absence of sufficient memory or computational resources).

ImageVis3D is an open source volume rendering project, developed by the NIH/NCRR Center for Integrative Biomedical Computing (CIBC) and is written in pure C++ TR1 with Boost C++ extensions [59]. It provides wrappers for both DirectX (Microsoft Corporations' Proprietary Graphics Library) and OpenGL (***Open*** ***G****raphics* ***L****ibrary*) [9] and uses the Qt [60] cross-platform

[16] In computer graphics the term "Level of Detail" refers to the ability of an application to adaptively change the detail of displayed 3D objects according to the needs of the application or the user.

application framework developed by Nokia to implement its GUI controls and widgets. The fact that it does not rely on any other proprietary platform-dependent third party *Software Development Kits* (SDKs) and *Application Programming Interfaces* (APIs) and works with OpenGL, an open royalty free cross-platform standard, means that it can be easily ported to any hardware/software platform capable of compiling the code and supporting OpenGL 3.3. In fact a mobile version for iPhone called *ImageVis3D Mobile* is now available in Apple's iTune App store.

All of the extensions added in the course of this thesis work have been designed and implemented with the above unofficial portability development policy in mind, and thus also do not rely on any third party libraries (although some of the implemented algorithms necessitated using OpenGL 4.1 instead of 3.3).

3.2 Introduction to Volume Rendering

In the field of computer graphics *visualization* is the process by which data is represented through the means of images, diagrams and animations in such a way as to aid its understanding and cognition. Scientific visualization is a subfield that deals with visualization of data representing some sort of physically based phenomena (such as seismic activity, air currents or X-ray reflection rates measured by a CT scanner).

Volume Rendering (VR) is one type of scientific visualization technique that is used to generate 2D projections of discreetly sampled 3D datasets. It is inherently different from another more commonly used approach to represent 3D information, which is based on surface representation (knows as *geometry*- or *polygon*-based graphics).

In polygon-based graphics 3D objects are constructed from interconnected points (usually arranged in the form of triangular meshes). The points have various attributes describing their position in space, surface normal orientation, texture coordinates, etc. In order to view these 3D objects on a 2D screen a process known as *rendering* is used, during which the models are converted into pixels on the screen by applying spatial transformations, texturing, lighting, shading and other operations. This polygonal representation is quite efficient in terms of memory consumption and is suitable for highly parallel data processing algorithms (this explains the massively parallel architecture of modern day GPUs). However, in cases when fuzzy or complex data (e.g. smoke, clouds, water, various body tissues) has to be described, the polygon-based representation is far from ideal. Another obvious drawback is that in a polygonal representation no interior information about an object is represented.

On the other hand, in the case of a volumetric data representation, objects are stored in the form of a 3D array of cubic[17] elements, where each element occupies a curtain volume of space (Figure 1). These elements are called *voxels* (volumetric pixels) and have at least one scalar value (often referred to as *intensity* since it represents some sort of a physical property measured during the scanning process) associated with each of them. A collection of all these values is known as *a scalar field* of the volume. Thus volume visualization, also commonly referred to as *volume rendering*, can be defined as a process of displaying volume scalar fields [61] by projecting the interior information of a dataset onto a screen using one of several known techniques [62].

[17] This is the case when a regular grid is used and is the most common.

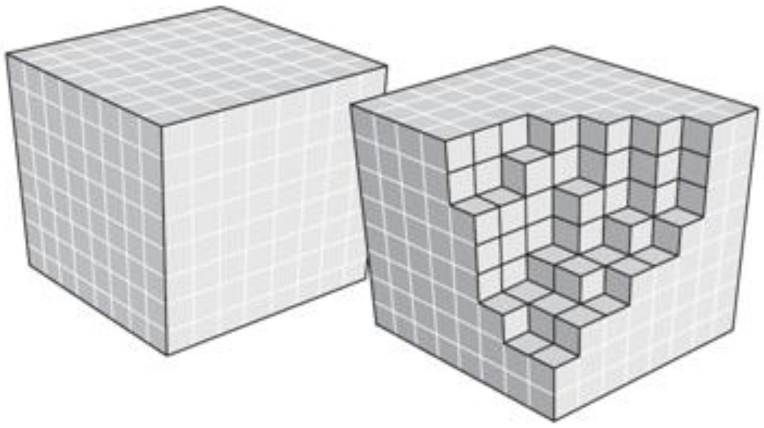


Figure 37: Voxels representing a discretized volumetric object [6].

3.2.1 Ray Casting

The advent of powerful GPUs has revolutionized the field of VR. Where in the past either the resultant image quality or performance or both had to be compromised due to the lack of sufficient computational resources, today highly parallel brute force rendering techniques dominate the field [63].

Ray casting [62] is by far the most popular brute force VR technique. It is a rendering method in which every voxel in a volumetric dataset is evaluated as part of a ray traversal process through the volume. For each pixel in the resulting image a ray is cast into a scene from the current view point (Figure 38). At certain intervals along the ray (equidistant in the simplest case) the volume dataset’s values are sampled, usually employing some kind of interpolation technique (tri-linear interpolation is often considered to provide a good balance between speed and quality). The resulting sample value is then used to determine the color/opacity (RGBA value) of that point in space using a lookup table, typically containing previously defined TF mappings. At this stage various lighting and shading techniques can be also applied in order to achieve the desired visual effects. The process continues by blending the obtained RGBA values until the ray exits the volume or the opacity of the desired level is achieved (called an *early ray termination* condition).

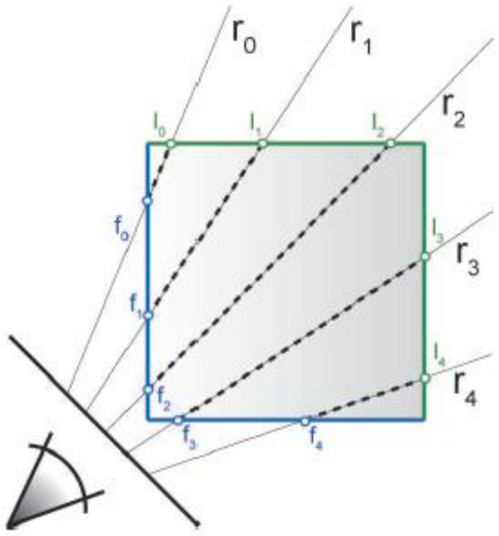


Figure 38: Rays cast through a volume are sampled between starting (f_n) and ending (l_n) positions [6].

The above *front-to-back* process of color and opacity accumulation can be described by following equation [6]:

$$\begin{cases} c_i^* = c_{i-1}^* + (1 + \alpha_{i-1}^*)\alpha_i c_i \\ \alpha_i^* = \alpha_{i-1}^* + (1 + \alpha_{i-1}^*)\alpha_i \end{cases} \tag{3.1}$$

where $c_i = (r_i, g_i, b_i)$ and α_i are the current color and opacity respectively at the sample point P_i, and c_i^* and $\alpha_i^*$are the accumulated color and opacity.

Because of this in-place evaluation of the visual model, ray casting is also often referred to as *Direct Volume Rendering* (DVR), though strictly speaking DVR is a broader term that includes all VR techniques dealing with direct voxel data evaluation during a rendering process.

3.3 Methodology and Implementation Details

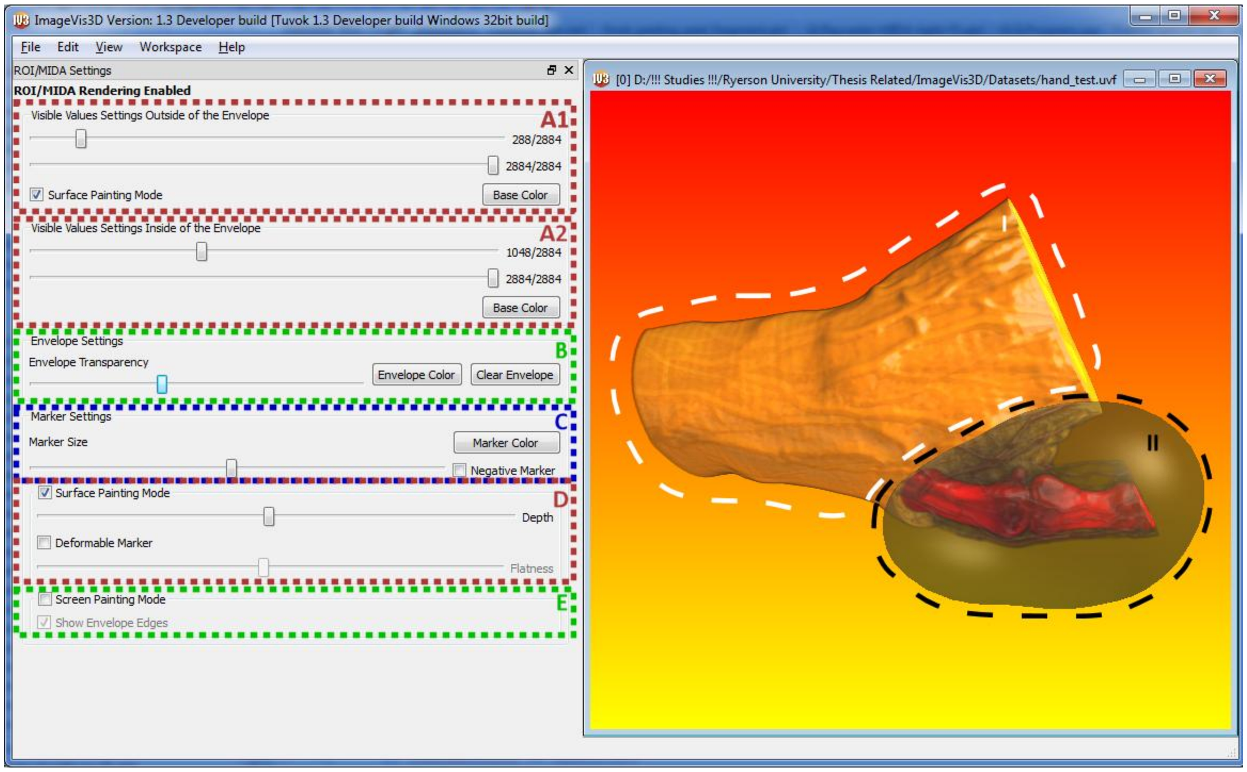
Since the proposed painting metaphor is a focal point of this thesis, it and the user interaction model that realizes it will be presented first, providing the necessary context for the following subsections.

3.3.1 User Interface and Camera Control

The user interface consists of two areas (Figure 39):

- The area containing the GUI components controlling the visualization parameters.
- The visualization area, where the user interaction with a virtual scene takes place and the results of the visualization are displayed.

Both areas are implemented as dockable *Multiple Document Interface* (MDI) child windows and thus can be freely resized and rearranged within the limits of the main application window according to user preferences.


ImageVis3D Version: 1.3 Developer build [Tuvok 1.3 Developer build Windows 32bit build]

[0] D:/!!! Studies !!!/Ryerson University/Thesis Related/ImageVis3D/Datasets/hand_test.uvf

ROI/MIDA Settings

ROI/MIDA Rendering Enabled

Visible Values Settings Outside of the Envelope: 288/2884, 2884/2884, ☑ Surface Painting Mode

Visible Values Settings Inside of the Envelope: 1048/2884, 2884/2884

Envelope Settings: Envelope Transparency

Marker Settings: Marker Size, ☐ Negative Marker

☑ Surface Painting Mode, Depth

☐ Deformable Marker, Flatness

☐ Screen Painting Mode

☑ Show Envelope Edges

Figure 39: User interface with the GUI controls (on the left) and the visualization area (on the right).

3.3.1.1 Camera Control

The visualization area is the area where the user interacts with the visualized volumetric dataset. In order to control the orientation and position of the camera relative to the volumetric object an arcball rotational model has been used. By clicking the left mouse button anywhere in the visualization area and holding it while moving the mouse, the user can rotate the camera around the object. The camera can also be moved in space so that the object appears closer or farther away by using the scroll wheel on the mouse. Finally, right clicking and holding while moving the mouse allows the scene to be panned in the desired direction.

3.3.2 Painting

The painting metaphor allows the user to define an ROI by placing “brush strokes” directly in the space of the visualization area. By placing additional brush strokes the paint is “spread”, thus creating the envelope that defines the ROI and encloses the features of interest. Painting can be done in either 2D or 3D. When the 2D painting mode is used the painting is done in the visualization area screen space coordinates. That means that the envelope always stays in the same position in terms of the screen coordinates regardless of the orientation of the visualized volumetric object[18] (Figure 40).

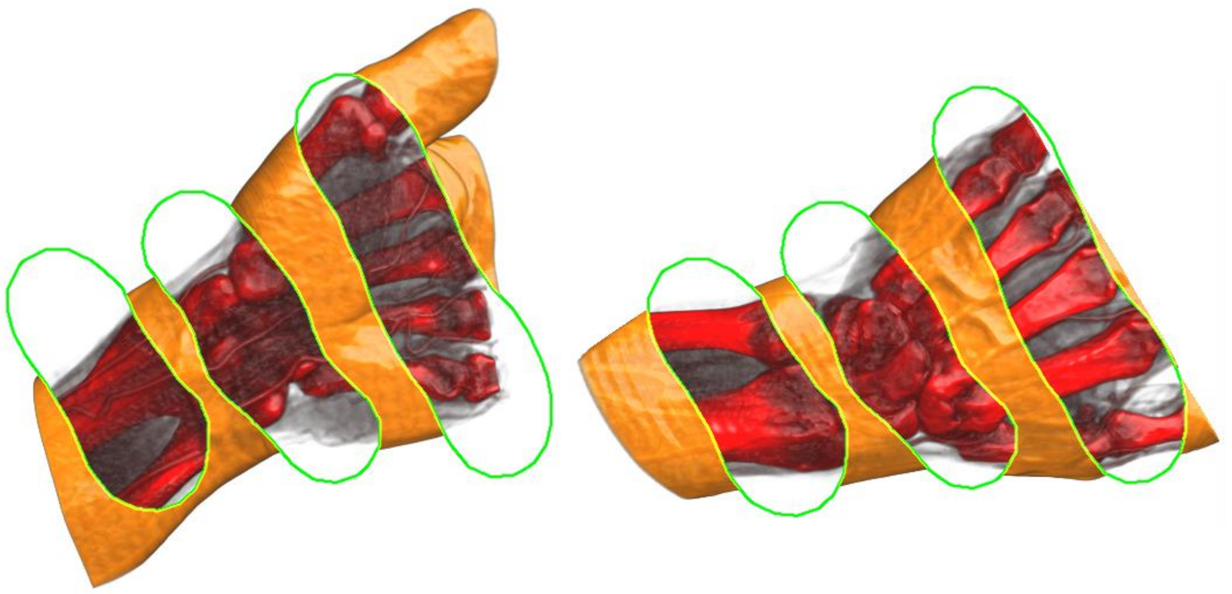


Figure 40: A 2D envelope defined in the space of the screen as three 2D stripes. The shape and position of the envelope stays the same regardless of the orientations of the visualized volumetric object.

[18] A 2D envelope acts in a manner similar an X-ray machine that projects its rays all the way through the scene.

When the 3D painting mode is used the envelope is painted in the 3D volumetric space of the target object. Hence the defined envelope always maintains its relative position to the volumetric object in 3D space (Figure 41).

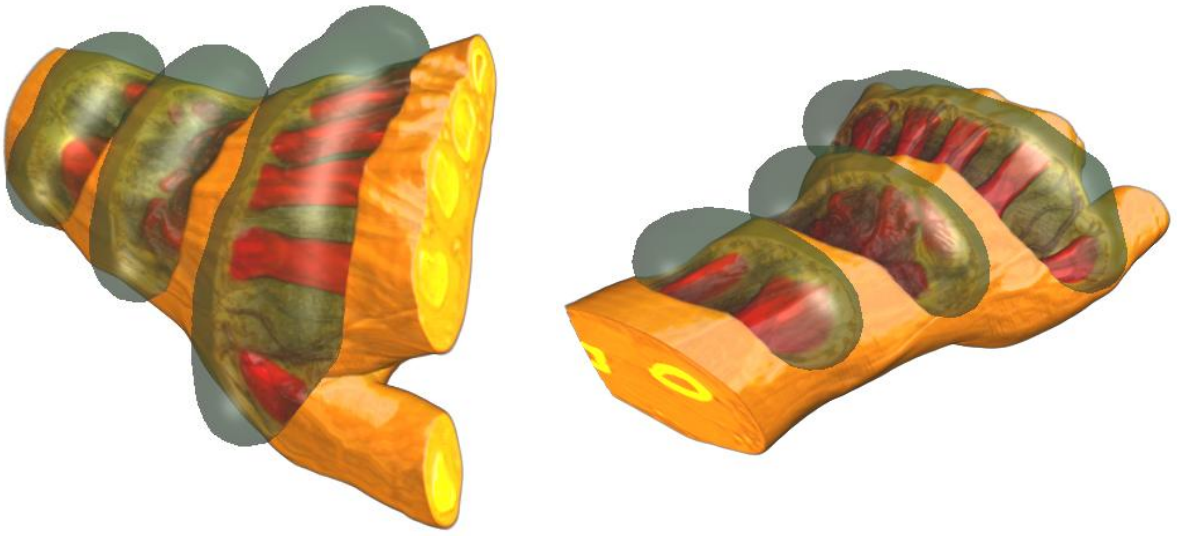


Figure 41: A 3D envelope defined in the space of the target volumetric object. The position and orientation of the envelope change according to the position and orientation of the visualized object.

To place a new brush stroke a *brush tip*, which represents a circular or a spherical user defined area either in a 2D or 3D space respectively, is used. In the current implementation, the brush tip and the painted envelope are realized using the technique of defining implicit 3D surfaces known as *Metaballs* [7] (Section 3.3.6). Metaballs have a characteristic of merging with each other based on their proximity, radius and the chosen field function, creating organic-looking and flowing shapes. This behavior is similar to the painting operations performed in various raster editing programs (e.g. Photoshop) and was the main reason behind choosing metaballs as the envelope definition mechanism.

3.3.2.1 Envelope Definition Process

After a new brush stroke is placed it specifies the 2D or 3D area where the envelope is now defined. To extend the envelope subsequent brush strokes are placed by moving the brush tip to the areas of interest. As the brush tip is being moved any parts of the volume that fall within its area of influence are instantly visualized using a different visualization style (defined for the parts of the volume inside the envelope (Section 3.3.3)). Thus the brush tip provides a real-time *preview* functionality that allows the user to see exactly what the envelope will look like and what features inside of the volumetric object it will enclose if a new brush stroke is placed at the current position of the brush tip. Hence the process of painting an envelope can be described as a sequence of placing new brush strokes in positions where the brush tip preview results achieve the desired visual effect. The process of painting a 3D envelope consisting of several preview and brush stroke placing steps is illustrated in Figure 42[19].

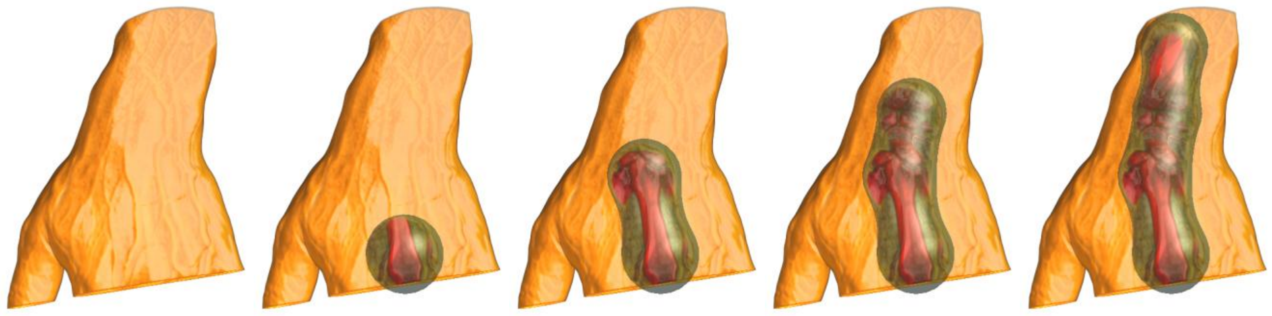


Figure 42: Envelope painting in 3D depicted as a sequence of steps of: a) previewing the results of the visualization by moving the brush tip to the desired location; b) placing a new brush stroke at the brush tip's current location.

[19] A 2D envelope is painted in the same way, but in the screen space of the visualization area.

3.3.2.2 Brush Tip Positioning

To control the position of the brush tip either in a 2D or 3D space a control scheme very similar to the previously described camera navigation model (Section 3.3.1) is used. Under this navigation model the user first orients the volumetric object into the position that they consider to be suitable for painting (see explanations in the next paragraph). Then by pressing down and holding the *Alt* key the user initiates the painting mode. By continuing to hold the Alt key and moving the mouse pointer in the visualization area on the screen they are able to see the brush tip as a circle or a sphere (depending on the chosen 2D or 3D painting mode) of the currently specified size centered at the current position of the mouse cursor. The parts of the volume that fall within the area defined by the brush tip are instantly highlighted (Section 3.3.3) using the alternative visualization parameters specified by the user. If the user decides to place a new brush stroke at the current position of the brush tip, they do it by clicking the left mouse button while still holding down the Alt key. When the Alt key is released the scene interaction mode reverts back to the camera navigation mode, allowing the user to adjust the orientation of the object in the scene and evaluate the results of the painting operation (the envelope remains in the scene until it is cleared (Section 3.3.3)). The above procedure can be repeated any number of times allowing the user to extend the envelope to enclose the desired volumetric features.

In the 3D painting mode, in addition to the screen space position, the depth of the brush tip needs to be controlled as well. For this purpose, similarly to the camera navigation model (Section 3.3.1), the scroll mouse button is used. By scrolling the wheel of the mouse while holding down the Alt key the user can adjust the current depth of the brush tip, thus moving it further away or closer to the viewer. After the desired depth of the brush tip is reached its value is preserved and is used to define a virtual *painting plane* that orients itself in such a way as to always stay parallel to the screen (Figure 43).

The painting plane defines the plane in which all of the new brush strokes are placed. Thus when the user moves the brush tip by changing the position of the mouse cursor in the screen space, they actually change the position of the brush tip as projected onto the painting plane defined at the current brush tip's depth. By moving the brush tip in this manner (moving the cursor on the screen and using the scroll wheel) the user controls its 3D position within the 3D space of the scene. Naturally the relative position and orientation of the volumetric object and the painting plane can be adjusted by the user at any moment by releasing the Alt key and using the standard camera navigation model. This rapid switching between the two very similar navigation models by using just the Alt key allows the user to quickly paint the envelope of the desired arbitrary shape. Furthermore, any parts of the volumetric object that fall within the 3D boundaries of the envelope are instantly visualized using the specified alternative visualization parameters (exactly the same way as in the case of the brush tip preview mode), thus allowing the user to see if the envelope they are currently painting achieves the desired contextual view.

It should be noted, that by default (i.e. when a new dataset is loaded into the program) the depth of the brush tip, and hence the depth of the painting plane, are set to a value such as to make the brush tip always visible (i.e. floating in front of the volumetric object in 3D space). This depth value is calculated by using the dimensions of the volumetric object's axis-aligned *Bounding Box*[20].

[20] In computer graphics a bounding box is the minimum box that encloses all parts of a 3D object.

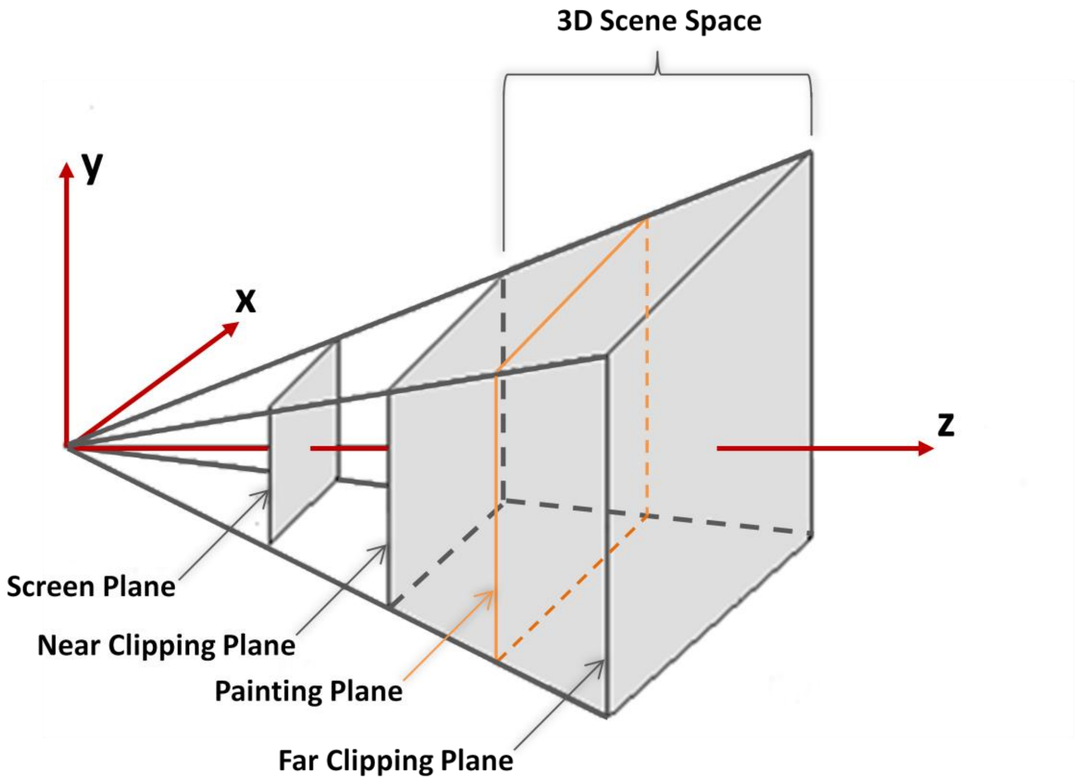


Figure 43: The painting plane always stays parallel to the screen plane, regardless of the orientation of the volumetric object in the 3D scene space. The current depth of the brush tip determines the depth of the painting plane, which can be adjusted within limits specified by the near and far clipping planes. When the user moves the mouse cursor on the screen plane its current position is projected onto the painting plane, thus defining the current 3D coordinates of the brush tip. New brush strokes can then be placed at the current 3D position of the brush tip on the painting plane, thus allowing the user to extend the envelope as needed.

In the 2D painting mode the depth of the painting plane is always set to a constant value and cannot be changed by the user. The position and the shape of the brush tip (and of the envelope if it is defined) are projected onto the painting plane and are used as a *stencil* to select parts of the volumetric object that fall inside of the projected contour along the current viewing direction (Z axis in Figure 43). Any parts of the volume that fall within the stencil's projected contour are visualized using the same alternative visualization mode as in the case of the 3D painting mode.

The difference from the 3D painting mode is that the depth values of the parts of the volume that fall inside of the stencil are disregarded (i.e. a stencil works in a way similar to an X-ray machine, by applying the visualization effect all the way through the object).

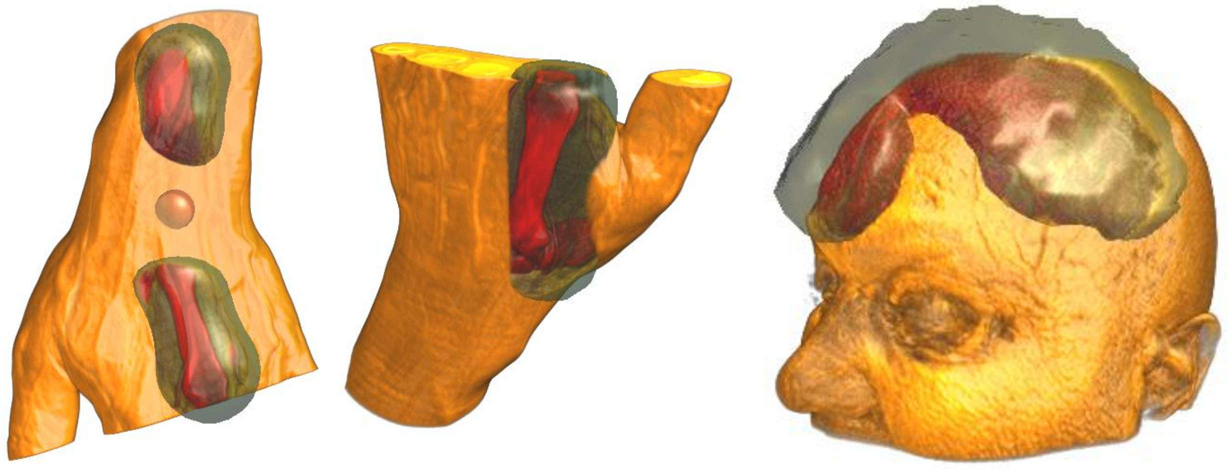
3.3.2.3 Free-hand and Surface 3D Painting Modes

The 3D painting mode described above, where the user fully controls the depth of the brush tip at all times, is called a *free-hand painting* mode. Naturally, depending on the desired type of exploration, cases might arise when it would be preferable to be able to automatically follow the shape of a volumetric object with the brush tip (e.g. when the user wants to paint an envelope that encloses only the features close to the surface of the object). In that case a mode called *surface painting* is available. In this mode the depth of the brush tip, and hence the depth of the painting plane, at the current projected position is overridden by the depth value of the object's surface, obtained by intersecting the ray fired from the current mouse cursor position on the screen into the volume space with the visible surface of the volumetric object (Section 3.3.7).

3.3.2.4 Additional Brush Tip Properties

As was described earlier, metaballs (Section 3.3.6) are used as the underlying mechanism for envelope definition. Any envelope is thus defined by a series of metaballs that merge themselves with each other based on their size, proximity and selected field function. In general the larger a metaball and the closer it is to other metaballs, the greater its influence and merging factor with other metaballs. As the brush tip is always the first metaball in the scene, it defines the attributes of every new metaball that is added to the series. Thus by controlling the size and other parameters of the brush tip the user can have more control over the shape of the painted envelope.

The brush tip can be defined as a negative brush tip, thus allowing the user to negate or *erase* the results of the previous painting operations (Figure 44(a)). In addition, in the 3D surface painting mode the depth of the object's surface penetration by the brush tip as well as its shape can be changed. In the current implementation the only shape (aside from a sphere) that is allowed is that of a cylinder[21]. A cylindrical brush tip will try to always orient itself perpendicular to the surface of a volumetric object (Figure 44(b)) (Section 3.3.7). In addition to its size, the *height* of a cylindrical brush tip can also be adjusted, thus providing additional flexibility for envelope definition (e.g. a flat envelope with a small height value can be used to follow the curvature of the skull to include only the bone but not the brain matter) (Figure 44(c)).



(a) (b) (c)

Figure 44: (a) A negative brush tip erasing a part of the previously painted envelope. (b) A cylindrical brush tip self-aligned with the cross-section of a finger and depth-adjusted to show the whole digit of the finger. (c) A flattened envelope defined using a cylindrical brush tip and surface-painted following the curvature of the skull.

[21] A cylindrical shape was chosen as the most useful one in terms of aiding in the envelope definition process. Other superquadratic shapes, such as stars and ellipsoids, did not contribute much in terms of usability and thus were excluded from the implementation.

3.3.3 GUI Controls

As was mentioned in Section 3.3.1, in addition to the visualization window, the interface also contains the GUI controls window that allows adjusting all the parameters related to the visualization, the envelope definition process and the painting modes. The GUI controls window is further subdivided into the following functional areas (labeled with capital letters in Figure 39):

Areas **A1** and **A2** contain slider controls allowing the user to adjust *the ranges of visible volume values* outside the envelope and inside the envelope respectively (Figure 45) (in Figure 39 the area outside the envelope is highlighted by the white dashed line, and the area inside the envelope is highlighted by the black dashed line). Maximum and current values are displayed to the right of each slider. "*Base Color*" buttons in each area allow the user to specify the color used to *tinge* volumetric features inside and outside the envelope (see Section 3.3.4 for details on MIDA technique).

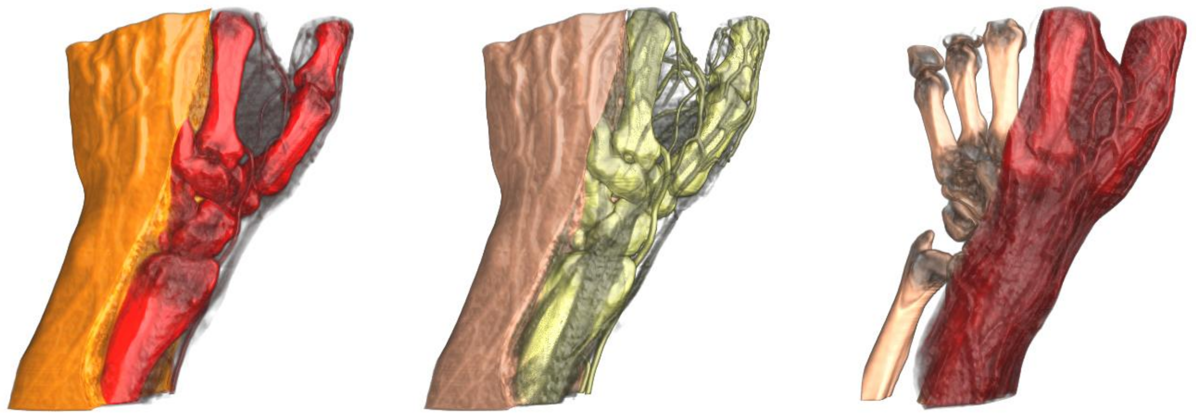


Figure 45: Visualizations of the palm using different visible value ranges and base colors for parts of the volume inside and outside the envelope.

Area **B** contains a slider controlling the *transparency* level of the envelope's surface and two buttons that allow the user to specify the *envelope color* and to *clear* it (i.e. completely remove it) (Figure 46).

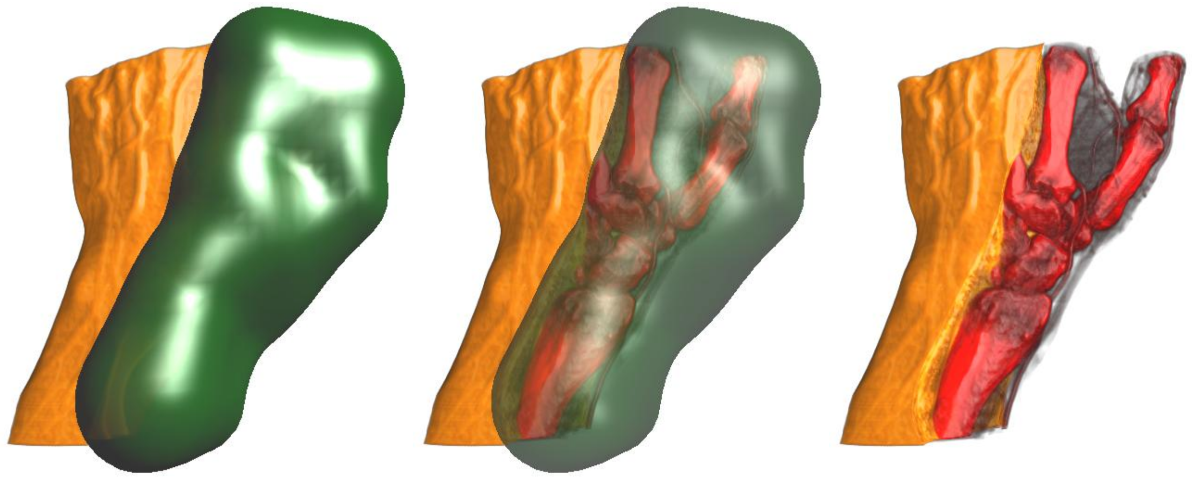


Figure 46: A green 3D envelope visualized in an opaque, semitransparent and transparent manner.

Area **C** contains controls to change the *size* of the brush tip, its *color* and to define it as *negative* (Figure 47).

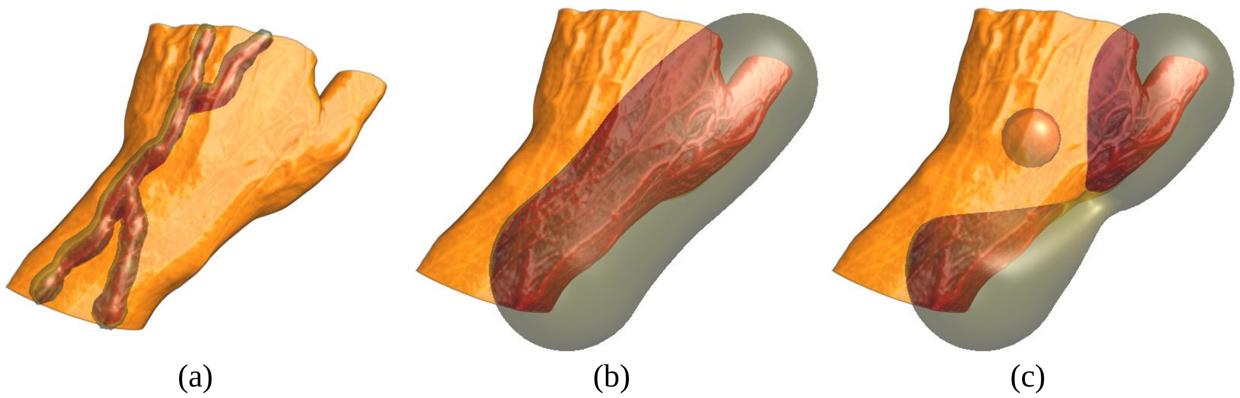


Figure 47: (a) A small-sized brush tip used to define the envelope tracing the outline of the veins on the surface of the palm. (b) A large brush tip used to quickly envelope a big part of the palm. (c) A negative brush tip used to "push away" the previously painted envelope.

Using controls in area **D** a user can switch to the *surface painting* mode, adjust how deep the brush tip penetrates the surface of a volumetric object by adjusting the *depth* slider or change the shape of the brush tip to a cylinder by selecting the *deformable brush tip* option and set the desired height of a cylinder by using the *height* slider control (Figure 44).

The last area **E** controls the activation of the *screen painting* mode. The only parameter that can be adjusted in this group is represented by a *show envelope edges* checkbox, that controls whether a semitransparent or a transparent but outlined version of a 2D envelope should be displayed (Figure 48). Note that in either case a level of transparency of either the outline or the envelope itself can be controlled with the transparency slider contained in area **B**.

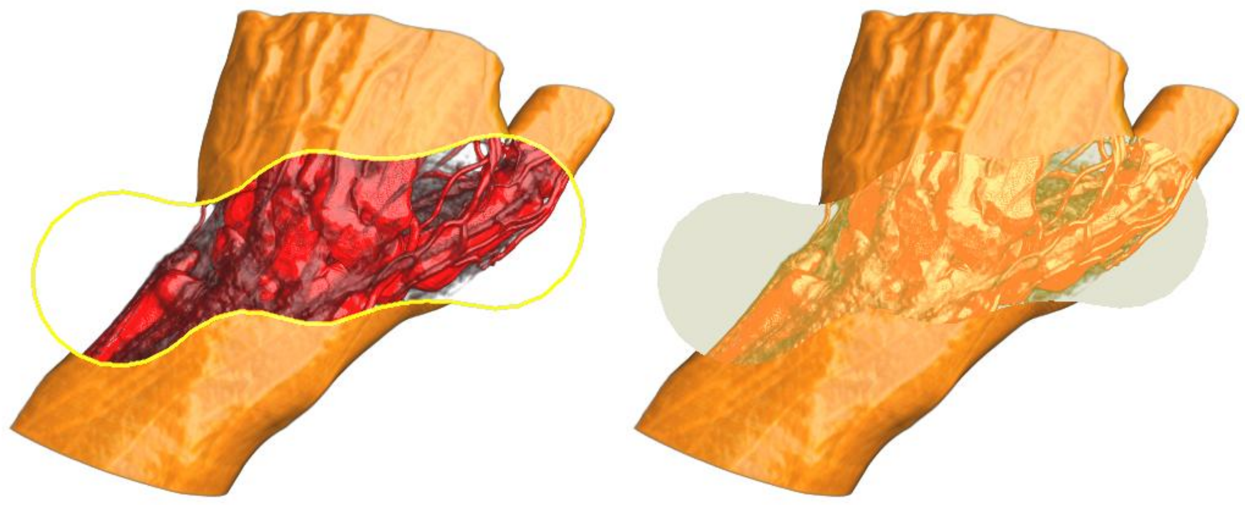


Figure 48: A transparent outlined version of a 2D envelope vs. a solid semitransparent version.

3.3.4 Maximum Intensity Difference Accumulation

As was mentioned in Section 2.1.7 MIDA [5] is an approach that allows the user to visualize volumetric datasets without having to specify a TF. Instead it exploits inherent data characteristics to map scalar field values to unique color/opacity values. The basic idea behind this approach is to alter the opacity accumulation behavior of a traditional DVR to incorporate some of the characteristics of MIP. In a traditional DVR as a ray traverses a volume and hits voxels on its path, the intensity values of the voxels are mapped to the color and opacity values specified by the TF (Figure 49). These values are then blended together using Equation (3.1 (Page 48), resulting in a monotonically growing accumulated opacity value (Figure 50(a)). This means that regardless of the prominence of the underlying volumetric features[22], if they are occluded by an opaque region, they won't contribute to the final 2D image. MIDA tries to address this issue by modulating any previously accumulated color/opacity value by an amount of the positive difference between the current and the previous intensity value (Figure 50(b)). Thus color and opacity accumulation formula from Section 3.2 becomes:

$$\begin{cases} c_i^* = \beta_i c_{i-1}^* + (1 + \beta_i \alpha_{i-1}^*)\alpha_i c_i \\ \alpha_i^* = \beta_i \alpha_{i-1}^* + (1 + \beta_i \alpha_{i-1}^*)\alpha_i \end{cases} \tag{3.2}$$

where $\beta_i = 1 - \delta_i$, and

$$\delta_i = \begin{cases} 1 & , if\ f_{P_i} > f_{max_i} \\ 0 & , otherwise \end{cases} \tag{3.3}$$

Here f_{P_i} represents the data value at the sample point P_i and f_{max_i} is the current maximum value along the ray which is also updated to f_{P_i} whenever $f_{P_i} > f_{max_i}$.

[22] The assumption here is that more prominent features are usually represented by higher voxel intensity values.

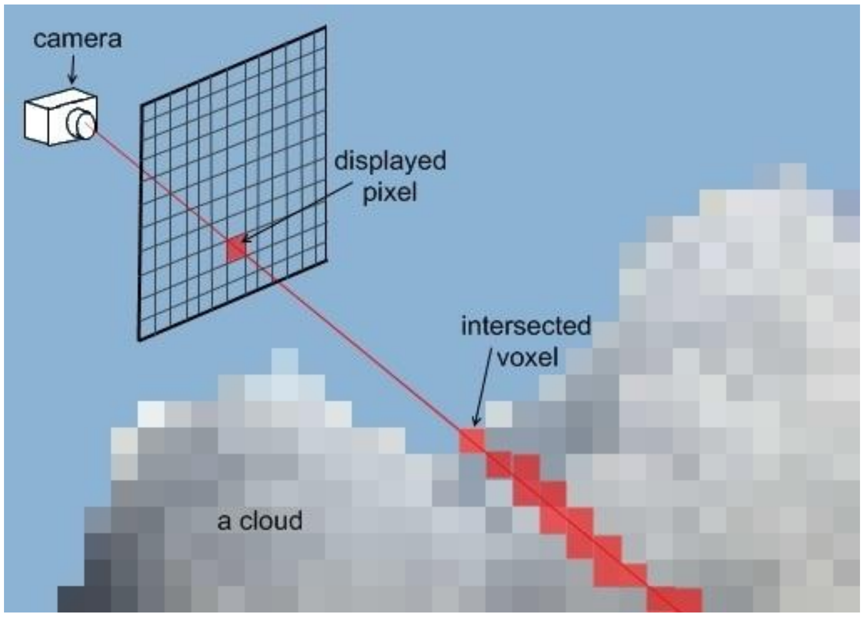


Figure 49: A ray traversing a volumetric space, while encountered voxels are mapped to color and opacity values and blended with the previously accumulated results.

The difference between DVR and MIDA opacity accumulation behaviours can be better understood by looking at Figure 50. As can be seen in the case of MIDA, the opacity of any previously accumulated results is modulated by the difference in the previous and the newly encountered local maximum. As can also be seen, some of the rather prominent features (check the last peak on the graph in Figure 50b) can still be obscured if the corresponding local maximum does not quite go over the previous value. To address this issue, a sub-range of sampled data values can be defined by the user to include only the desired volumetric features. This is also useful to exclude data at the lower part of the total range that usually corresponds to noise resulting from a scanner picking up the surrounding air and dust particles floating around the object as it is scanned.

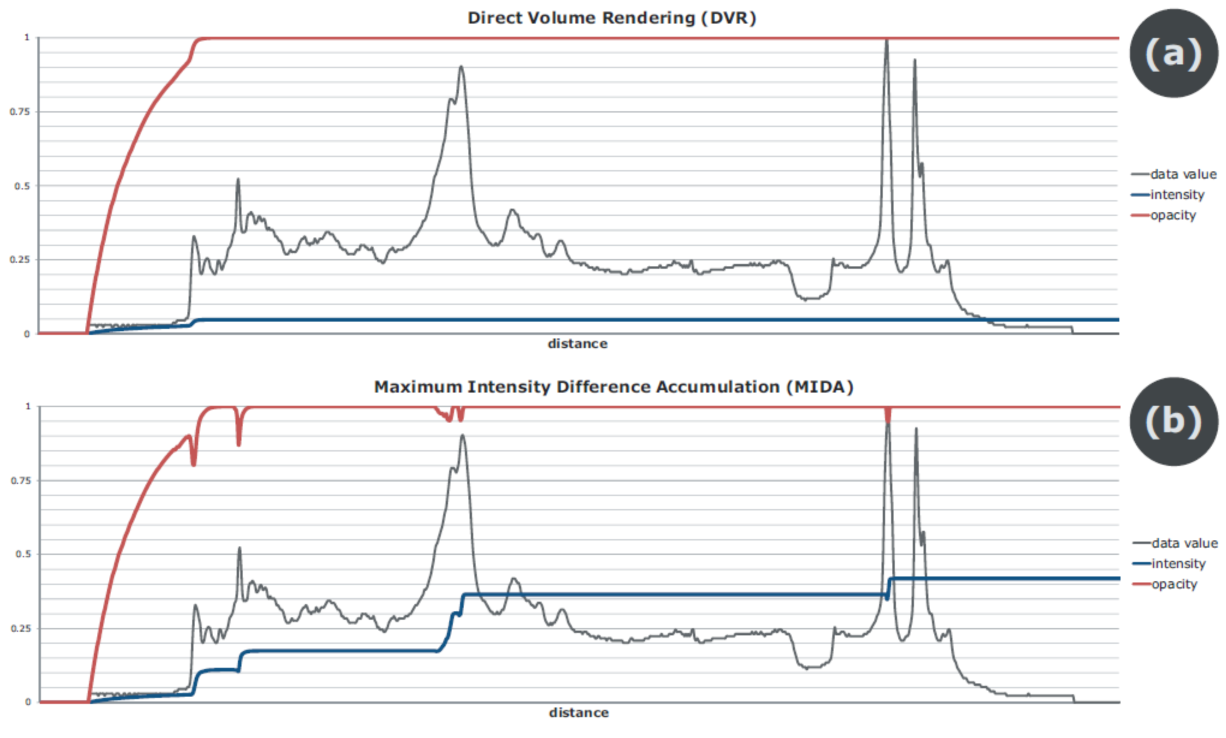


Figure 50: A comparison between (a) typical DVR and (b) MIDA ray profiles. In the case of MIDA the accumulated opacity value (red line) is modulated by the amount of the difference between the current and the previous local data value maximum (grey line). The blue line represents the accumulated color intensity of the resulting projected pixel on the screen.

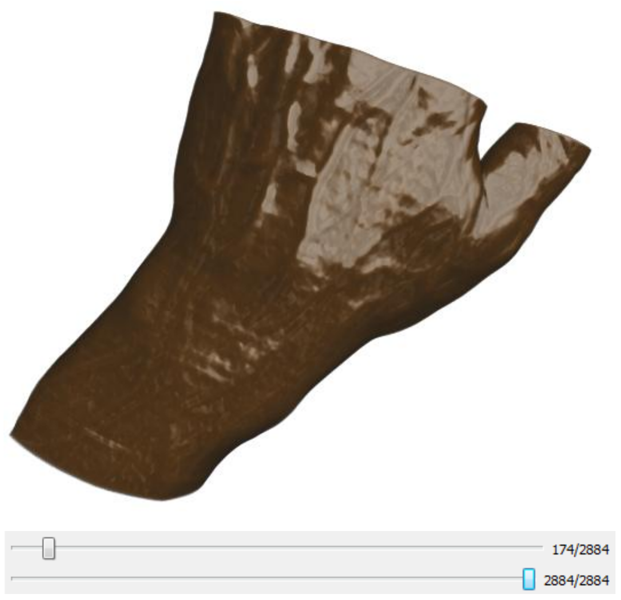
Under MIDA, in the absence of a TF, any sampled values (in accordance with a specified sub-range) are linearly mapped to a full range of grey scale intensities from black to white and opacities from 0 to 1. To provide for more pleasing visual results and a better degree of comprehension, grey scale intensities can also be multiplied by a so called *base color*, which *tinges* the resulting rendered volumetric objects according to user preferences.

One of the inherent issues with MIDA, that was discovered while implementing it, is that the resulting images suffer from inconsistent brightness; that is they are usually too dim when a MIDA sub-range is large and too bright when a MIDA sub-range is small. This behaviour is explained by the nature of the modulation process. When a MIDA sub-range is large chances are high that any previously accumulated color/opacity values will be modulated by the newly encountered values corresponding to the more prominent features. Since rays are travelling in the front-to-back order, any front facing structures will appear dimmed out. In contrast, when a MIDA range is small, the possibility of encountering a prominent feature and thus of any previous value being significantly modulated is much less, hence allowing for the original unmodulated value to be displayed (Figure 51). Naturally, results will also vary depending on the relative position of processed scalar values within the defined sub-range. To combat this shortcoming, a simple brightness compensation scheme was devised that adjusts the intensity of the base color depending on the ratio of the defined sub-range to the total range of all possible values and the user specified brightness compensation factor, as follows:

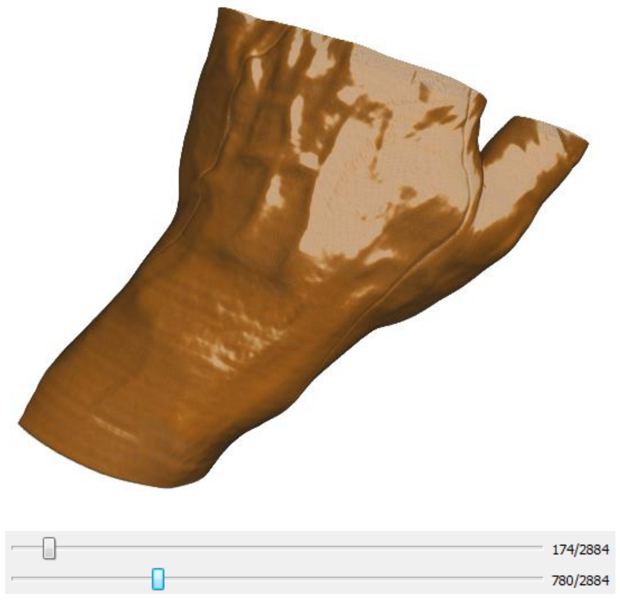
$$c_{base}^{*} = c_{base}\beta(\alpha_{const} + S/R) \tag{3.4}$$

where c_{base} is the user specified base color, β is the brightness compensation factor, α_{const} the constant amplification factor to avoid the dimming effect for extremely narrow sub-ranges, and S and R are the sub- and full value ranges respectively. The empirically determined values that provided good visual results for most datasets and that were used in the course of the conducted experiments and the user study (Section 4.2) were $\beta = 4.0$ and $\alpha_{const} = 0.5$[23] (Figure 52).

[23] In case the resulting image becomes too bright the colors could be easily adjusted by reducing the intensity of c_{base} using the standard Windows color selection window.

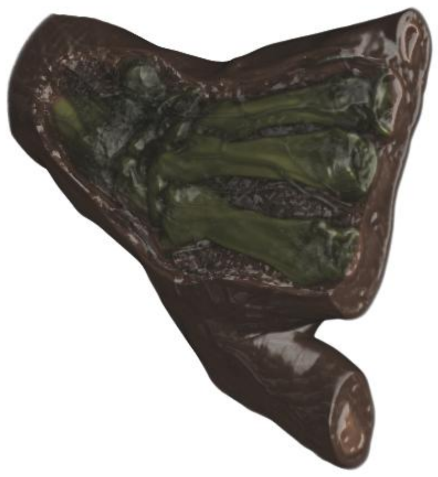


(a)

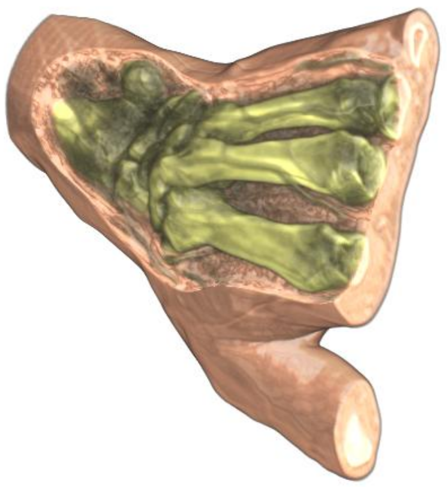


(b)

Figure 51: In the absence of the brightness compensation (a) a visualization of a hand using a wide MIDA sub-range has more definition, but is too dim. (b) Under the same conditions a much narrower MIDA sub-range produces a less detailed but noticeably brighter image.



(a)



(b)

Figure 52: A visualization of the same volumetric dataset (a) without and (b) with brightness compensation applied.

3.3.5 Front- to-Back Depth Peeling

In order to be able to define an ROI directly in a 3D space of a volume it was necessary to come up with an approach that would satisfy the following conditions:

- Define an ROI as a 3D shape.
- Allow real-time updating of a 3D shape as an ROI is being defined and modified.
- Provide enough depth cues for the user to be able to see clearly the spatial relationship of the volumetric features and the 3D shape defined by an ROI.

To achieve these requirements an approach based on *Depth Peeling* (DP) described in [64] was chosen. DP is a well known technique that allows for visualizing transparent polygonal objects of any complexity in an order-independent manner. It is based on the idea that while a standard *depth test*[24] provides the coordinates of the projected parts of the object (also known as *fragments*) closest to the viewer for every pixel, there also might be parts of the object in the scene that are second closest, third closest and so on. These second, third and generally nth fragments correspond to the surfaces of a polygonal object that under normal circumstances would be *culled* away. However when a polygonal object needs to be rendered in a semi-transparent way, this information has to be preserved and taken into account.

DP addresses this task by rendering the scene using *n* passes (Figure 53), where *n* corresponds to the number of rendered layers, or how deeply into the scene a viewer can peer. The first pass is rendered in a regular way and depth information for all of the closest fragments in the scene is

[24] In computer graphics a depth test is an operation typically performed as a part of the rendering process that allows only the front-most parts of the rendered 3D objects to be visible. Without it the resulting image would be incorrect with parts of the object's internal structures showing through.

generated. The second pass goes over the whole scene again, but now it compares the depth of any processed fragments to the depth information obtained in the first pass. Any new fragments with the depth less or equal to the values from the first pass are ignored and the depth buffer containing the depth values for all second closest fragments is generated. Naturally, color values from the first rendering pass are blended with the newly generated color values to achieve the correct semitransparent look. The process continues until the rendering pass fails to find any new fragments that satisfy the depth test condition (i.e. there is no more geometry left to render at the depth greater than the previously generated one). The process is illustrated in Figure 54, where each pass peels away layers of front-most facing fragments.

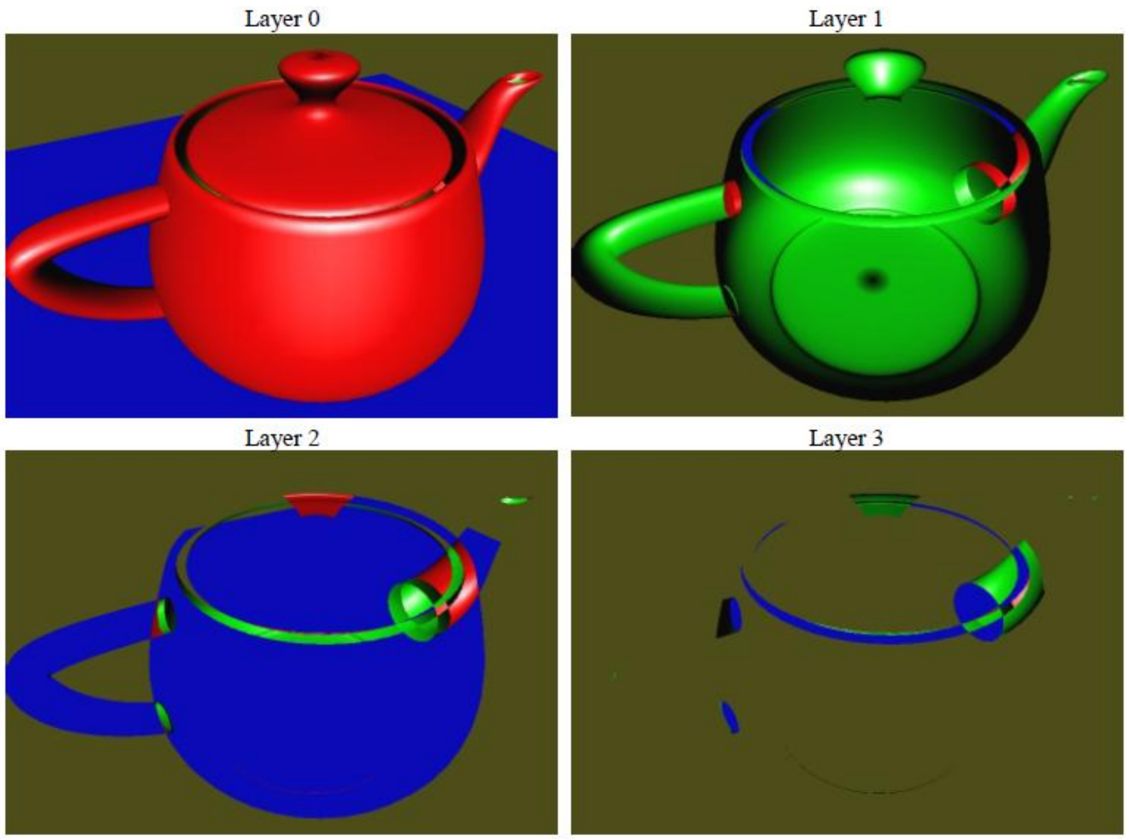


Figure 53: A series of images displaying peeling of the fragments using DP algorithm [64].

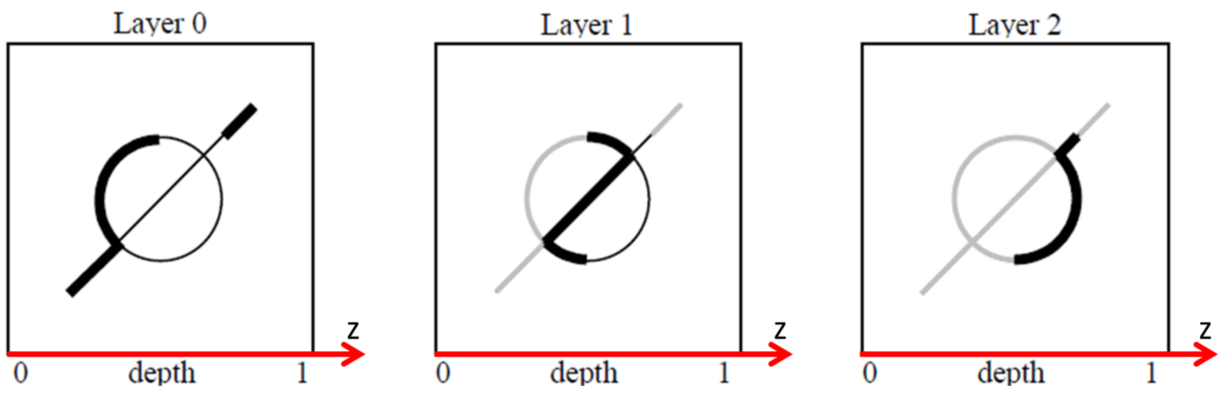


Figure 54: DP stripping away layers with each successive pass. The first pass peels away the front-most (leftmost) fragments, with hidden fragments shown in thin black lines. In the subsequent passes the already peeled away fragments are shown in light grey lines.

Unlike the original technique, which deals only with peeling of polygonal models, the solution implemented in this thesis combines polygon-based peeling with VR by breaking down the ray casting process into stages according to the intermediate results of the peeling process. Since ray casting is done in front-to-back order, the peeling process is also done in the same order and thus is called *Front-to-Back Depth Peeling* (FtBDP)[25] [65]. FtBDP can be described as a sequence of the following steps:

1. The front-most polygonal geometry corresponding to the front surface of the envelope facing the user is rendered in a normal way. The depths of all corresponding fragments for every pixel in the frame are stored in the depth buffer.

2. The MIDA ray casting stage is initiated with the color and depth buffers from step 1 passed into it using special memory buffers called Texture buffers (further referred to as simply *textures*). The depth test is performed for any rays cast into the scene and only

[25] Technically depth peeling can be performed in any order, as long as the depth test is performed properly and the results are blended together correctly.

those parts of the volume that fall outside of the envelope are rendered. The colors of the previous peeling pass are blended with the results of the ray casting to achieve the correct visual effect. The positions of the rays intersecting the peeled surface from step 1 are kept in a separate buffer for future use.

3. The next envelope peeling stage is initiated with the depth results of the previous peeling pass passed to it in a texture. New fragments are chosen based on satisfying the depth test condition (i.e. comparing their depths to the values stored in the above texture). The depth values of all newly chosen fragments are stored in the depth buffer.

4. The MIDA ray casting stage is initiated with the color and depth buffers from step 3 as well as the ray positions and accumulated color results from step 2 passed into it as textures. The ray starting positions are initialized according to the previously stored values. The depth test is performed for any rays cast into the scene. Depending on whether the sampled values fall inside or outside of the envelope, they are rendered using different MIDA visualization settings (Section 3.3.6). The colors of the previous peeling pass and the previous ray casting pass are blended with the current ray casting results to achieve the correct visual effect. The positions of the rays intersecting the peeled surface from step 3 are kept in a separate buffer for future use.

5. Steps 3 and 4 are repeated until the envelope peeling stage reports 0 newly chosen fragments.

Several things are also considered during execution of the aforementioned sequence of steps:

- To distinguish between the parts of the volume that fall inside and outside of the envelope, the information about polygon face orientation is used. This means that any ray sampled values that fall between a front facing and a back facing fragment are considered to be inside of the envelope; otherwise – outside.

- Since ray positions are properly kept and initialized between separate MIDA ray casting stages no additional overhead is put on the GPU in terms of computational workload associated with ray casting. There is, however, overhead associated with each stage's shader code initialization and execution as well as with an *overdraw* effect resulting from the DP algorithm itself (i.e. the full geometry of the scene has to be drawn and tested in each polygon peeling pass).

- The geometry corresponding to the envelope is tagged on a vertex attribute level by a special key, thus allowing shaders to distinguish between the regular polygon-based geometry present in the scene and the envelope geometry. This in turn allows combining visualization of any number of polygon-based objects in the scene, that properly intersect and are correctly blended with volumetric and envelope features.

- Extreme cases where the envelope geometry crosses the boundaries of the specified metaball grid (Section 3.3.6) and thus no longer forms a closed surface are considered and handled correctly[26].

[26] In general in the current implementation an envelope does not have to be a closed surface.

3.3.6 Envelope Definition with GPU Accelerated Metaballs

The envisioned 3D painting metaphor necessitated defining an envelope in such a way as to allow for its growing and editing in a manner similar to placing brush strokes. A metaball representation of the envelope was chosen since it allows for an implicit surface definition and easy creation of flowing and blending shapes, as well as for editing of the envelope by using metaballs with negative radii. In its general form, a metaball is defined by a function in *n*-dimensions [7]. For a Euclidian 3D space, a common representation of a regular quadratic metaball can be expressed as:

$$f(x, y, z) = 1/((x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2) \tag{3.5}$$

where (x_0, y_0, z_0) are the coordinates of the metaball's center. However, due to the floating point division operation this representation is computationally expensive and thus approximate polynomial functions are typically used.

For the purposes of this thesis two types of metaballs are used:

- A regular quadratic metaball, representing a spherical brush stroke, with a field falloff function defined by a six' degree Wyvill polynomial [66] as:

$$f(d, R) = \begin{cases} -\frac{4}{9}\left(\frac{d}{R}\right)^6 + \frac{17}{9}\left(\frac{d}{R}\right)^4 - \frac{22}{9}\left(\frac{d}{R}\right)^2 & , if\ 0 \le d \le R \\ 0 & , if\ d > R \end{cases} \tag{3.6}$$

  where d is the distance from the current location in space to the metaball's center, and R is the metaball's radius. Wyvill's function was primarily chosen because of its visually pleasing metaball blending results.

- A superquadratic metaball, that is used to define shapes used primarily in surface painting/peeling operations:

$$f(x,y,z) = 1/\left(\left(\left|\frac{d_x}{R_x}\right|^{\frac{2}{n_2}} + \left|\frac{d_y}{R_y}\right|^{\frac{2}{n_2}}\right)^{n_2/n_1} + \left|\frac{d_z}{R_z}\right|^{\frac{2}{n_1}}\right) \qquad (3.7)$$

  where $d(x,y,z)$ is the distance from the metaball's center to the current point in space, $R(x,y,z)$ specifies the size of the metaball, and n_1 and n_2 are the exponents defining the shape of the superquadratic metaball.

Within the implementation each metaball is described by a set of parameters, such as position, radius, color, a negative flag (in case the metaball emits a negative field) and a superquadratic flag. If a metaball is superquadratic, then additional parameters are specified, including scale factors along X, Y and Z axes, X, Y and Z exponents, and a rotation matrix, that specifies the orientation of the metaball's Z axis in space. These parameters are used in the process of constructing an *isosurface* [67] that represents a combined field effect of all metaballs present in the scene.

A common approach used to construct an isosurface is based on the idea of subdividing a volume space containing all metaballs into a regular grid of volumetric cells or voxels. The size and the resolution of the grid depend on the desired precision of the final isosurface approximation. The values of each metaball's field function are then evaluated for every vertex of the voxel grid. The resulting function values are added and stored and the vertex locations, thus representing a combined field effect of all metaballs at every vertex of the grid.

The process of defining an isosurface is inherently connected to a *threshold value*. An isosurface can be defined as a surface that consists of only points with the chosen constant threshold value. Once the isosurface threshold value is specified and the voxel grid containing the combined metaball field effect values is calculated, the isosurface can be approximated by a polygonal mesh representation using one of the well-known isosurface generation algorithms, such as *Marching Cubes* [68] or *Marching Tetrahedrons* [69]. These methods work by stepping or *marching* through the values of the combined field effect at each vertex of the voxel grid and comparing them to the chosen isosurface threshold value. If the currently considered voxel contains the specified threshold value (i.e. one or more corners of the voxel's cube have the field value that is greater or equal to the threshold value and one or more corners have the field value that is smaller or equal to the threshold value) then, depending on the particular configuration of the corners containing the value, the polygons needed to represent the corresponding part of the isosurface that passes through the voxel are generated. As the algorithm proceeds through the voxels of the grid the individual polygons are fused into the resulting polygonal mesh.

To provide for interactive frame-rates a GPU accelerated version of the marching cubes algorithm has been implemented as a part of this thesis. The implementation utilizes a massively parallel architecture of modern day GPUs and employs a two-stage approach to achieve optimum performance.

Stage 1: A grid of the user specified dimensions and resolution is generated (64x64x64 grid was used throughout all the tests and the user study). OpenGL vertex shader programs are then executed in parallel, each one processing a single vertex in a grid to provide for the maximum level of parallelism. For each vertex all of the specified metaballs are evaluated and the combined field effect value, the surface normal direction and the color are calculated. The results are written into a separate array that will be used as an input to the next stage.

Stage 2: In this stage instead of vertices every shader program is handling a cube combined of 8 previously computed vertices and their attributes. Depending on whether the cube is intersected by the isosurface, appropriate triangles are emitted by the OpenGL geometry shader. For every emitted triangle the normal and the vertex color values are interpolated using the information calculated in stage 1.

It should be noted that the configuration of the envelope and thus the combined field effect of all the metaballs present in the scene are changed only when the user actively explores the volume by editing the envelope's geometry. In all other cases it would be wasteful to reconstruct the isosurface for every newly generated frame (e.g. in the cases when the scene is merely being rotated or zoomed, without the envelope actually being changed). Therefore geometry generated by the aforementioned two-stage approach is captured in a dynamically allocated vertex array buffer through an operation known as *Transform-Feedback* [9]. Later this vertex array could be reused any number of times to render the envelope without having to calculate its underlying geometry.

3.3.7 Surface Painting Mode Implementation

The surface painting mode allows painting on the surface of a volumetric object by following its shape. To provide this functionality the isosurface of the volumetric object is generated. The generation of the isosurface is performed by employing the same MIDA ray casting algorithm, with the only distinction being that the process of evaluating the samples along the ray's path is stopped as soon as the first non-transparent voxel is encountered. At this point the ray's position and the object's normal value are stored in two separate 2D buffers that will be later used to guide the surface painting process. Since each pair of values in the two buffers corresponds to a certain point on the screen plane, by moving the mouse cursor to the desired location on the screen the user can specify which pair of values from the two buffers will be used to override the current brush tip's (and hence the corresponding metaball's) depth and, in the case of a superquadratic metaball, orientation (the direction of the normal at the current position is used to calculate the rotation matrix of the superquadratic metaball).

One thing that has to be considered when it comes to superquadratic metaballs is that, due to the nature of volumetric datasets, the obtained 2D buffer or *map* containing the normal values can be quite noisy. This in turn can result in unpleasant abrupt changes in the orientation of a superquadratic metaball as it glides along the surface. To combat this problem two techniques are used:

- First, a 5 step isosurface refinement routine is used inside the isosurface generation shader program. It finds a finer matching isosurface by subdividing the current ray stepping size into increasingly smaller values and probing the volume space in both directions from the current position along the ray. This produces not only a smoother isosurface but also a more uniformly distributed map of normal values.

- To further reduce the noise a simple *convolution* operation with a Gaussian blurring kernel [70] is applied to the normal map.

In a general sense, a convolution is a mathematical operation that combines two different functions and produces a third function that usually represents a modified version of one of the initial functions. In applications involving 2D images, a convolution is often used for filtering and can be defined as an operation in which a final pixel's RGBA values are a result of a weighting operation on neighboring pixel values. Typically it is represented by a square matrix that defines weights assigned to each one of the neighboring pixels. This matrix is called a *convolution kernel* and depending on its size and coefficients allows achieving various visual effects, such as blurring, edge detection, embossing, etc.

One of the most popular filtering kernels is the Gaussian kernel that can be described by the following function:

$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{3.8}$$

where σ is a parameter that controls the kernel's width or area of effect.

When applied in the context of 2D images a Gaussian function is both separable and radially symmetric. This in turn means that instead of computing the pixel's value by evaluating all of its neighboring pixels in accordance with the size of the kernel (i.e. a kernel of size 5x5 will require 25 pixel value lookup operations) it is possible to achieve the same visual effect by dividing the operation into two separate passes. In the first pass a 1D kernel is used to blur the image in either a horizontal or a vertical direction. In the second pass the same 1D kernel is used to blur the image in the remaining direction (against the original 5x5 Gaussian kernel the use of two 1D kernels of sizes 1x5 and 5x1 will result in 15 less pixel value lookup operations). The results of this two stage approach can be observed in Figure 55.

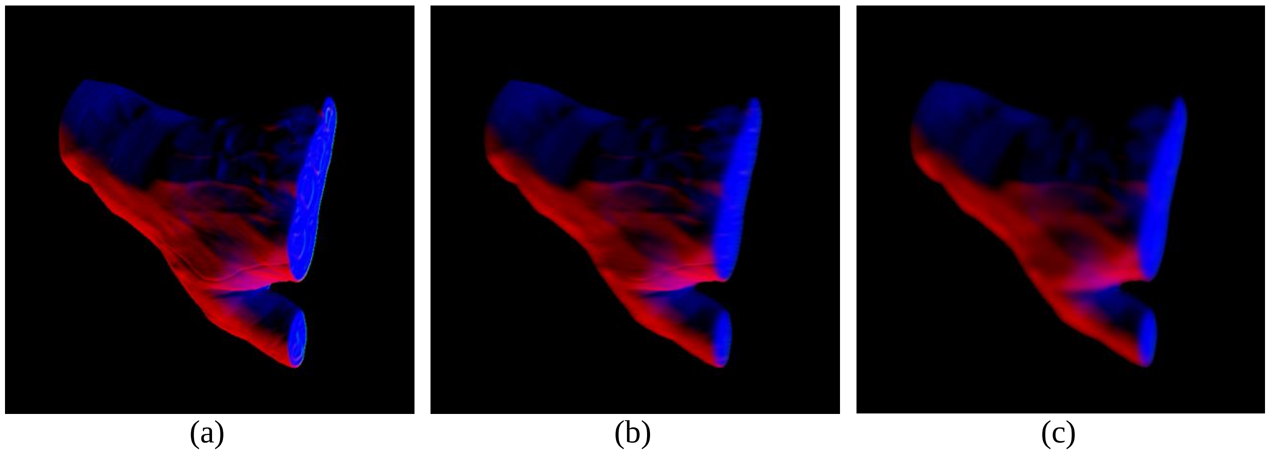


Figure 55: (a) A normal map without filtering applied. (b) A normal map with the 1x9 horizontal Gaussian kernel applied. (c) A normal map with the 9x1 vertical Gaussian kernel applied.

It should be noted that a traditional full-image Gaussian filter would produce erroneous filtered normal values in the areas corresponding to the edges of the projected volumetric shape (by weighting the neighbouring values that correspond to the empty space in the scene). Hence an edge-preserving variation of the filtering technique is used, that takes into account only the pixels of the normal map that fall inside of the projected volumetric shape.

Chapter 4: Evaluation and Results

The primary goal of this thesis was to provide an alternative approach to volumetric dataset exploration that would combine qualities such as a fast learning curve, intuitiveness, ease of control and flexibility[27]. A user study (Section 4.2) was conducted in order to evaluate the first three qualities. A series of experiments presented in Section 4.1 demonstrates the flexibility of the proposed approach by generating various context-preserving visualizations of several volumetric datasets. In addition, considering the current state of maturity of similar products in the industry, the ability to provide visually appealing and contextually meaningful visualizations at interactive frame-rates throughout the whole interaction and exploration user experience was a given. Section 4.3 is dedicated to the performance evaluation of the system under different visualization conditions. Finally, designing and implementing a future-proof visualization model should consider a matter of extensibility[28] as well. Section 4.4 demonstrates the extensibility potential of the proposed approach by integrating two additional visualization modes into the system.

4.1 Exploration Flexibility

This section highlights the flexibility of the proposed approach in terms of the types of exploration tasks that can be performed. It presents a number of examples with each one focusing on exploring a given volumetric dataset while trying to achieve the desired contextual view by exposing certain features of interest. The painting modes and techniques as well as the relevant GUI controls used to achieve each of the desired contextual views are also covered.

[27] Flexibility here means providing several ways of visualizing features of interest, i.e. ROIs.
[28] Extensibility here means the ability to extend the visualization capabilities of the system by adding alternative visualization modes.

4.1.1 3D Painting Mode

The 3D painting mode allows exposing the features of interest in a volumetric dataset by enclosing them in a 3D envelope. In its turn the operation of exposing can be performed in two distinct ways that are commonly recognized in the field of VR.

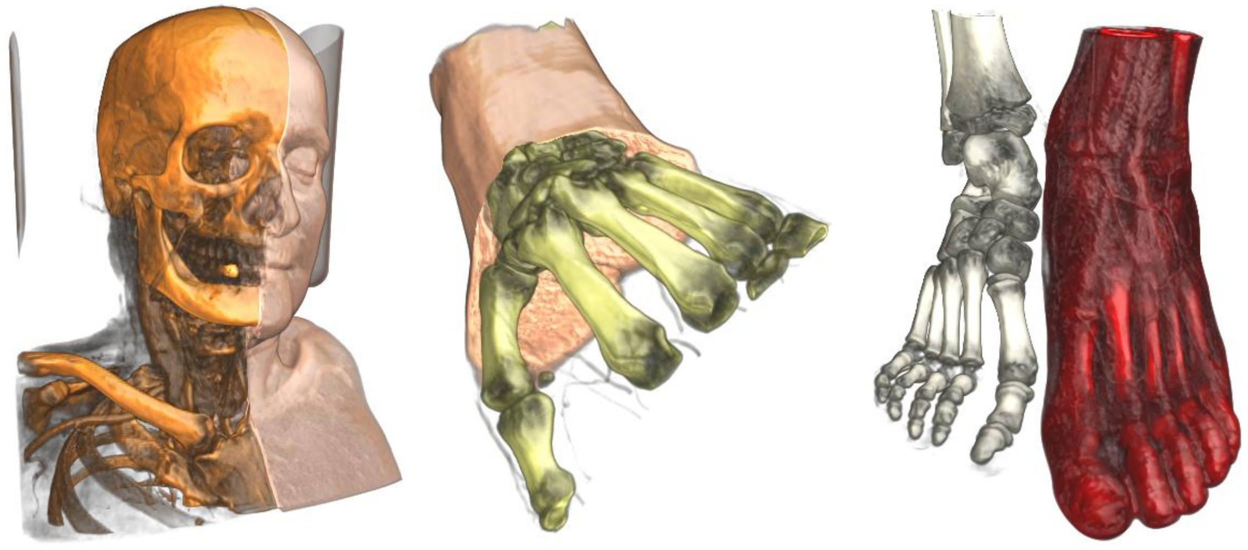
4.1.1.1 Context-Preserving Exploration

One of the most common tasks in the field of VR in general and in the medical VR in particular is to provide a contextual representation of the data by displaying the features of interest in a distinctly different way from the surrounding (i.e. contextual) parts of the volume. This subsection demonstrates the flexibility of the proposed 3D painting approach by achieving a number of contextual data views. These contextual views correspond to volumetric data exploration tasks, typically encountered in the medical field, which are centered around the process of revealing various hard and soft tissues in the context of the surrounding hard and soft tissues (Figure 56). Both free-hand and surface painting modes were used to achieve the visual results presented in Figure 56.

For example, in Figure 56(a) the free-hand painting technique was used to paint a 3D envelope in the plane that separated the volume space approximately into two equal parts. Then the MIDA range sliders for both the parts of the head inside and outside of the envelope were used to reveal the bone and tendons, and the skin layer respectively. Finally, the base colors were adjusted to achieve a more realistic look.

In the case of the hand screenshot (Figure 56(b)) the surface painting technique with a flattened brush tip was used to roughly isolate the skin layer from the rest of the dataset's features. The MIDA sliders corresponding to the parts of the volume enclosed in the envelope were adjusted to exclude any values (i.e. some of the blood vessels and muscle tissue close to the skin layer were also enclosed by the envelope, and thus had to be filtered out) except those corresponding to the skin layer. Since the envelope only enclosed the parts close to the skin layer, adjusting the MIDA sliders corresponding to the parts of the volume outside of the envelope to reveal the bones was a straightforward task.

To produce the third screenshot (Figure 56(c)) containing the image of two feet (one showing only the bones and the other the bones and the muscles), first the surface painting mode with a spherical brush tip was used to roughly isolate one foot from the other, then the free-hand painting mode was employed to tweak the envelope's shape to achieve the exact desired enclosure of the features (i.e. to provide a clean separation of the two feet). Finally, the MIDA sliders and the base colors corresponding to the parts of the volume inside and outside of the envelope were adjusted to produce the desired contextual view.



(a) (b) (c)

Figure 56: The results of visualizing different volumetric datasets ((a) a head , (b) a palm and (c) two feet) while providing the desired level of contextual information.

4.1.1.2 Carving

Another commonly encountered type of 3D exploration task exposes the features of interest by completely removing the occluding parts of the volume. This type of interaction with a volume is often referred to as *carving* (or *peeling* when applied in the context of a surface), and is commonly used in cases when the user wants to examine the outline or shape of the structures inside the volume. Due to the fact that MIDA visualization works with a settable range of visualized values, carving can be easily achieved by specifying an empty range (i.e. a range in which the minimum and maximum visualized values are the same). This subsection demonstrates the flexibility of the proposed approach by performing a series of carving/peeling tasks. The rest of the exploration process remains exactly the same as in the case of the context-preserving exploration and all of the aforementioned painting techniques can be applied (Figure 57).

In Figure 57(a) a part of a volumetric piggybank dataset was carved away using the free-hand painting mode.

The carving of the head in Figure 57(b) was performed by free-hand painting a 3D envelope in the plane roughly aligned with the profile of the head (much like the case presented in Figure 56(b)).

To achieve the contextual view shown in Figure 57(c) the surface painting mode with a flattened brush tip with a low height value (so that the skull, but not the brain underneath, could be peeled away) was used.

In the three examples above, the MIDA sliders for the parts of the volume enclosed by the envelope were adjusted to set the visible sub-range of volume values to 0.

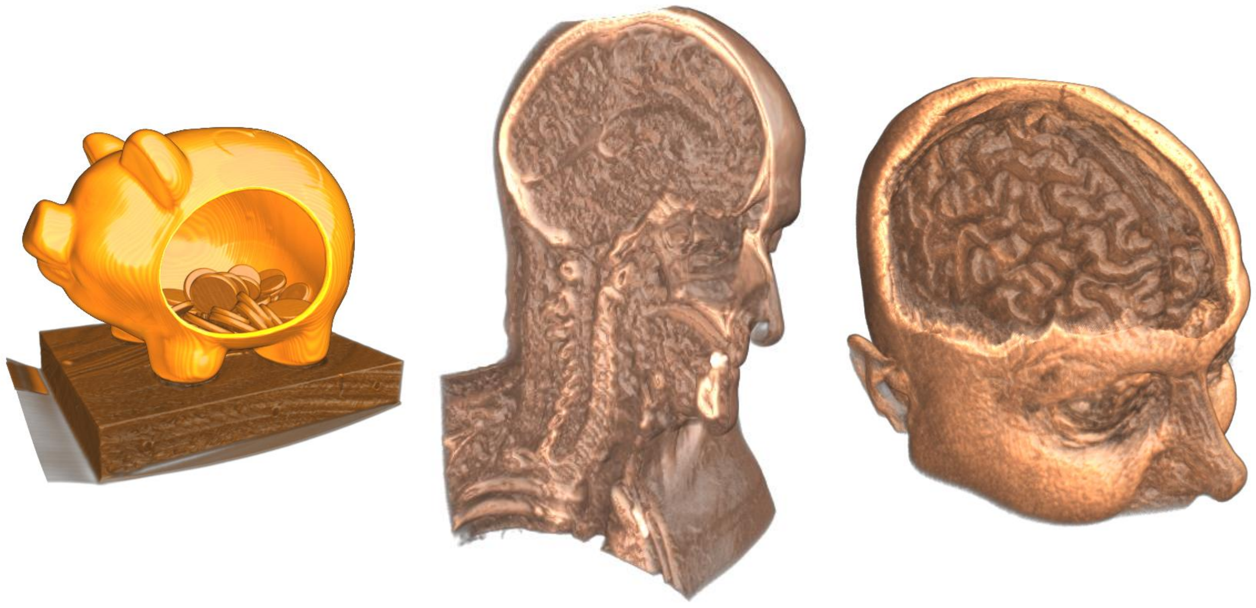

(a) (b) (c)

Figure 57: Carving and peeling are used to expose internal features and to remove the occluding parts in volumetric datasets.

4.1.2 2D Painting Mode

Often in the presence of fuzzy data with a low level of intensity value variability (such as MRI scans of various soft tissues, e.g. a soft part of a fruit, brain matter or various small structures that are simply beyond a scanner's resolution ability) it is difficult to isolate features of interest while operating in a 3D space. This is due to the fact that fuzzy data doesn't have a clearly defined surface and suffers heavily from partial volume effects[29]. To visualize such data, techniques that provide semi-translucent spatial representations of the data are often used. This in turn can lead to ambiguities while interpreting the results of the visualized information as features that are

[29] In other words it is difficult to tell where one type of tissue or matter ends and another begins.

closer to the viewer clash with the ones that are further away. In this case the 2D painting mode can provide additional cues that aid in the understanding of spatial relationships and internal structure of the features of interest. By painting an arbitrarily shaped 2D window on the screen the user can rotate and move the target object and adjust the visualization parameters in order to obtain the sought after information.

As an example of such a scenario can be demonstrated using an MRI scan of an orange. The goal is to collect information about the internal structure of the pulp, its shape and spatial relationship to the rind. The results of the experiment illustrating this scenario are shown in Figure 58.

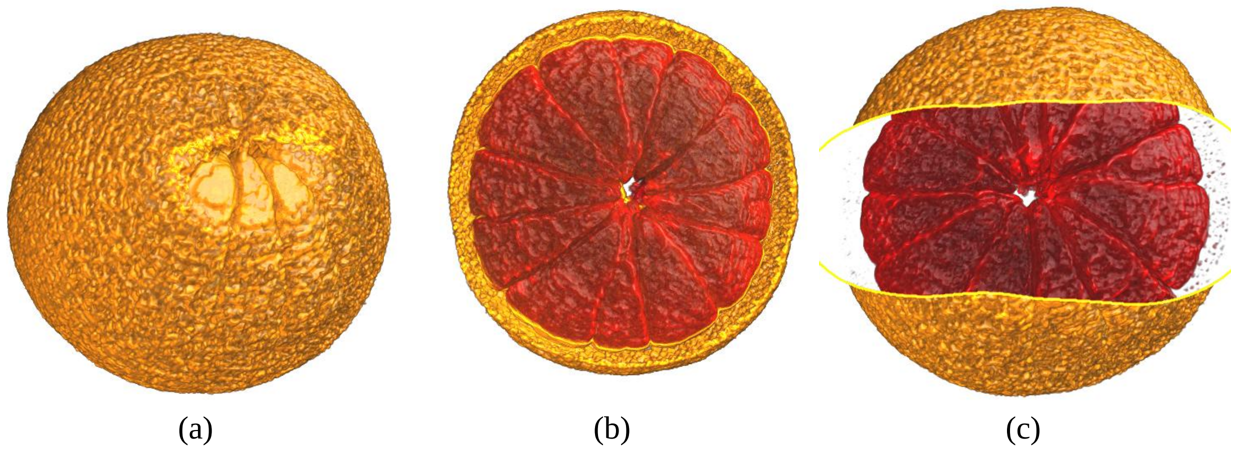


Figure 58: (a) An orange visualized without an envelope applied. (b) A 3D envelope is painted enclosing a part of the orange. (c) A 2D envelope is used to see all the way through the orange to better reveal internal structures.

Note that the 2D painting mode is also useful when the user doesn't want to deal with the depth of the painted envelope and instead wants to apply the visual effect inside of the envelope to all the structures in the scene regardless of their distance to the viewing point.

4.2 User Study

A user study was conducted in order to evaluate how intuitive and easy to learn and control the proposed painting metaphor and its implementation were.

Ten male and one female subject participated in the study. Ten of the subjects were Ryerson undergraduate students and one subject was a Ryerson MSc graduate student. All of the participants were full-time students of either the department of computer science or engineering. All subjects were first-time users of the implemented exploration technique and none had any previous experience with any medical or VR software. Approximately half however did have limited experience with various 3D modelling programs (i.e. Blender, Maya, etc.). In order to better gauge the level of participants' proficiency performing navigational tasks in virtual 3D environments, they were also asked to state the approximate number of *Hours per Week* (HPW) spent using a mouse and playing video games, with the average numbers coming to 25.5HPW and 11.5HPW respectively.

The participants were asked to use the implemented software to perform several trial tasks. The goal of each trial was to achieve an approximate visual match to a pre-rendered image (Figure 59). A total of nine trials, each based on a different dataset, were used. The trials were further broken down into 3 groups: three 2D painting trials, three 3D volume carving trials and three 3D context-preserving trials. In the course of the trials from both the 3D carving and 3D context-preserving groups the users had to use both the free-hand and the surface painting modes. Before each set of trials each participant was given instructions using a separate demonstration dataset and was also given time to play around with the software to familiarize themselves with the visualization controls and the painting mechanism. The time for each trial was recorded for statistical purposes, thought the participants were informed that achieving a visual match in the shortest possible time was not an objective. The average measured trial completion time was 1 minute. Upon completion of all trials the participants were asked to fill out the questionnaire and indicate their level of agreement or disagreement with each statement using a 7-point Likert scale[30]. The results of the questionnaire are shown in Figure 60[31].

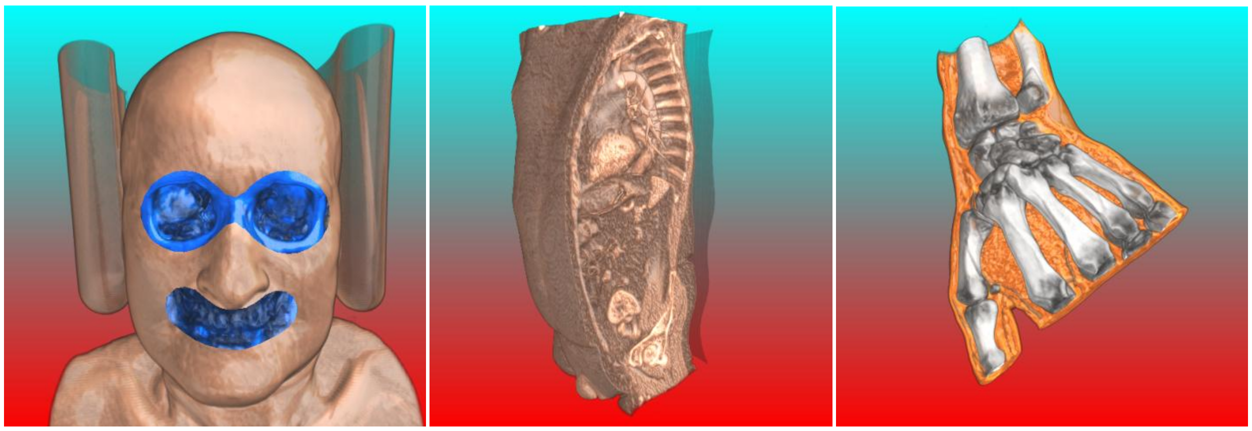

(a) (b) (c)

Figure 59: Examples of pre-rendered images from each trial group: (a) an example of a 2D painting trial, (b) an example of a 3D carving trial and (c) an example of 3D context-preserving trial.

[30] A Likert scale is a psychometric scale, where a respondent specifies their level of agreement or disagreement to a statement on a symmetric agree-disagree scale.
[31] The white bars correspond to the minimum and maximum marks given by the participants for each question.

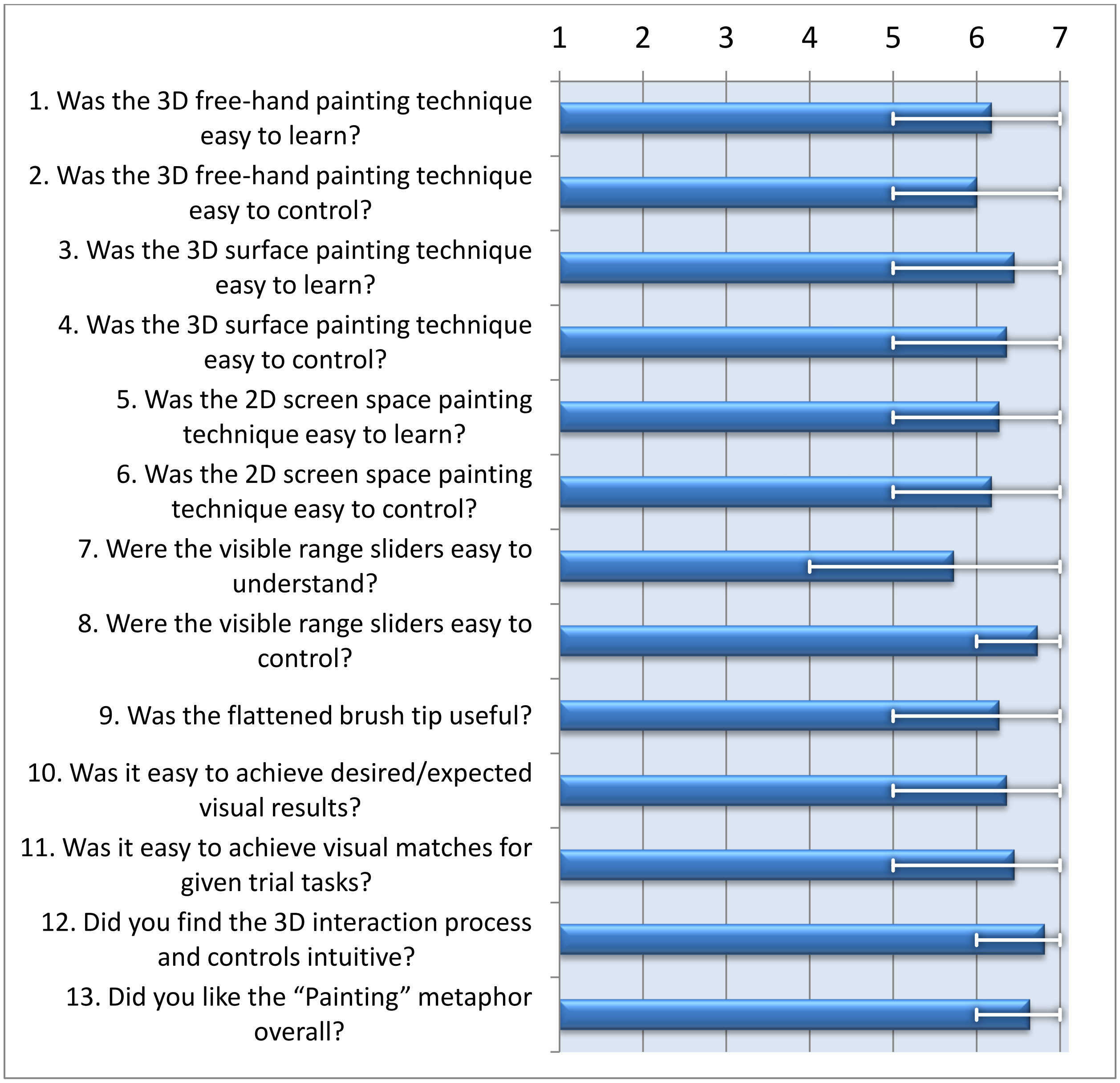


Figure 60: The results of the user study questionnaire based on a 7-point Likert scale.

As can be seen from the answers the results were quite positive. In all categories but one the mean average was 6.0 or more. Also as can be seen by the spread of the marks for each question, there was not much variability in the given answers. The only question that received a mean below 6.0 was related to the sliders that control the visible range of values under MIDA visualization mode. Perhaps this aspect of the interaction process can be improved by providing a single more user-friendly custom range control (instead of the two separate sliders), that would allow users to achieve the same visual results but with a fewer number of adjustments. Another common comment from participants was that even though gauging the depth of the brush tip in relation to volumetric structures in the scene was usually not a problem, under certain conditions it still could be somewhat confusing and thus they would prefer to have additional depth cues. This is a known issue in the current implementation and possible ways of addressing it will be discussed in Chapter 5:.

Overall, the results of the user study show that all of the participants liked the proposed visualization and exploration model and found it to be both intuitive and quite easy to learn and control.

4.3 Performance

As was stated previously, one of the characteristics of a satisfactory exploration experience can be expressed in terms of the *fluidity* of a user interaction with a virtual scene. That means that rendering frame-rates for any given visualized dataset should be maintained at interactive levels as not to appear choppy or unbearably slow, regardless of the types of operations and visualization modes. A number of 30 FPS was chosen as an acceptable target frame-rate. A total of four different 16 bit[32] per voxel datasets of varying sizes were chosen to measure the performance both with and without 2D and 3D envelopes. Different combination of rendering modes for cases with a defined envelope including MIDA only, MIDA and 1D TF, MIDA and

[32] Here 16 bit refers to the number of bits used to represent voxel intensity values in a dataset.

2D TF, and 1D and 2TF were also benchmarked. The results of the measurements expressed in average FPS for all combinations are presented in Figure 61.

It should be noted that measuring performance for different volumetric datasets, even if they have the same physical size and are visualized using the same visualization parameters, is not a straightforward process and can produce quite different results in terms of performance. This is due to the fact that the complexity of computations is heavily dependent on the nature of the data being processed and such aspects of the visualization process as color mapping, gradient evaluation, early ray termination, the number of values sampled along a ray, volume and lighting model interaction and many others can heavily influence the results. Thus to minimize the possible effects of the aforementioned conditions all of the tests were performed using the same resolution of the rendering window (which was left at the ImageVis3D default value of 400x400 pixels) and the same default visualization parameters for all the tested rendering modes (i.e. MIDA slider values were left in their default positions). Also, a 3D envelope (for the tests where the envelope was present) covering approximately half of the volume of a dataset was used.

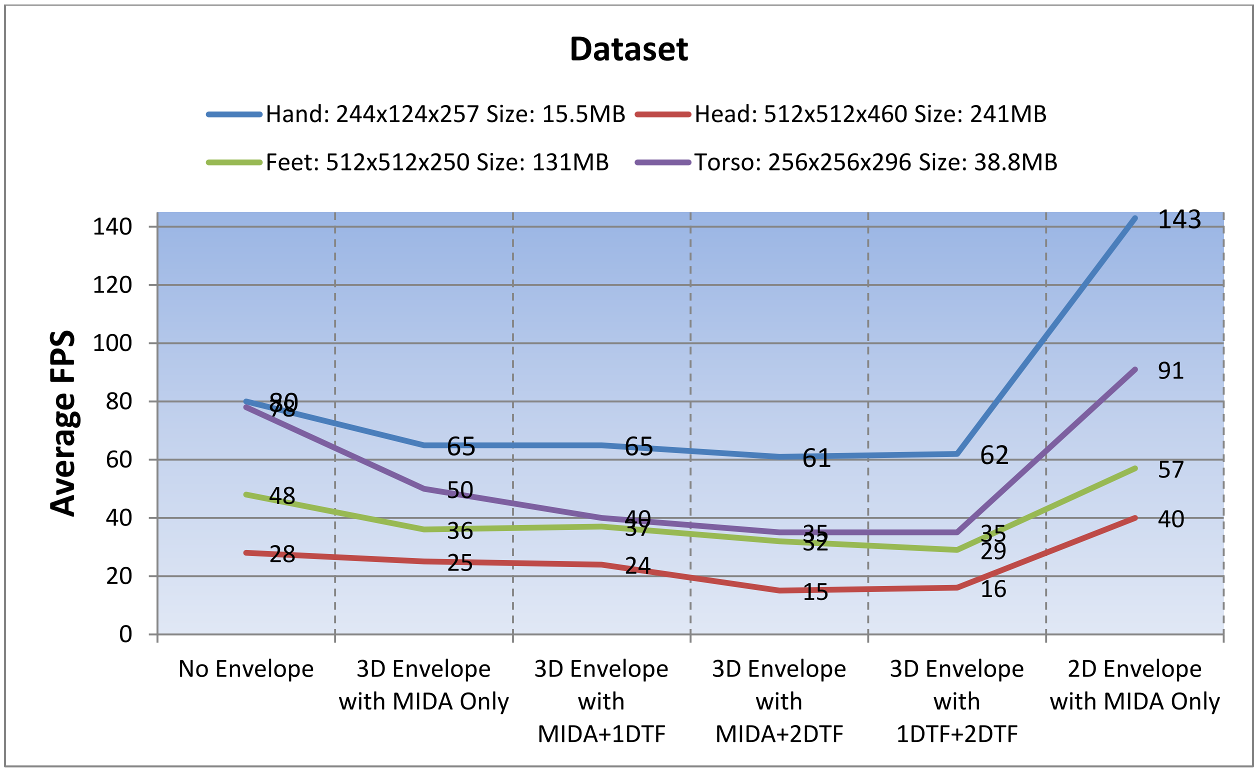


Figure 61: Average FPS measured under different visualization conditions.

As can be seen from the results, for very large datasets (i.e. greater than 200MB), under certain combinations of visualization techniques, the average frame-rate can drop quite considerably below the target value of 30 FPS. This is both due to the overhead of the depth peeling algorithm and the heavy computational load VR is putting on even a powerful GPU[33]. In the future these figures will be considerably improved by employing various optimization techniques (Section 5.2), such as *Per-Pixel Linked List* [71] for order-independent transparency rendering, *Empty Space Skipping* (ESS) [72], *Frame Temporal Coherence* [73] and others. Currently, if the rendering frame-rate is considered to be inadequate by the user, two options for increasing it are available (albeit at a price of losing some of the visual quality in the resulting images).

[33] All of the tests were performed using nVidia 480 GTX video card.

The user can adjust the sampling rate of a volumetric dataset, forcing the VR algorithm to skip some of the data during the ray traversal process. This can be done directly from within the ImageVis3D user interface allowing the user to choose the desired level of compromise between quality and performance. The visual results with respective frame-rates for different levels of the sampling rate are shown in Figure 62.

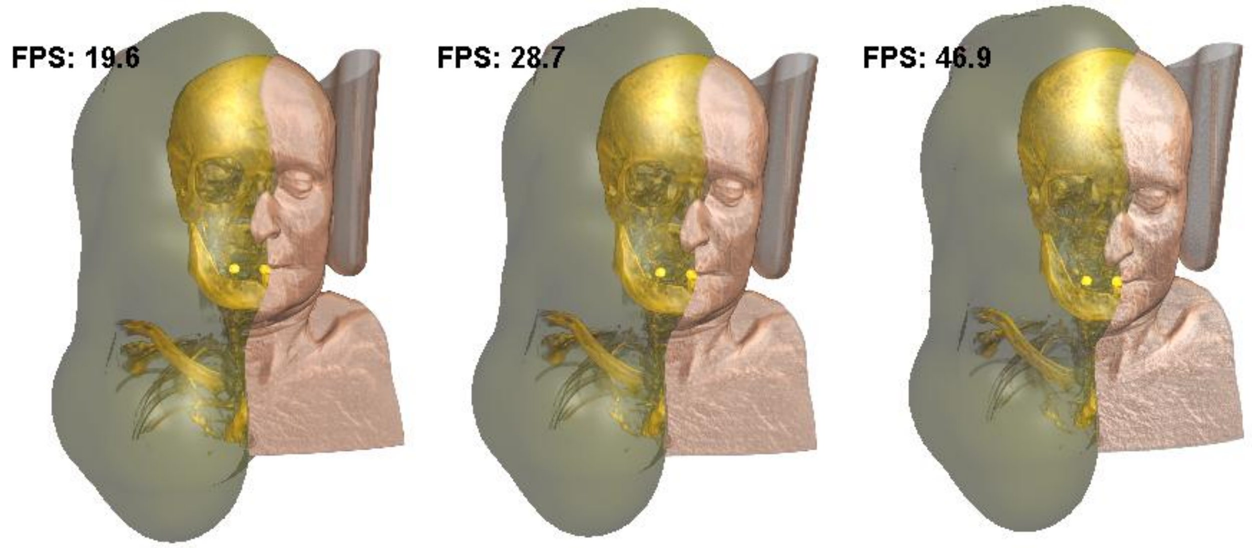


Figure 62: A head dataset visualized at (from left to right) 100%, 65% and 30% sampling rate.

Another available option allows the user to either force the desired LOD for the visualized volumetric dataset, or let the system choose it automatically by specifying the minimum allowed frame-rate. In the latter case the system will adjust the LOD on the fly by reducing the dimensions of the rendered dataset by half until the target frame-rate is reached. Examples of using differ LODs with corresponding frame-rates are presented in Figure 63.

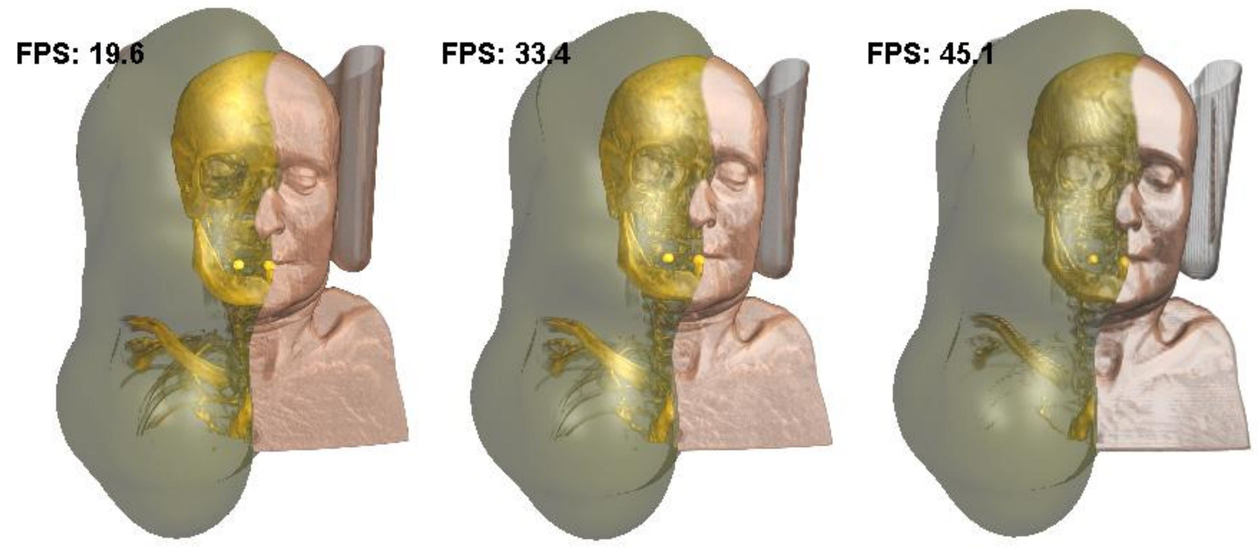
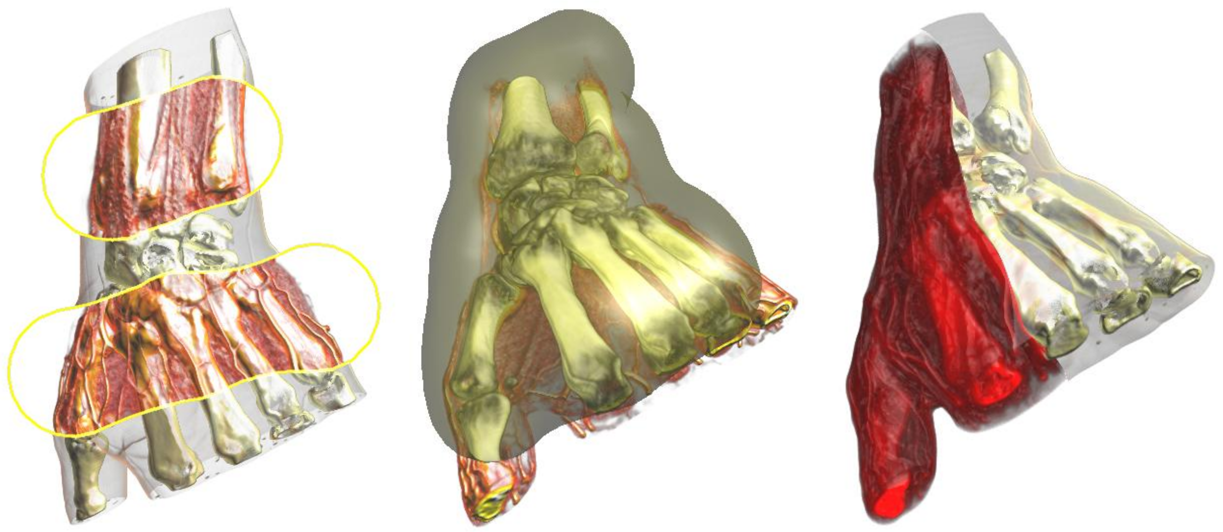


Figure 63: A head dataset visualized at different LODs (the highest on the left to the lowest on the right).

4.4 Extensibility

Despite its flexibility the MIDA visualization technique also has its limitations. For instance in datasets where feature separation based on intensity value ranges alone is not possible (e.g. due to similarity of intensities of different materials measured during a scanning process) MIDA produces poor visual results. Also, using just one base color for feature coloration can be somewhat restrictive. Realizing that, the implemented visualization and envelope definition techniques have been designed in such a way as to allow for seamless and straightforward integration of other volume visualization techniques (such as the ones described in Sections 2.1 and 2.2).

To demonstrate the extensibility potential of the proposed approach, two other VR techniques, 1D and 2D TF rendering modes, already implemented in ImageVis3D have been integrated into the code. The total time of the integration was about 3 hours. With this newly added functionality the user is now able to select MIDA, 1D TF or 2D TF as the rendering mode for parts of the volume both inside and outside of the envelope. Any combination of rendering modes is possible and the original functionality, including the 3D surface painting, is fully preserved. Figure 64 shows examples of combining different rendering modes as well as using both the 2D screen painting and the 3D surface and free-hand painting modes. The screenshot in Figure 64(a) depicts using the 2D painting mode with the parts of the volume inside of the envelope rendered using a 1D TF and the parts outside of the envelope using a 2D TF. The screenshot in Figure 64(b) uses results of the 1D TF rendering process as the basis for the 3D surface painting to define the 3D envelope enclosing features rendered using the MIDA technique. The last screenshot in Figure 64(c) shows a combination of the MIDA and 2D TF visualization modes and the use of the 3D free-hand painting technique.



(a) (b) (c)

Figure 64: (a) The 2D painting mode with 1D and 2D TF rendering modes. (a) An example of the 3D surface painting over the surface defined by a 1D TF. (c) A combination of MIDA and 2D VR techniques using a free-hand painted 3D envelope.

Chapter 5: Conclusion and Future work

5.1 Conclusion

This thesis presented an implementation of an alternative context-preserving volume image exploration model. The image exploration is realized by an interaction model based on a novel painting metaphor, where a user encloses the volumetric features of interest by painting a 2D or 3D envelope directly in the space of a virtual scene. Combined with the MIDA volumetric visualization mode, the envelopes allow the user to quickly achieve the desired context-preserving views by visualizing the envelope enclosed features in a distinctly different way from the rest of the volume. Furthermore, the ability to adjust the range of visible volume intensity values when using the MIDA visualization mode provides additional volume exploration flexibility by allowing the user to easily filter out the occluding volumetric features, as well as perform such operations as carving and peeling.

The proposed interaction model was implemented in the form of extensions to the existing volume rendering system ImageVis3D. The extensions were designed and implemented in such a way as to blend seamlessly with the rest of the ImageVis3D framework, provide easy extensibility in terms of possible usage of other rendering modes and support interactive frame-rates even for very large volumetric datasets. The results and contributions of this thesis have been evaluated in terms of flexibility of possible modes of volume exploration, performance and extensibility of visualization modes by carrying out a series of experiments presented in Chapter 5:. Finally, the results of the conducted user study validate such claimed qualities of the proposed approach as the fast learning curve, intuitiveness and ease of control.

5.2 Future work

Following one of the initially set objectives, the volume exploration paradigm and the painting metaphor implemented as a part of this thesis are quite generic. This allows taking the future development of this project in a number of directions.

One of the very popular and rapidly evolving areas in the field of VR, that has not been covered in this thesis, is *Volume Segmentation* [74]. This area is somewhat related to volume exploration in a sense that segmenting and labeling volumetric features allows for a much wider choice of ways of visualizing them. One popular approach to performing a user guided semi-automatic segmentation process is based on the idea of *Seed Growing* and *Edge Detection* [74]. There are quite a few techniques and mathematical models that try to address this issue, but, despite all the research, most of them still (especially when applied to noisy datasets) suffer from a problem known as *leakage*, where a segmentation process *spills* outside of the shape that it is meant to detect. One of the solutions to this problem is to restrict the segmentation process by enclosing it in some sort of a user adjustable envelope. Coincidentally, this is exactly what the proposed approach of enveloping the features of interest is designed to do, and combined with one of the segmentation techniques it could prove to be a useful constraining mechanism.

Another possible direction for future work is to concentrate on evolving the painting interaction model and envelope editing capabilities. Unlike parametric models that allow the user to define control points directly on the surface of a described 3D parametric model, implicit models are lacking this characteristic and are thus difficult to use when precise control over the shape of the modelled surface is required. This is however changing with the introduction of some novel techniques [5] that allow controlling the geometry of implicit surfaces in a manner similar to their parametric counterparts. Integrating such functionality into the project would allow it to become a much more precise and flexible exploration tool.

Furthermore, a few ways of improving and optimizing the current implementation can also be explored.

Firstly, as was already mentioned, additional depth cues could be used to improve spatial comprehension of the relative position of the envelope and volumetric features. This could be achieved in several ways, including, but not limited to, using optically correct self-casting shadows for all volumetric and polygonal objects present in the scene, providing additional views of the objects in the scene from a number of different viewpoints, or introducing some sort of a projection plane directly into the space of a 3D scene onto which the outlines of the objects in the scene will be cast.

Secondly, the rendering frame-rates could be improved considerably by employing various optimization techniques, such as ESS [72], temporal coherence [73], data compression and on-demand streaming [75], etc. The applicability of each of these techniques however has to be carefully considered and evaluated, as reconciling and marrying different algorithms in the same visualization pipeline can pose non-trivial problems. In fact, due to exactly these reasons, an ESS algorithm that had been implemented at the early stages of the project and provided on average a 40% boost in frame-rates, had to be "put on ice", as, in view of the project deadlines, it was not possible to integrate it properly with the rest of the system.

Finally, as new research is being done and new work is being published, some of the techniques presented in this thesis could be improved or replaced entirely by other more efficient versions.

For instance *Generalized Metaballs* [76] could be used to extend the project's implicit surface definition capabilities. As a bonus they provide more freedom in terms of choosing a suitable field function. An order independent FtBDP algorithm used in this thesis could be replaced by a recently published more efficient single-pass *Per-Pixel Linked Lists* technique [71]. This in turn will allow implementing a single-pass ray-casting algorithm with simultaneous peeling and blending, which should result in the frame-rates that are almost identical regardless of whether a 3D envelope is present and not. In addition, if implemented, this approach will open doors for various other polygon-based and volumetric rendering techniques, as all of the information (i.e. fragment coordinated, normal directions, color values, etc.) about all objects in the scene will be readily available through a single linked-list structure.

REFERENCES

[1] Opitz, A. (2009). *Classification and Visualization of Volume Data using Clustering.* Master Thesis, Technischen Universität Wien, Wien.

[2] ImageVis3D. (NIH/NCRR Center for Integrative Biomedical Computing). Retrieved November 24, 2011, from http://www.sci.utah.edu/cibc/software/41-imagevis3d.html

[3] Hayward, K. Volume Rendering 102: Transfer Functions. (Kyle Hayward). *Graphics Runner Blog.* Retrieved November 24, 2011, from http://graphicsrunner.blogspot.com/2009/01/volume-rendering-102-transfer-functions.html

[4] The community of Slicer developers. (2011). 3DSlicer (Version 4.0): Free open source software under BSD license. Retrieved November 24, 2011, from http://www.slicer.org/

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

[6] Hadwiger, M., Ljung, P., Salama, C. R., & Ropinski, T. (2008). *Advanced illumination techniques for GPU volume raycasting.* In proceedings of ACM SIGGRAPH Asia 2008 courses, Singapore, 1-166.

[7] Bloomenthal, J., Bajaj, C., Blinn, J., Cani-Gascuel, M.-P., Rockwood, A., Wyvill, B., et al. (August 15, 1997). *An Introduction to Implicit Surfaces* (1 ed.): Morgan Kaufmann.

[8] Monclus, E., Dıaz, J., Navazo, I., & Vazquez, P.-P. (2009). *The virtual magic lantern: an interaction metaphor for enhanced medical data inspection.* In proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, Kyoto, Japan, 119-122.

[9] Khronos Group. (2011). OpenGL (Version 4.2): Khronos Group. Retrieved November 24, 2011, from http://www.opengl.org/

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

[11] Zin, Y. C., Zheng, W., Chee, M. W. L., & Zagorodnov, V. (2009). Evaluation of performance metrics for bias field correction in MR brain images. *Journal of Magnetic Resonance Imaging, 29*(6), 1271-1279.

[12] Lum, E. B., & Ma, K. L. (2004). *Lighting transfer functions using gradient aligned sampling.* In proceedings of the 15th IEEE Conference on Visualization 2004 (VIS'04), Austin, TX, USA, 289-296.

[13] Sereda, P., Bartroli, A. V., Serlie, I. W. O., & Gerritsen, F. A. (2006). Visualization of boundaries in volumetric data sets using LH histograms. *IEEE Transactions on Visualization and Computer Graphics, 12*(2), 208-218.

[14] University of Münster and Linköping University. (2011). Voreen (Version 2.6.1): University of Münster and Linköping University. Retrieved November 24, 2011, from http://www.voreen.org/

[15] Pixmeo. OsiriX (Version 4.0): Pixmeo. Retrieved November 24, 2011, from http://www.osirix-viewer.com/

[16] Kindlmann, G., Whitaker, R., Tasdizen, T., & Moller, T. (2003). *Curvature-based transfer functions for direct volume rendering: methods and applications.* In proceedings of the 14th IEEE Conference on Visualization 2003 (VIS'03), Seattle, WA, USA, 513-520.

[17] Prassni, J. S., Ropinski, T., Mensmann, J., & Hinrichs, K. (2010). *Shape-based transfer functions for volume visualization.* In proceedings of Pacific Visualization Symposium 2010 (PacificVis '10), IEEE, Taipei, 9-16.

[18] Sato, Y., Westin, C., Bhalerao, A., Nakajima, S., Shiraga, N., Tamura, S., et al. (2000). Tissue classification based on 3D local intensity structures for volume rendering. *IEEE Transactions on Visualization and Computer Graphics, 6*(2), 160-180.

[19] Correa, C., & Kwan-Liu, M. (2008). Size-based Transfer Functions: A New Volume Exploration Technique. *IEEE Transactions on Visualization and Computer Graphics, 14*(6), 1380-1387.

[20] Caban, J. J., & Rheingans, P. (2008). Texture-based Transfer Functions for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics, 14*(6), 1364-1371.

[21] Patel, D., Haidacher, M., Balabanian, J. P., & Groller, E. M. (2009). *Moment curves.* In proceedings of Pacific Visualization Symposium 2009 (PacificVis '09), IEEE, Beijing, 201-208.

[22] Haidacher, M., Patel, D., Bruckner, S., Kanitsar, A., & Groller, M. E. (2010). *Volume visualization based on statistical transfer-function spaces.* In proceedings of Pacific Visualization Symposium 2010 (PacificVis '10), IEEE, Taipei, 17-24.

[23] Salama, C. R., Keller, M., & Kohlmann, P. (2006). High-Level User Interfaces for Transfer Function Design with Semantics. *IEEE Transactions on Visualization and Computer Graphics, 12*(5), 1021-1028.

[24] Haidacher, M., Bruckner, S., Kanitsar, A., & Gröller, M. E. (2008). Information-based Transfer Functions for Multimodal Visualization. *IEEE Transactions on Visualization and Computer Graphics, 14*(6), 1380-1387.

[25] Tzeng, F.-Y., & Ma, K.-L. (2004). *A cluster-space visual interface for arbitrary dimensional classification of volume data.* In proceedings of Eurographics VGTC Symposium on Visualization 2004 (VisSim '04), IEEE, Norköping, Sweden, 17–24.

[26] Sereda, P., Vilanova, A., & Gerritsen, F. A. (2006). *Automating transfer function design for volume rendering using hierarchical clustering of material boundaries.* In proceedings of Eurographics VGTC Symposium on Visualization 2006 (VisSim '06), IEEE, Nicosia, Cyprus, 243–250.

[27] Heidrich, W., McCool, M., & Stevens, J. (1995, 29 Oct-3 Nov 1995). *Interactive maximum projection volume rendering.* In proceedings of the 6th IEEE Conference on Visualization 1995 (VIS '95), Atlanta, GA , USA, 11-18, 433.

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

[29] Bruckner, S. (2006). *Interactive illustrative volume visualization techniques for exploration and communication.* In proceedings of ACM SIGGRAPH 2006 Courses, Boston, MA. USA, 6.

[30] Bruckner, S., & Gröller, M. E. (2007). Style Transfer Functions for Illustrative Volume Rendering. *Computer Graphics Forum, 26*(3), 715-724.

[31] Diepstraten, J., Weiskopf, D., & Ertl, T. (2002). Transparency in Interactive Technical Illustrations. *Computer Graphics Forum, 21*(3), 317-325.

[32] Bruckner, S., Grimm, S., Kanitsar, A., & Groller, M. E. (2006). Illustrative Context-Preserving Exploration of Volume Data. *IEEE Transactions on Visualization and Computer Graphics, 12*(6), 1559-1569.

[33] Bruckner, S., Rautek, P., Viola, I., Roberts, M., Sousa, M. C., & Gröller, M. E. (2010). Hybrid Visibility Compositing and Masking for Illustrative Rendering. *Computers & Graphics, 34*(4), 361-369.

[34] Wang, L., Zhao, Y., Mueller, K., & Kaufman, A. (2005, 23-28 Oct. 2005). *The magic volume lens: an interactive focus+context technique for volume rendering.* In proceedings of the 16th IEEE Conference on Visualization 2005 (VIS'05), Baltimore, MD, USA, 367-374.

[35] Taerum, T., Sousa, M. C., Samavati, F. F., Chan, S., & Mitchell, J. R. (2006). *Real-Time Super Resolution Contextual Close-up of Clinical Volumetric Data.* In proceedings of EuroVis'06, Lisbon, Portugal, 347-354.

[36] Luo, Y. (2011). The Distance-based Focus+Context Models for Exploration of Large Volumetric Medical Datasets. *Computing in Science & Engineering, PP*(99), 1-1.

[37] Bruckner, S., & Groller, M. E. (2006). Exploded Views for Volume Data. *IEEE Transactions on Visualization and Computer Graphics, 12*(5), 1077-1084.

[38] Ruiz, M., Viola, I., Boada, I., Bruckner, S., Feixas, M., & Sbert, M. (2008). Similarity-Based Exploded Views. In Andreas Butz, Brian Fisher, Antonio Krüger, Patrick Olivier & Marc Christie (Eds.), *Smart Graphics* (Vol. 5166, pp. 154-165): Springer Berlin / Heidelberg.

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

[40] Bernhard, O. K.-V., Preim, B., & Littmann, A. (2004). *Virtual Resection with a Deformable Cutting Plane.* In proceedings of Simulation und Visualisierung 2004 (SimVis'04), Magdeburg, Germany, 203-214.

[41] Bruckner, S., & Groller, M. E. (2005). *VolumeShop: an interactive system for direct volume illustration.* In proceedings of the 16th IEEE Conference on Visualization 2005 (VIS'05), Baltimore, MD, USA, 671-678.

[42] Chen, H.-L. J., Samavati, F. F., & Sousa, M. C. (2008). GPU-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer, 24*(7), 689-698.

[43] McGuffin, M. J., Tancau, L., & Balakrishnan, R. (2003). *Using deformations for browsing volumetric data.* In proceedings of the 14th IEEE Conference on Visualization 2003 (VIS'03), Seattle, WA, USA, 401-408.

[44] Correa, C. D., Silver, D., & Chen, M. (2006). Feature Aligned Volume Manipulation for Illustration and Visualization. *IEEE Transactions on Visualization and Computer Graphics, 12*(5), 1069-1076.

[45] Correa, C. D., Silver, D., & Min, C. (2007). Illustrative Deformation for Data Exploration. *IEEE Transactions on Visualization and Computer Graphics, 13*(6), 1320-1327.

[46] Correa, C. D., Silver, D., & Chen, M. (2010). Constrained illustrative volume deformation. *Computers and Graphics (Pergamon), 34*(4), 370-377.

[47] Christie, M., Olivier, P., & Normand, J.-M. (2008). Camera Control in Computer Graphics. *Computer Graphics Forum, 27*(8), 2197-2218.

[48] Bade, R., Ritter, F., & Preim, B. (2005). *Usability comparison of mouse-based interaction techniques for predictable 3d rotation.* In proceedings of Smart Graphics 2005, Frauenwoerth Cloister, Germany, 138-150.

[49] Schmidt, R., Singh, K., & Balakrishnan, R. (2008). Sketching and composing widgets for 3D manipulation. *Computer Graphics Forum, 27*(2), 301-310.

[50] MeVis Medical Solutions AG and Fraunhofer MEVIS. (2011). MeVisLab (Version 2.2.1): MeVis Medical Solutions AG and Fraunhofer MEVIS. Retrieved November 24, 2011, from http://www.mevislab.de/

[51] The Institute of Computer Graphics and Algorithms. (2011). VolumeShop (Version 1.7): The Institute of Computer Graphics and Algorithms. Retrieved November 24, 2011, from http://www.cg.tuwien.ac.at/volumeshop/download/

[52] Scharsach, H., Hadwiger, M., Neubauer, A., Wolfsberger, S., & Bühler, K. (2006). *Perspective Isosurface and Direct Volume Rendering for Virtual Endoscopy Applications.* In proceedings of Eurographics VGTC Symposium on Visualization (VisSim '06), IEEE, Nicosia, Cyprus, 315-322.

[53] Haigron, P., Bellemare, M. E., Acosta, O., Goksu, C., Kulik, C., Rioual, K., et al. (2004). Depth-map-based scene analysis for active navigation in virtual angioscopy. *IEEE Transactions on Medical Imaging, 23*(11), 1380-1390.

[54] Hong, L., Muraki, S., Kaufman, A., Bartz, D., & He, T. (1997). *Virtual voyage: Interactive navigation in the human colon.* In proceedings of SIGGRAPH 1997 Los Angeles, CA, USA, 27-34.

[55] Kruger, A., Kubisch, C., Strauss, G., & Preim, B. (2008). Sinus Endoscopy - Application of Advanced GPU Volume Rendering for Virtual Endoscopy. *IEEE Transactions on Visualization and Computer Graphics, 14*(6), 1491-1498.

[56] McClymont, J., Shuralyov, D., & Stuerzlinger, W. (2011). *Comparison of 3D navigation interfaces.* In proceedings of IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems 2011 (VECIMS'11), Ottawa, ON, Canada, 7-12.

[57] Diepenbrock, S., Ropinski, T., & Hinrichs, K. (2011, 1-4 March 2011). *Context-aware volume navigation.* In proceedings of Pacific Visualization Symposium 2011 (PacificVis '11), IEEE, Hong Kong, China, 11-18.

[58] Visage Imaging. Amira (Version 5.4.1): Visage Imaging. Retrieved November 24, 2011, from http://www.amira.com/

[59] Boost Community. Boost C++ Libraries (Version 1.48.0): Free open source software under Boost license. Retrieved November 24, 2011, from http://www.boost.org/

[60] Nokia. Qt (Version 4.7.4): Nokia. Retrieved November 24, 2011, from http://qt.nokia.com/products/

[61] Foley, J. D., Dam, A.V., Feiner S.K., Hughes J.F. (Ed.). (August 14, 1995). *Computer Graphics: Principles and Practice in C* (2 ed.): Massachusetts: Addison-Wesley Publishing Company.

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

[63] Çelebi, O. C. Scientific Visualization and 3D Volume Rendering (Volume Rendering Tutorial). Retrieved November 24, 2011, from http://www.celebisoftware.com/Tutorials/volume_rendering/Index.aspx

[64] Everitt, C. (2001). Interactive order-independent transparency (Technical Report): NVIDIA Corporation.

[65] Bavoil, L., & Myers, K. (2008). Order Independent Transparency with Dual Depth Peeling (Technical Report): NVIDIA Corporation.

[66] Wyvill, G., McPheeters, C., & Wyvill, B. (1986). Data structure for soft objects. *The Visual Computer, 2*(4), 227-234.

[67] Blinn, J. F. (1982). Generalization of Algebraic Surface Drawing. *Computer Graphics (ACM), 16*(3), 273.

[68] Lorensen, W. E., & Cline, H. E. (1987). MARCHING CUBES: A HIGH RESOLUTION 3D SURFACE CONSTRUCTION ALGORITHM. *Computer Graphics (ACM), 21*(4), 163-169.

[69] Chan, S. L., & Purisima, E. O. (1998). A new tetrahedral tesselation scheme for isosurface generation. *Computers and Graphics (Pergamon), 22*(1), 83-90.

[70] Nguyen, H. (September 12, 2007). *GPU Gems 3* (1 ed.): Addison-Wesley Professional.

[71] Yang, J. C., Hensley, J., Grün, H., & Thibieroz, N. (2010). Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum, 29*(4), 1297-1304.

[72] Zou, H., Gao, X., & Lv, X. (2008). An accelerating algorithm for 3D texture volume rendering with octree encoding. *Journal of Xi'an Jiaotong University, 42*(12), 1490-1494.

[73] Scherzer, D. (2009). *Applications of temporal coherence in real-time rendering.* PhD-Thesis, Technischen Universität Wien, Wien.

[74] Pham, D. L., Xu, C., & Prince, J. L. (2000). Current Methods in Medical Image Segmentation. *Annual Review of Biomedical Engineering, 2*(1), 315-337.

[75] Crassin, C., Neyret, F., Lefebvre, S., & Eisemann, E. (2009). *GigaVoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Rendering.* In proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2009 (I3D'09), Boston, MA, USA, 15-22.

[76] Jin, X., Li, Y., & Peng, Q. (2000). General constrained deformations based on generalized metaballs. *Computers and Graphics (Pergamon), 24*(2), 219-231.