

View-Dependent Peel-Away Visualization for Volumetric Data

Åsmund Birkeland

November 2008



Master Degree Thesis

Department of Informatics
University of Bergen

Master Degree Thesis

View-Dependent Peel-Away Visualization for Volumetric Data

by Åsmund Rognerud Birkeland
November 2008



Supervised by Ivan Viola

Visualization Group
Department of Informatics
University of Bergen

<http://www.iu.uib.no/vis/teaching/thesis/2008-birkeland/>

Abstract

Traditional illustration of three-dimensional structures, has developed techniques to provide clear view on internal features that are otherwise hidden underneath other outer structures. Many techniques attempt to create a better view on features of interest, while still conveying information about surrounding contextual structures. In illustrative volumetric visualization, techniques developed in early illustrations have been adopted to increase visibility and improve the comprehension of volumetric datasets during interactive visualizations.

In this thesis a novel approach for peel-away visualization is presented. Newly developed algorithm extends existing illustrative deformation approaches which are based on deformation templates and adds a new factor of view-dependency to the peeling process. View-dependent property guarantees the viewer unobstructed view to structure of interest. This is realized by rotating the peel-template so that the structures peeled-away always face away from the viewer. Furthermore the new algorithm computes the underlying peel-template on-the-fly, which allows animating the level of peeling. This is very helpful for understanding the original spatial arrangement. When structures of interest are tagged with segmentation masks, an automatic scaling and positioning of peel deformation templates allows guided navigation and clear view at structures in focus as well as feature-aligned peeling. The overall performance allows smooth interaction with reasonably sized datasets and peel templates as the implementation maximizes utilization of computation power of modern GPUs.

Contents

1	Introduction	1
1.1	Technical Illustrations	1
1.2	Explorative Visualization	3
1.3	Illustrative Visualization	5
1.4	Occlusion in Three Dimensional Data	5
1.5	Goals of this Thesis	6
2	State of the Art	7
2.1	Slice Rendering	8
2.2	Semi-Transparency and Contours	9
2.3	Clipping and Cut-Aways	13
2.4	Splitting	15
2.5	Deformation	17
3	View-Dependent Peel-Away in Direct Volume Rendering	25
3.1	Visualization Pipeline	27
3.1.1	Peel-Template Setup	27
3.1.2	Ray-Casting through Vector Fields	30
3.2	Interaction	33
3.2.1	Peel-Template Positioning and Scaling	34
3.2.2	Automatic Placement Orientation and Scaling	35
4	Implementation	39
4.1	Building the Peel-Template	40
4.2	Template-Aware Ray-Casting	42
4.3	Feature Projection	46
4.4	Interaction	46
5	Results	49
5.1	Peel-aways for Exploration	52
5.2	Peel-aways for Presentation	53

5.3	Performance	54
6	Summary and Conclusions	57
6.1	Conclusions	62
6.2	Future Work	63

Chapter 1

Introduction

This work is in the research field of Illustrative Scientific Visualization. The main purpose is applying methods inspired by illustrators for describing three dimensional objects, to the field of scientific volumetric visualization. Visual representations convey information to the observer very quickly. Illustrators have provided visual representations in various fields since the stone age. Primitive men carved illustrations of animals on the cave walls roughly 75,000 years ago describing for instance hunting methods so that future generations could learn. Although the profession as an illustrator has changed drastically since then, the basic purpose remains the same: convey information in a way that is easily perceived by the human brain. Illustrators have used techniques for giving informative representations of real world structures, but the traditional illustrations had a very limited degree of interaction. Johann Remmelin published *Catoptrum Microcosmicum* in 1613, an extensive anatomical description of the human body. Remmelin was one of the first to incorporate peels as a means of interaction in medical illustrations. In his illustrations one could remove each layer showing a hidden structure underneath, this is shown in Figure 1.1.

Scientific data is only getting larger as data acquisition techniques are becoming more advanced. In the field of scientific visualization the use of illustrative methods such as peel-aways and cut-aways are useful for presentation of scientific datasets.

1.1 Technical Illustrations

Creating illustrations of three dimensional structures such as the human body has been done for a long time. Leonardo da Vinci started with anatomical studies to improve his

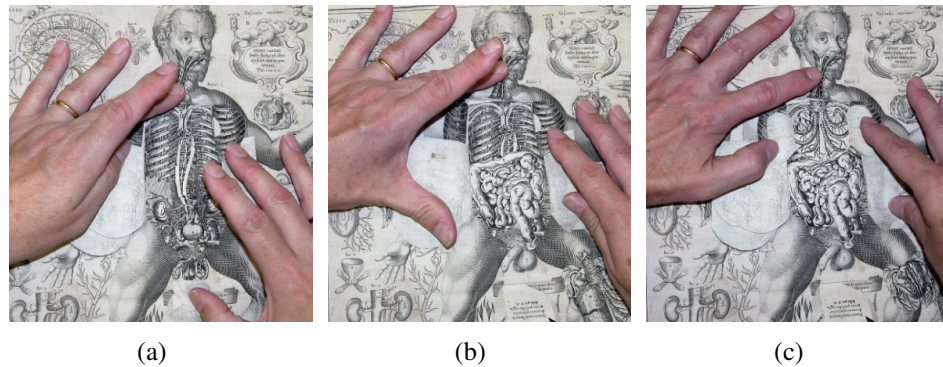


Figure 1.1: Johann Remmelin; *Catoptrum Microscopicum*. 1613, Hardin Library. An interactive illustration of the human body.

artistic skill [15]. Human dissection was illegal at that time, so illustrations of the human anatomy was not very common. Even though Vinci had a very limited time with the cadavers, he was able to create artistic representations of the human body as one of the first medical illustrators. Figure 1.2 shows an early illustration revealing a baby within the womb. Andreas Vesalius published his work *De Human Corporis Fabrica* in 1542 [35]. Containing more than 300 illustrations of the human body based on observing dissections of corpses. Da Vinci and Vesalius were pioneers in medical illustrators, and their illustrative methods are still being incorporated in today's visualization systems

There are several restrictions an illustrator must consider when he or she wishes to create a representation of a physical object. One can not assume that the reader has complete knowledge of the given object. The illustrations are also limited to a two dimensional surface for representing a three dimensional object. Thus one can not give a complete image of the real world structure. The illustrator has to focus the attention on the relevant information and leave out the irrelevant. Illustrating 3D structures leaves a great challenge with showing features hidden by occluding structures while still keeping a clear perception of the context. If one is given a layered structure, the intuitive method of exploring is to peel away each layer to discover what is underneath. Already in the renaissance, Andreas Vesalius illustrated how peels gives a better overview as more of the context can be drawn while not distorting the image. In his illustrations the human body where shown in different layers in different images. When moving from one layer to the next, some of the occluding structures from the previous illustration were drawn as a peel, as shown in Figure 1.3. When a physician performed the dissection he would



Figure 1.2: Leonardo da Vinci; "The babe in the womb". 1511.

then retract and peel of layer of skin, fat and muscle to see the inner structures. This is represented in Vesalius's illustrations and it gives a better perception of the context rather than removing the occluding structures from the image.

1.2 Explorative Visualization

While the need for illustrations of three dimensional structures is present, new technology has made it possible to record more or less all the information of a 3D structure. There are several techniques to acquire such data sets. Computed tomography (CT), nuclear magnetic resonance imaging (MRI) or positron emission tomography (PET) have the ability to create scalar fields representing the scanned object. There are also other techniques such as simulation and modeling to acquire volumetric data sets, however the challenges with visualizing synthetic data sets as compared to measured data sets remains the same. Volumetric data sets, later referred to as volumes, contain a large amount of information. While there are several types of acquisition for such data sets it has certain common properties. A volume contains a three dimensional structure of one or more attributes. A volume often consists of a single three dimensional scalar

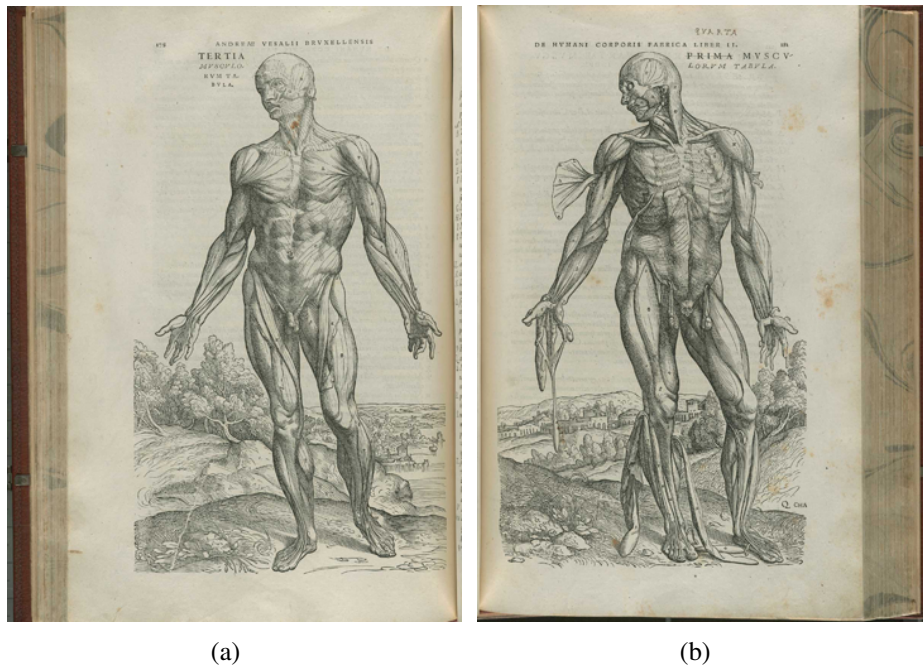


Figure 1.3: Andreas Vesalius; *De Humani Corporis Fabrica*. 1543. An interactive illustration of the human body. The image (b) show certain structures from image (a) peeled away.

field, for instance a CT scan. The size of a volume is determined by its resolution and its components. The resolution of a CT is usually higher than 256^3 . While a CT only contains a single scalar value (usually of 12 bit), the size of the volume become greater than $256 \times 256 \times 256 \times 12\text{bit} = 25165824\text{Bytes}$. This is too much data to show in a single image. Techniques for altering the representation and exploring the volume must be used to gain insight into the data.

Traditional medical illustrations are usually a description of the human anatomy in the general. A dataset from a CT scan, will contain information specific for the patient. A physician needs to explore a CT data set to be able to discover anomalies and diseases. An oil field is hidden underneath the earth and exploration of the seismic data is required to locate it. There are many different approaches in exploring a volume. Whether it is seismic data, MRI, simulation or modeling a very common factor is that the interesting features are hidden underneath the surface of the scanned object.

1.3 Illustrative Visualization

”Illustrative visualization is computer supported interactive and expressive visual abstraction motivated by traditional illustrations” [29, and references therein]. In essence the field of illustrative visualization strive to implement techniques for illustrating data, such as CT scans, inspired by the tools created in traditional illustration. These tools have been designed for conveying information about interesting features and their context efficiently. When methods from illustrative visualization are applied to scientific data, one must be careful not to distort the information. For instance, small objects may be excluded in non-photorealistic rendering and this might be crucial in a cancer diagnosis, as small lesions may be go undetected. A level of abstraction is required due to the large amount of data. So in the use of abstract illustrative methods, one must consider the aspect of information distortion. Non-photorealistic rendering must be used with care so that the information contained in the data is not lost. Different abstraction methods deal with different types of structure. For instance enhanced shading might be applied for better perception of the feature in focus, and semi-transparency might be applied to render the context while the focus is shown.

1.4 Occlusion in Three Dimensional Data

There are several techniques for visualizing volumes, one of the more basic rendering methods are two dimensional slice rendering [34, and references therein]. Slice rendering is a simple technique which requires little processing power, and has been one of the main methods for volume exploration. 3D rendering has recently become more frequently used as a means to display a volume. Techniques, such as ray-casting, first proposed by Levoy [23], and iso surface extraction, for instance using the marching cubes algorithm introduced by Lorensen and Cline [24].

Generating a 3D representation of the data and can provide a better overview of the volume compared to the more basic techniques such as slice rendering. When 3D representations are generated, more processing power is required to achieve interactive framerates. Today, with the use of hardware based rendering techniques, interactive rendering can be achieved on a regular workstation.

While both direct volume rendering and polygonal mesh rendering of iso surfaces

can create a good overview of the data, focusing on occluded features can become difficult using the basic 3D rendering techniques. Since the features of interest in a volume are often hidden by other structures, occlusion becomes a central issue in volume visualization. Even though the tools for creating images are different, both illustrators from the renaissance and visualization researchers today face the same problem: illustrating occluded structures hidden underneath the surface while still providing context for the structure in focus.

There are many existing techniques for dealing with occlusion, a common technique is reducing the opacity of the structures in front. In rendering techniques, such as ray casting, mapping color and opacity to each element in the volume, from now on referred to as voxels, is essential. The mapping from intensity to color and opacity can be done with a transfer function. Transfer functions can be difficult to handle by a non-expert user and hidden structures might be hard to reveal as the occluding matter might have the same value as the interesting features. In Section 2, different techniques which deal with occluding structures in 3D volume rendering will be discussed.

1.5 Goals of this Thesis

The goal of this thesis is to provide mechanisms for volume exploration and visual communication using peel-aways. Peel-aways have been commonly been used in both traditional illustration and illustrative visualization. The approach, which will be presented, applies the peel-away metaphor in direct volume rendering to efficiently explore and present features within a volume. This work aspire to extend peel-aways so that we incorporate the view-point as an important factor. Also an intuitive user interface for positioning and scaling peels will be presented. For presentation of features, automatic scaling and positioning of the peel will be implemented. This work will then be integrated and presented in a framework for volumetric rendering created by Bruckner called VolumeShop [3].

Chapter 2

State of the Art in Volume Exploration And Presentation

Illustrators have been handling occlusion of inner structures since the renaissance. Today computerized visualization can be applied to several types of data. This suggests that generic methods for exploring and presenting volumetric data are desired. Volumetric visualization can be divided into two major areas; Two dimensional slice rendering and 3D volume rendering. These two types contain different challenges, but the goal remains the same. Provide better insight in the data.

For 2D slice rendering, the information is easily represented on a 2D screen. However, a major challenge is to provide a good overview of the surrounding structures. 3D rendering of volumetric data can provide a good overview, but focusing on certain features might become more challenging. The structure in volumetric data can often be described as convex layered structures. The outer layers often occlude the inner structures which might be the feature of interest. For instance the human head consists of skin being the outer layer covering the skull, which in turn covers the brain. Thus occlusion becomes a major topic in volumetric rendering.

In this chapter we will discuss methods which deal with challenges for both 2D slice rendering and 3D rendering. Since this thesis is in the scope of volumetric visualization, the main focus in this chapter will be on occlusion handling in 3D volume rendering.

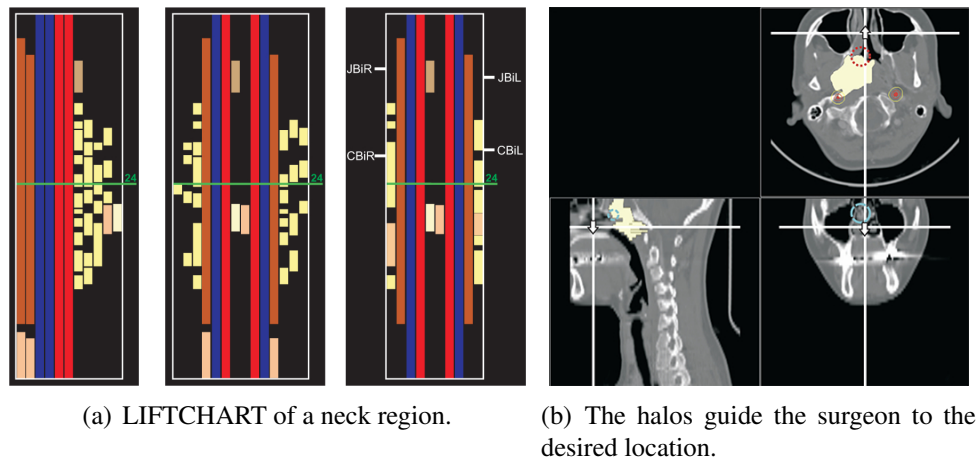
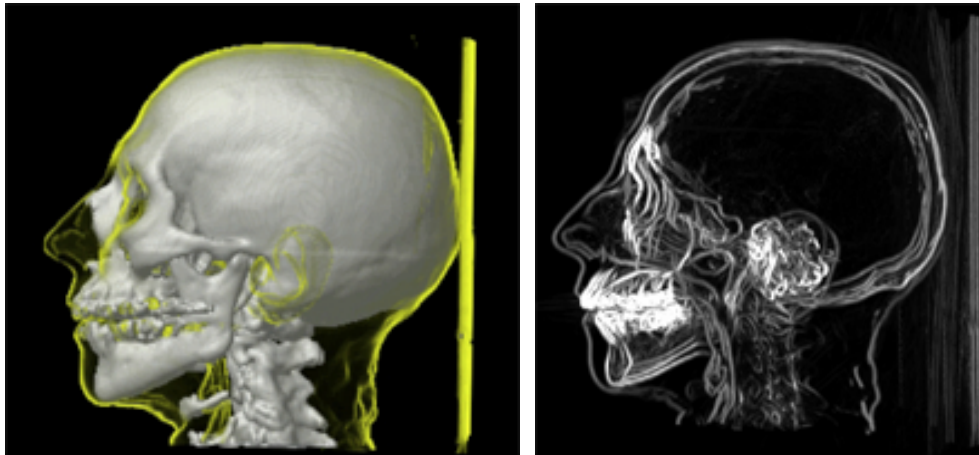


Figure 2.1: Images from *Enhancing Slice-based Visualizations of Medical Volume Data* [34].

2.1 Slice Rendering

For medical visualization, 2D slice rendering is a very important topic. Most medical visualization frameworks have a slice renderer, it is one of the main rendering methods for medical diagnosis. The basis for 2D slice rendering is a plane which intersect with the volume and the values in the volume at the intersection are rendered on the plane. A common slice rendering scenario have three slices which are orthogonally aligned with the x , y and z axis. To explore the volume one would control the depth of the plane within the volume. Another scenario might be to have a plane at an arbitrary angle, the user would then control the angle and the depth of the plane. A slice at an arbitrary angle might be more difficult to read, but it offers a greater flexibility.

The basic slice renderer only shows the current slice and determining the extent of certain structure might be difficult. Due to the low computational cost of rendering a 2D plane, several planes might be rendered simultaneously and might create a better overview. Structures which are not on either plane are not rendered and in a medical scenario this might be important for diagnosis or pre-operative planning. In the scenario of a neck dissection, the surgeon must consider a lot of structures and the spatial relationship between them. This is difficult on a basic slice renderer. A better overview of all the structures within the neck can help the surgeon to make the surgery as little evasive as possible.



(a) MIP rendering of the intensity function $I(\mathbf{p}, \mathbf{V})$. (b) Rendering of the contours of the skin with the skull underneath.

Figure 2.2: Images from *Fast Visualization of Object Contours by Non-Photorealistic Volume Rendering* [10].

Tietjen et al. introduced a novel approach called LIFTCHART [34] for enhancing slice-based visualization in medical data sets. LIFTCHART, see Figure 2.1(a), shows the anatomical structures in separate bars according to their position. This provides a better overview and helps localization of the features within the data. Tietjen et al. also introduced halos which were used within the rendering to provide a better perception of the distance from the current slice to targets above or below the current position, as shown in Figure 2.1(b).

2.2 Semi-Transparency and Contours

When rendering 2D slices, the structures in front of the plane is not included in the image. However, when generating 3D representation of a volume, the outermost surface will cover the inner structures. For instance, in a CT scan of a patient with a bone fracture, there is a need to see the bone underneath the skin. An approach is to make everything except the bone transparent. The spatial relationship between the feature of interest and the outer surface, in the latter case the bone and the skin, can be crucial in pre-operative planning, thus context preservation is an important topic in 3D volume rendering.

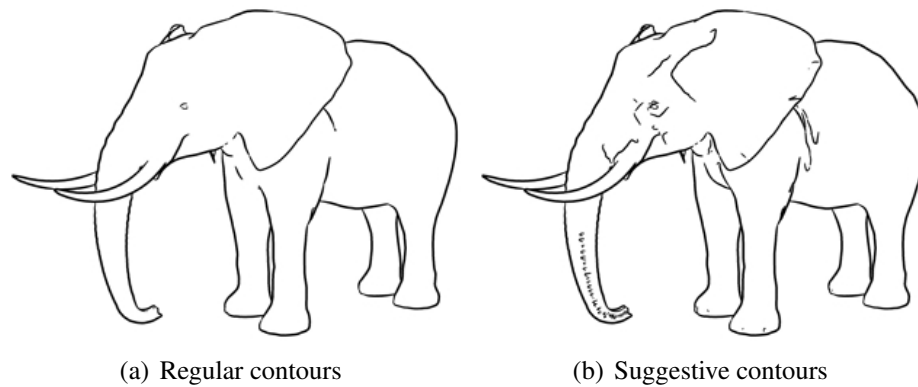


Figure 2.3: Images from *Suggestive Contours for Conveying Shape* [11].

Non-photorealistic rendering has been used to simplify the surfaces which are of low interest. Contours are a common tool for dealing with occluding surfaces and is a very sparse and effective representation of 3D structures. This technique is often used in traditional illustration. The contours of an object can be defined as where the normal of a surface is orthogonal to the viewing direction. Contours describe the basic outline of a surface while keeping occlusion to a minimum. A novel technique for fast contour rendering was introduced by Csébfalvi et al. [10]. The contours in volume rendering can be calculated from the normalized gradient of the sampling function and the view-direction. Csébfalvi et al. used the following functions: $s(\mathbf{p}, \mathbf{V}) = (1 - |\nabla(\mathbf{p}) \cdot \mathbf{V}|)^n$, where \mathbf{p} is any give position within the volume, and \mathbf{V} is the view direction. This provides a view-dependent function which assigns higher values to voxels which are on a contour of the surface. An intensity function is then given as: $I(\mathbf{p}, \mathbf{V}) = g(|\nabla(\mathbf{p})|) \cdot s(\mathbf{p}, V)$. The function g is a function depending on the magnitude of the gradient at position \mathbf{P} . Having this intensity function sampling in the renderer is done according to $I(\mathbf{p}, \mathbf{V})$, as shown in Figure 2.2. Using few control parameters, contours of inner and outer structures becomes clearly visible.

Other techniques like *Suggestive Contours*, presented by Doug Decarlo et al. [11] has been suggested for enhancing the shape perception. The suggestive contours are the locations which almost defines a true contour, true contour being where the dot product between the normal and the view direction are 0. They also correspond to the true contours calculated from a nearby view point. The suggestive contours are combined with the true contours to generate the final image. The difference between

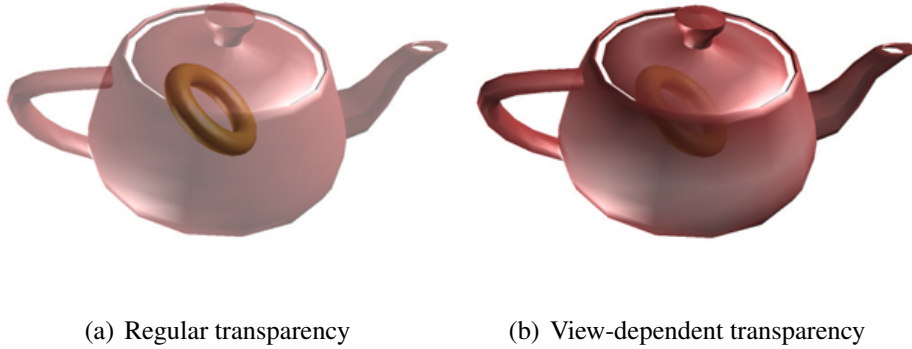


Figure 2.4: Images from *Transparency in Interactive Technical Illustrations* [12].

regular contours and suggestive contours are illustrated in Figure 2.3.

While contours convey shape information, some of the information of the context are lost. Semi-transparency is also a commonly used tool for occlusion handling and presenting hidden structures. Compared to contours which makes most of the surface completely transparent, semi-transparency show the structure directly in front of the feature of interest. Diepraten et al. presented a novel approach, *Transparency in Interactive Technical Illustrations* [12], for automatic view-dependent semi-transparency, as shown in Figure 2.4. The opacity at a position \mathbf{p} on the semi-transparent object is calculated as the $\alpha = 1 - \left(\frac{d_{min}}{d_{object,max}}\right)^k$, where d_{min} is the minimum distance from the \mathbf{p} to the projected silhouette of the semi-transparent object and $d_{object,max}$ is the maximum distance from the center of the object to the surface.

Illustrative context preserving illustrative volume rendering [2], inspired by ghosting from traditional artistic illustrations, is a novel approach to volume exploration. The opacity, α , for each sample point is calculated from the following function $\alpha_i(p) = \alpha(f_{p_i}) \cdot |g_{p_i}|^{(k_i \cdot s(p_i) \cdot (1 - |p_i - e|) \cdot (1 - \alpha_{i-1}))^k}$, where $1 - |p_i - e|$ is the distance from the eye to the sample, $|g_{p_i}|$ is the gradient magnitude, $s(p_i)$ a shading intensity function. and two user controlled factors. A result of this approach can be seen in Figure 2.5. This techniques is also related to clipping and cutting described in section 2.3.

Transfer functions are often used to generate semi-transparent surfaces. As mentioned in Section 1.4, transfer functions map color and opacity for each value within the volume but can be difficult to define. More sophisticated transfer functions has been developed. Bruckner and Gröller introduced *Style Transfer Functions for Illustrative*

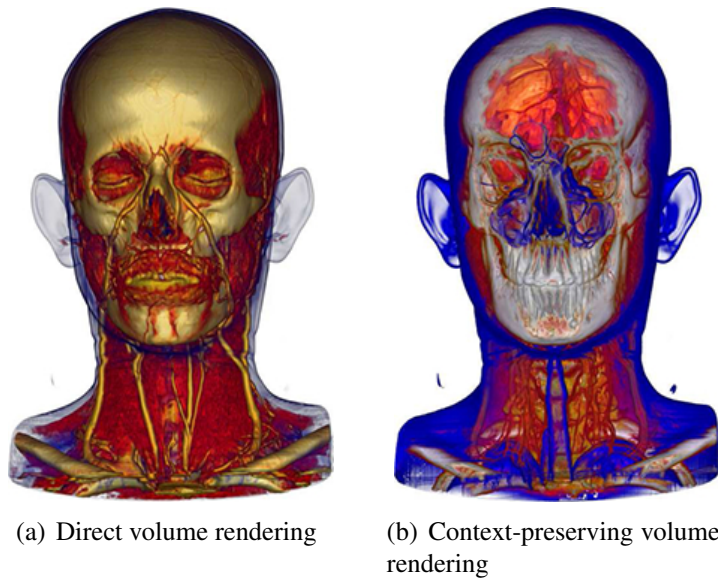


Figure 2.5: Images from *Illustrative Context-Preserving Volume Rendering* [2].

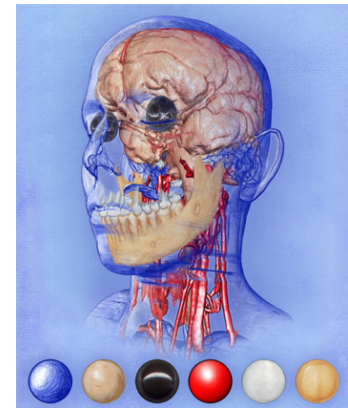


Figure 2.6: Rendering using different styles for different structures. Image from *Style Transfer Functions for Illustrative Volume Rendering* [5].

Volume rendering [5]. Regular shading of a sample in ray-casting is based on a the view point, a light source and the normal at the sample position. The shading of a sphere will sample an opacity value and a color for all possible normals which are visible to the viewer. Style transfer functions use a pre-generated lit sphere texture. The calculated normal at each voxel will then serve as a lookup for the lit sphere. Since the lit sphere is not restricted to one basic color, one can now combine different rendering styles for different structures within the data. Since the shading is decided by a pre-rendered texture, the different styles can be changed easily, to for instance contour shading, for the outer structures. This makes occlusion easier to handle than a using regular a transfer function. The contours can be created from a sphere which only contain values at the outline. Applying this style to the outer structures will reveal inner structures which can be rendered in a separate style. Figure 2.6 show how different styles for different features provide a clear distinction between the features.

2.3 Clipping and Cut-Aways

A solution to the problem with occluding objects is to remove the data which are covering the feature of interest in the final image. This technique is called clipping. Clipping is a metaphor derived from the traditional illustrators toolbox. In essence a geometrical shape is removed from the rendering scene. The most basic approach for clipping is using a plane to define what is clipped away. A plane is positioned within the volume and the data which is above the plane is not rendered. This technique can provide the fast and easy exploration from slice rendering while still containing the context and is a popular tool for volume exploration. With the use of only clipping planes, complex clippings becomes difficult. Weiskopf et al. [39] introduced a clipping technique where geometrical structures served as the clipping tool. Examples of different clipping structures can be seen in Figure 2.7.

Technical illustrators have used cut-aways for presenting features hidden within solid structures. In computer graphics the basic cut-away tool is an automated clipping technique mainly used for presenting features within the data. Diepstraten et al. presented some of the first automated computerized cut-away techniques [13]. In Figure 2.8 a cut-away is applied to reveal the previously occluded structure.

Viola et al. introduced a novel approach for automated focus + context rendering, by automatically generating cut-aways in direct volume rendering. *Importance Driven Volume Rendering* [36] incorporates an "importance dimension" for various structures. Structures, such as internal organs, which often are more interesting than skin and bone will have a higher importance factor. Creating cut-aways for the important features can be done using *Maximum Importance Projection* (MImP). With MImP only the features

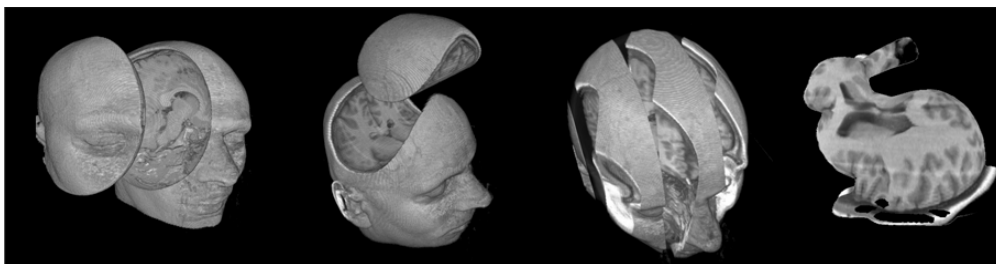


Figure 2.7: Image from *Volume Clipping via Per-Fragment Operations in Texture-Based Volume Visualization* showing different types of clipping geometries [39].

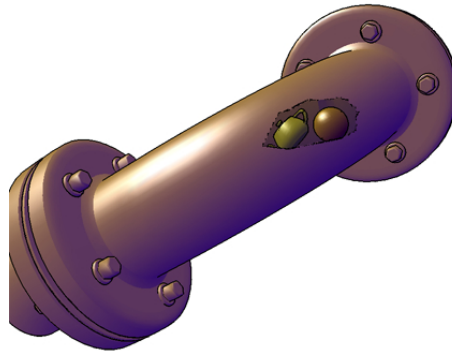


Figure 2.8: Image from *Interactive Cutaway Illustrations* [13] using cut-aways to reveal hidden structures.

of highest importance are visible. Other composition techniques was also presented such as *Average Importance compositing*. Cutting out only what is directly in front of the interesting feature does not give a good depth perception, and localization can be a problem. Viola et al. propose a conical shaped cut-away. Conical shaped cut-outs, compared to cylindrical shaped cut-outs, make the sides of the cut visible to the viewer. Visible layers on the sides of the cut enhance the visible depth cue, as shown in Figure 2.9. Cut-aways similar to Viola et al. approach was also implemented in Stephan Bruckner's VolumeShop [2].

Cut-aways is a powerful tool for presenting data, but the cut-aways require some form of pre-defined structures, for instance segmentation. Segmentation is the process of defining structures within a dataset and can be time-consuming. Other methods for cutting away different occluding structures without the need of pre-defined structures are desired. Volumetric data are often structured as several layers covering each other. To gain visibility of the features within the volume, the layers can be peeled away. A problem might occur when the feature of interest and an occluding layer has the same value. This may turn opacity modulation in the form of a transfer function insufficient as a means for exploration. To solve this problem Rezk-Salama and Kolb introduced opacity peeling [30]. The basic algorithm goes as follows: a ray accumulates color and opacity, α , through the volume. When α reaches a certain threshold, color and opacity is set to zero, the ray is moved through the current layer, then the ray starts accumulating again. The user defines how many times this process should repeat itself. Figure 2.10 show how opacity peeling can be used to reveal inner structures.

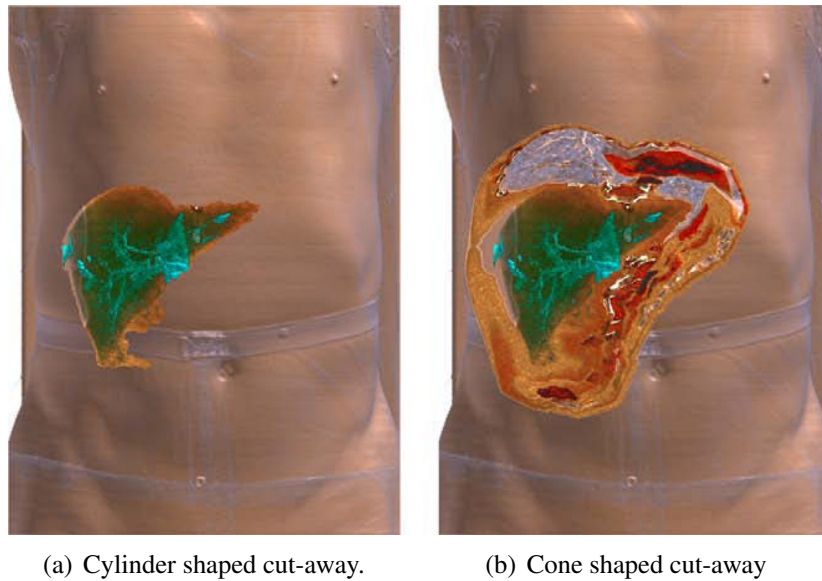


Figure 2.9: Images from *Importance-Driven Volume Rendering* [36]. Cone shaped cut-aways provide a better perception of how deep the feature in focus is inside the volume.

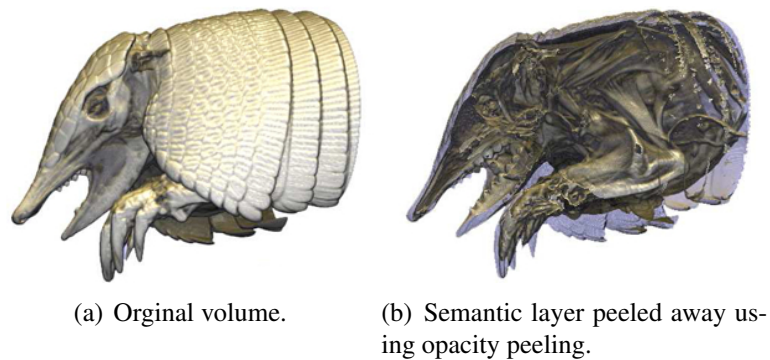


Figure 2.10: Images from *Opacity Peeling for Direct Volume Rendering* [30].

2.4 Splitting

Volume splitting is a useful tool for both volume exploration and presentation. As an extension of the volume clipping technique discussed in Section 2.3, the idea is to divide the volume into two or more separate parts. Each part can then be transformed and rendered separately. Extensive research has been done in splitting and can be considered as a form of rigid transformation of each part. In focus and context rendering, splitting has been used to deal with occlusion.

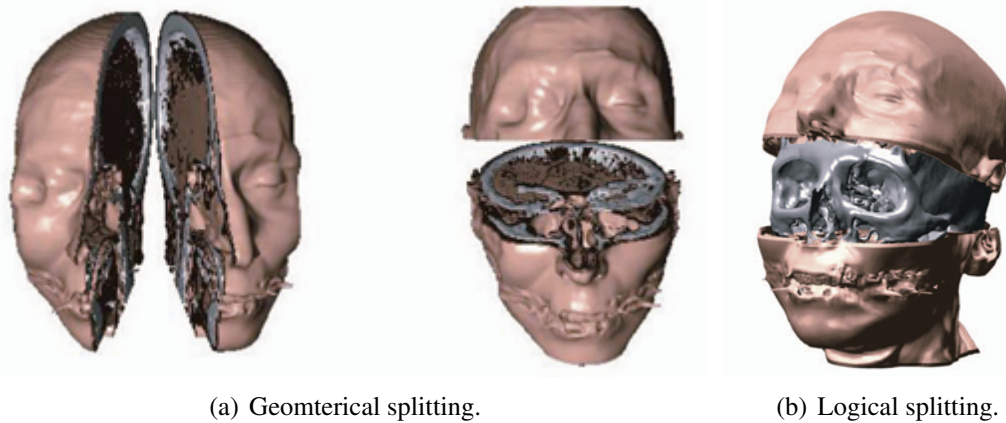


Figure 2.11: Images from *Volume Splitting and Its Applications* [20] showing the two types of splitting.

Splitting can be applied to several areas in computer graphics, for instance for special effects. Dividing an object into several separate parts can be used to simulate explosions. Splitting is often divided into two separate types; Geometrical splitting, where each part has a certain geometrical shape. Logical splitting is combining splitting with pre-defined structures. The split is then aligned with the surfaces of the pre-defined features. The difference between the two splitting types are shown in Figure 2.11. Islam et al. [19, 20] discuss various types of splitting, using spatial transfer functions introduced by Chen et al. [6] for volume animation, volume visualization and special effects.

Splitting does not automatically solve the problem of occlusion. The different parts can still cover the feature of interest. Exploded views is an illustration technique based on splitting which solve the problem of occluding parts. Traditional illustrators has used exploded views for a long time, da Vinci illustrated a baby within the womb using a form of exploded views as early as 1511, as shown in Figure 1.2.

Illustrations are static images. the view point is fixed and interaction is very limited. In volume rendering the view point can be changed. This generates a challenge with exploded views. The line of sight to the object in focus must be kept clear from the other parts. Bruckner and Gröller presented exploded views incorporating view-dependency as a crucial factor [4]. When splitting a volume into separate parts, objects that are normally occluded by outer structures can become visible. If the scene is rotated, a part can get in front of another and the occlusion has not been solved. In Bruckner

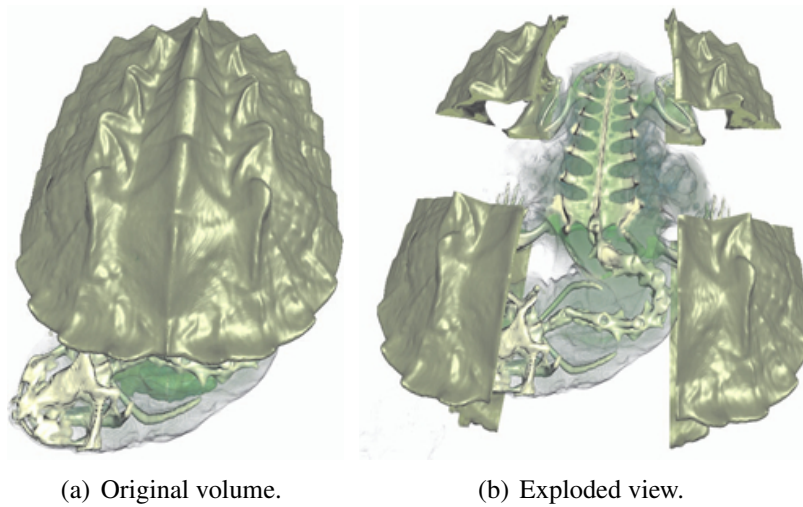


Figure 2.12: Images from *Exploded Views for Volume Data* [4].

and Gröller's exploded views each part is inserted into a physics system. The separate pieces apply a force onto each other. To solve the occlusion parts problem, a view-force is introduced. The view-force is orthogonal out from the view direction. This force creates a line of sight to the object in focus. An example of exploded views created by Bruckner and Gröller's technique can be seen in Figure 2.12

2.5 Deformation

Deformation is an illustration technique which has been used in traditional illustrating for a time. Deformation is the process of altering the shape of an object to enhance or reveal certain features. This process can be divided into four major areas; Continuous shape deformation, in essence altering the shape of the structure, discontinuous deformation, for instance creating cuts and peels, image based deformation, altering the representation of the object in the rendering process and object based deformation, reorganizing the structure of the object before rendering.

Altering the shape of an object can lead focus from one feature to another. The idea is to have a distortion effect, for instance a magic lense at the mouse cursor, to improve the details at the object of interest, similar to a magnifying glas over a map. This can also be done in volumetric visualization through shape deformation. Non-linear ray casting

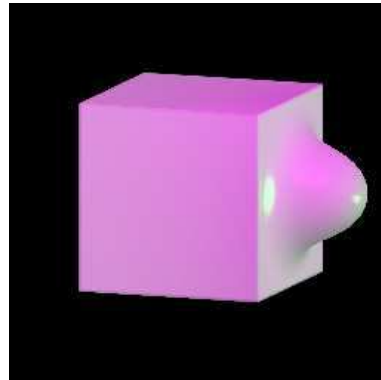


Figure 2.13: Image from *Space Deformation using Ray Deflectors* [21] showing the deformed shape using ray deflectors.

can be used for image based, shape deformation. Kurzion and Yagel introduced ray deflectors [21] as a mean for altering the representation within the rendering process. A simple deformation using ray deflectors are shown in Figure 2.13. Ray deflector serve as a gravity field within the ray caster, bending the path of the rays going through the data. Ray deflectors are a fast deformation technique, but complex deformations can become difficult with the use of simple gravity fields.

Object based deformation is the process of altering the structure of the object before rendering. An endoscopy is performed to investigate the colon for abnormalities. However, a ordinary endoscopy has certain limitations. It is difficult to obtain a good overview of the colon when the camera can not be turned around to see structures from another angle. Also the procedure is intrusive for the patient. Another solution is to take an image of the colon using a CT. The colon can be segmented out and the data can be investigated as many times as needed. To obtain a good overview of the colon, one can



Figure 2.14: Colon segment stretched out onto a plane. Image from *Virtual Colon Unfolding* [1].

stretch the colon out on a plane. Batroli et al. presented a technique for virtual colon unfolding [1]. In virtual colon unfolding the idea is to show the entire colon in one image, see Figure 2.14. This is done by resampling a colon segment to fit on a plane.

In the research field of surgical planning and simulation, deformation is an important tool. When performing an endoscopy, the physician can move structures with the instrument to get a better view. When performing virtual endoscopy, simulation of the instrument interacting with the colon can be desired. Schulze presented a technique for simulating endoscopic views [32]. using a shape deformation to allow the viewer to alter the structure of the colon, Figure 2.15 show how the user can alter the shape of the colon to simulate the interaction of the instrument. Deformation was done on the voxel level with the combination of a 3D ChainMail and a relaxation technique. The 3D ChainMail algorithm [17] was introduced for fast volume deformation by Gibson. ChainMail has low computational cost and is suited for large data sets. Florian Schulze et al. extended their technique for deformation of large volume dataset [31].

Defining deformations in volume space can be challenging, and intuitive methods for volume deformation is desired. Walton and Jones extended the use of wires [33] to create volume wires [37] as means for shape deformation. The volume wire creates an attribute field which contains information about the deformation for each voxel. Walton and Jones extended their work [38] to include support for multiple wires which allowed discontinuous deformations. Different types of deformations using volume wires are shown in Figure 2.16.

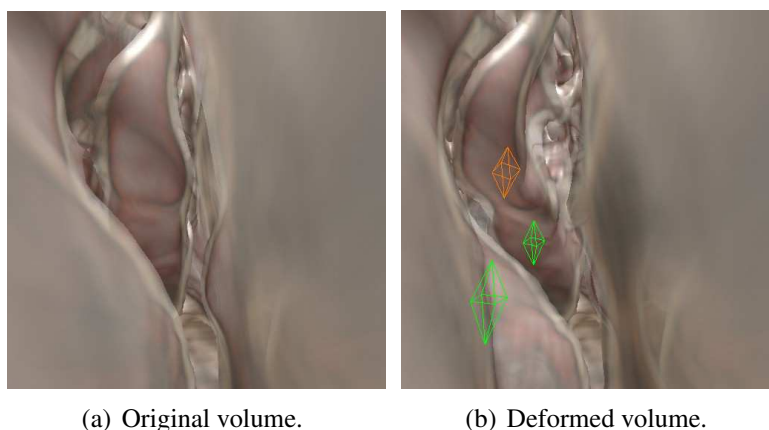
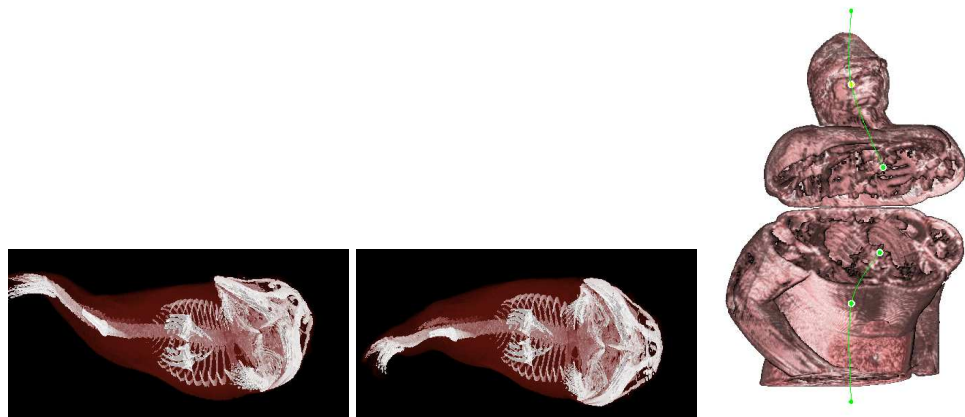


Figure 2.15: Images from *Direct Volume Deformation* [32].



(a) Images from *Volume Wires : A Framework for Empirical Non-linear Deformation of Volumetric Datasets* [37]. Image on the left show original volume.

(b) Image from *Interacting with Volume Data: Deformations using Forward Projection* [38].

Figure 2.16: Deformations using volume wires.

There has also been done extensive research in physics-based shape deformation. This may not be crucial in exploration and presentation, however it could give a better feel of how structures are linked together. More accurate techniques than Chain-Mail has been implemented for physics-based deformation. Finite Element Method and Mass-Spring models are well established algorithms for physics simulation [40, 28, and references therein]

The use of shape deformation can provide novel tools for volume exploration and presentation. Continuous deformations do not solve the problem with occlusion. Discontinuous deformations, such as cuts and peels can reveal structures hidden underneath outer layers. Discontinuous deformation is an important area in volume deformation. Discontinuity in a deformation raise a new challenge in volume rendering. Revoxelization is not always preferred since the original state of the volume is lost after the deformation has been applied. Rendering iso-surfaces is often done by generating polygonal structures, for instance with the marching cubes algorithm. The shape of a polygonal structure is easily distorted by transforming the vertices, but to illustrate a cut, re-meshing would be needed which can be a complex task.

Previously, in Section 2.4, splitting was discussed as a simple form for cutting the

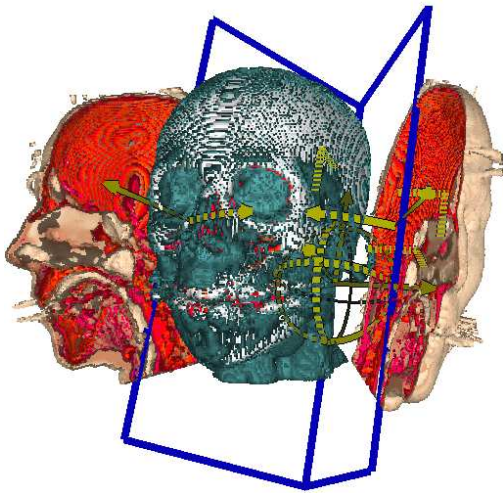


Figure 2.17: Image from *Using Deformations for Browsing Volumetric Data* [25]. The image quality suffers when a point based rendering scheme is used.

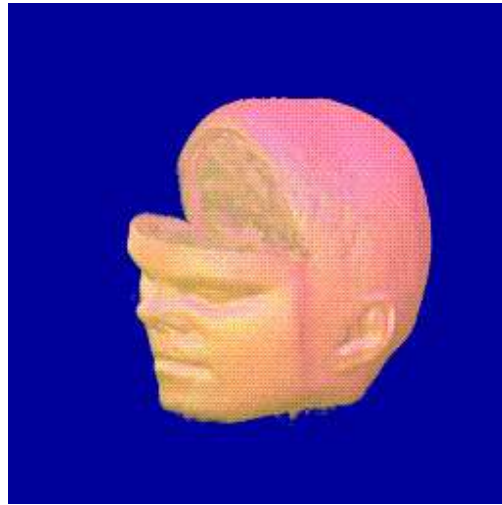


Figure 2.18: A cut generated using ray deflectors. Image from *Continuous and Discontinuous Deformation using Ray Deflectors* [22].

volume, but direct discontinuous shape deformation can be used to simulate more familiar deformations such as peel-aways. As mentioned earlier in Section 1, illustrating peel-aways have been used to show both feature of interest and occluding structures without the compromise of semi-transparency. Peel-aways provide a powerful tool for exploration and presentation in volumetric visualization. McGuffin et al. published an extensive research in various types of deformation for browsing volumetric data [25]. In the approach of McGuffin et al., points were used as primitives for rendering. This simplifies the problem of discontinuity because the empty space is not rendered and a forward transformation of each sample point can be used. A point based rendering scheme penalize the image quality, as shown in Figure 2.17.

Complex deformation in direct volume rendering is challenging. Discontinuous regions must be handled differently than in polygonal or point based rendering. Often direct volume rendering is done on dedicated graphics hardware and there are restrictions on what types of data one can store in the memory to the graphics processing unit (GPU). Many different approaches has been presented to handle discontinuous deformations in direct volume rendering. Kurzion and Yagel extended their work on ray deflectors to include discontinuous deformation [22], see Figure 2.18. The discontinu-

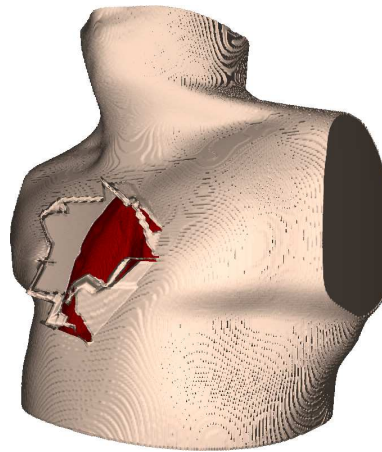


Figure 2.19: Discontinuous deformation created by deforming the proxy geometry. Images from *Real-Time Deformation for Illustrative Medical Volume Rendering* [26].

ous region was created using a gravity field as the center of the cut, and only allowing points to sample points from the same side of the field. If a point sampled a point from the other side of the field it would be treated as a hole.

In ray-casting, entry and exit points need to be calculated as start and end positions for each ray into the volume. When ray-casting is done on the GPU, the most simple calculation of the entry and exit points is done by rendering the front faces of a cube to one texture, and the back faces to another. Rendering time can be reduced by generating more complex structures for front and back faces of the volume. This is often referred to as the proxy-geometry of the volume. Mensmann [26] introduced a technique for deforming the proxy-geometry to obtain real-time illustrative deformation. The proxy-geometry was built using a surface extraction technique designed explicitly for the 3D ChainMail algorithm. Then cuts were created using a pre-defined cutting template. Discontinuous deformation by Mensmann are shown in Figure 2.19

Deformation can be obtained using a technique called displacement mapping. Displacement maps were introduced by Cook [7] as means for shape-deformation of a surface. Traditionally displacement maps are represented as 2D textures. The texture contains the displacement of each fragment of the surface of which it is mapped to, basically it is a 2D vector field. See *Multiresolution rendering with displacement mapping* [18] and *Adaptive view dependent tessellation of displacement maps* [14] for more information on regular displacement mapping.

Carlos D. Correa et al. introduced an extension of displacement mapping to include discontinuous deformation in volume graphics [9], as shown in Figure 2.20. For surfaces, only 2D displacement maps are required to create deformations. However, when dealing with volumes, 2D displacement maps may become insufficient because volumes contain information underneath the visible surface. Volumetric displacement maps are required to deform both the visible surface and the inner structures in a volume.

Volumetric displacement maps can be considered as a 3D vector field. Traditionally displacement is considered as the forward transformation of each sample. In direct volume rendering the origin of a sample position must be defined. For this problem Correa et al. suggests inverse displacement maps to handle discontinuities. A forward displacement function $\vec{D}(x, y, z)$ is the displacement of a point $\mathbf{p} = (x, y, z)$ to $\mathbf{p}' = (x', y', z')$. This leads to the equation:

$$\mathbf{p}' = \mathbf{p} + \vec{D}(\mathbf{p}). \quad (2.1)$$

The transformation \vec{D} is specified procedurally, then the inverse transformation \overleftarrow{D} is sampled as a texture of size $w \times h \times d$ at discrete positions $\{x, y, z | x = 0, 1, \dots, w; y = 0, 1, \dots, h; z = 0, 1, \dots, d\}$. The inverse transformation \overleftarrow{D} satisfies the equation:

$$\mathbf{p} = \mathbf{p}' + \overleftarrow{D}(\mathbf{p}'). \quad (2.2)$$

Discontinuous areas can be defined as where $\overleftarrow{D} = \emptyset$. When rendering, the transformation for each sample is retrieved from the discrete inverse transformation and sample data can be retrieved from the original position.

In *Discontinuous Displacement Mapping for Volume Graphics* [9] displacement maps are pre-defined templates which illustrate cuts, peels and other deformations. In the following chapter an approach for creating view-dependent peel-away based on the deformation technique given by Correa et al.



(a) Original volume.

(b) Peel created using a template based deformation scheme.

Figure 2.20: Images from *Discontinuous Displacement Mapping for Volume Graphics* [9].

Chapter 3

View-Dependent Peel-Away in Direct Volume Rendering

In chapter 2 several different techniques for exploring and presenting structures within a volumetric data set was presented. The problem with occlusion was discussed, and how it can be solved using different kinds of techniques. In Section 2.5, deformation was discussed as a way to handle the problem with occluding structures. In this chapter, an approach for intelligent positioning peels according to the view-point will be presented.

When exploring data for diagnosis or other purposes, the surrounding structures of a certain feature might be as interesting as the feature itself. Compared to semi-transparency and other non-photorealistic rendering techniques, peel-aways has the property of preserving information of the occluding structures. While techniques like cut-aways and contour rendering leave out much information, peels transform the occluding structures to a non-occluding position.

In this chapter, the techniques for generating view-dependent illustrative peel-aways in direct volume rendering will be presented. The approach in this work is based on the illustrative deformation technique presented by Correa et al. [9, 8]. This approach is suited for view-dependent peel since physics simulation is deemed unnecessary. With pre-defined templates, rendering time can be reduced significantly.

Figure 3.1 show the steps required for visualizing peels with a template based deformation scheme. In the first step after retrieving the data, the user controls the parameters which are used in the renderer and the degree of displacement in the peel-template. The peel-template is then generated and rendering without any displacement can be performed. Before rendering peels, the peel-template has to be positioned within the ren-

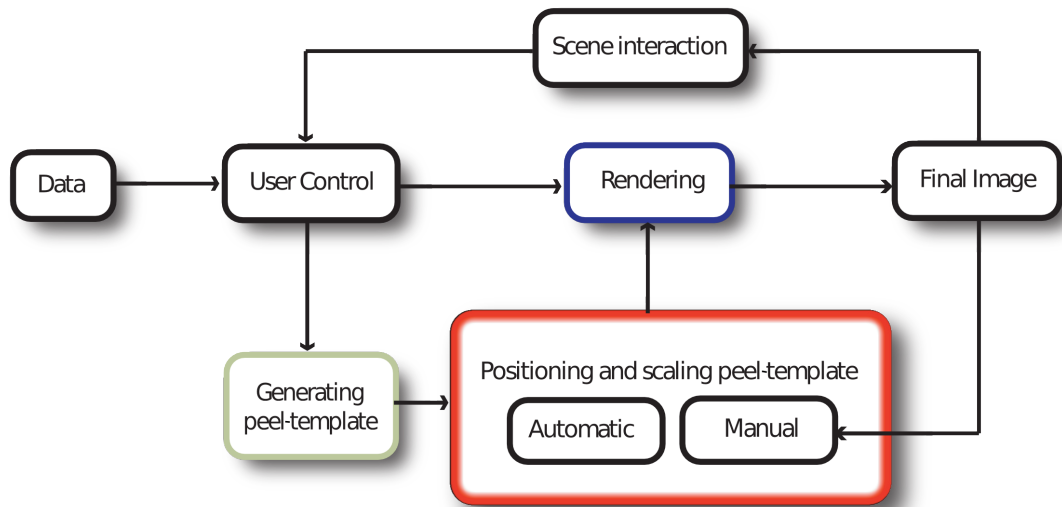


Figure 3.1: The process pipeline; After the data has been acquired, the user specifies the necessary parameters for rendering, and generating the peel-template. After an image has been rendered, the user can either position the peel-template within the rendering scene using simple mouse gestures or select a feature to be revealed. Rendering is then performed, including the peel in the final image.

dering scene. This can be done with two different techniques. Manual positioning and scaling of the peel-template allows the user to explore the desired areas of the volume, and automatic positioning and scaling provides fast presentation of features of interest. After the peel-template is positioned, volume rendering using discontinuous displacement maps can be performed. In Section 3.1, the steps needed for visualizing peels using discontinuous displacement maps will be presented. Furthermore, in Section 3.2, the techniques for creating view-dependent peels by positioning, orienting and scaling the peel-template are introduced.

The basic inputs for our approach are volumetric data and a segmentation mask which describe features of interest within the data. CT and MRI scans are the main source for data the peel-aways are demonstrated on. However, the technique described are not limited to medical data and could be applied to volumes from alternative sources. The segments are represented in a binary segmentation mask and is stored as a separate volume. This is required for automatically generating peel-aways to present data pre-

defined structures within the volume.

3.1 Visualization Pipeline

The pipeline for using vector field transformations for visualizing peel-aways in direct volume rendering can be divided into three major steps:

1. Generation of the peel-template: A peel-template is generated procedurally. The template is the inverse of the transformation for each voxel which are peeled away. The template is stored as a 3D texture. Setting up and generating templates in general will be described in 3.1.1.
2. Peel placement: The vector field is scaled, oriented and placed within the rendering scene using a transformation matrix. The orientation of the peel is always away from the viewer. This assures that the peel itself does not occlude the feature of interest. Interaction for both scaling and positioning is described in Section 3.2
3. Rendering: The scene is rendered with a ray-caster using front-to-back color compositing. The ray-caster contains several steps to handle transformation, discontinuities and peel-alignment. This is described in Section 3.1.2.

A fourth step is required for guided navigation. The size, position and depth of the vector field has to be calculated according the view-point in order to reveal features of interest. This step is performed between the first and second step in the visualization pipeline and is described in Section 3.2.

3.1.1 Peel-Template Setup

The peel-template is a map, which contain the inverse displacement vector for each element in a peel. If a vector field is of the same size as the volume, each voxel in the volume would get a unique displacement value. This imposes a problem with performance since volumes usually have a high resolution, and generating the vector field at interactive rates becomes difficult. The solution is to have a displacement map of a resolution low enough to maintain interactivity. In our approach the size of the vector field is set to $64 \times 64 \times 64$, regardless of the size of the volume.

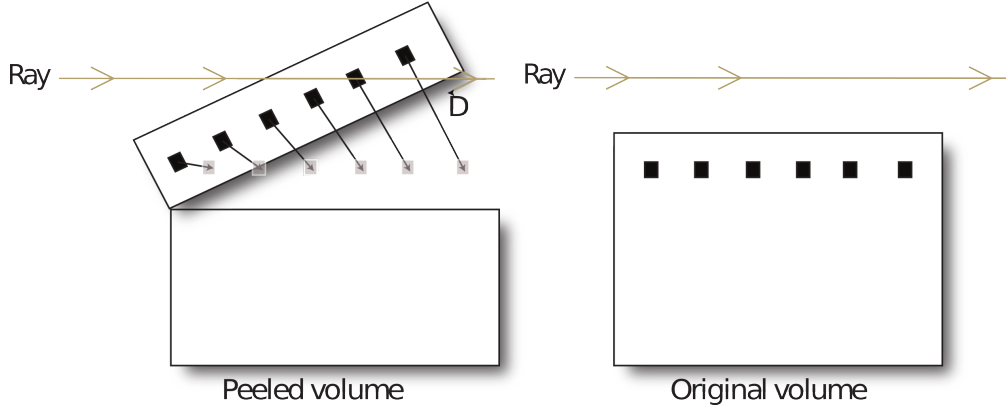


Figure 3.2: The vector field contains the transformation back to the original position.

To realize deformation in hardware based direct volume rendering there are certain restrictions which must be considered since the ray-casting is done on the fragment shader on the GPU. A volume coordinate, (x, y, z) in volume space is in the range of $\{x \in [0, w]; y \in [0, h]; z \in [0, d]\}$, where w , h and d are the width, height and depth of the volumetric dataset. However, since the volume is stored as a 3D texture, the volume coordinates in the fragment shader is in the range $[0, 1]$. This means that the displacement vectors must be normalized to from a transformation in volume space, to a transformation which is valid on the fragment shader.

The fragment shader renders each pixel separately. This means that a voxel can not be transformed to another position within the ray-caster. The shader retrieves samples from the volume at positions along the ray. Because of this limitation the inverse displacement is required. This is required because the shader needs the original position of the voxel.

The first step in generating displacement vectors is to define a transformation function $T_f(\mathbf{p})$. This is the transformation from \mathbf{p} to \mathbf{p}' . To build the the vector field \overleftarrow{D} , the inverse transformation function T_f^{-1} is calculated, to obtain the following equation:

$$\mathbf{p} = T_f^{-1}(\mathbf{p}') = \mathbf{p}' + \overleftarrow{D}(\mathbf{p}'). \quad (3.1)$$

The inverse gives us the original position each sample within the vector field. This means that the displacement vectors can be calculated as follows:

$$\overleftarrow{D}(\mathbf{p}') = T_f^{-1}(\mathbf{p}') - \mathbf{p}'. \quad (3.2)$$

The inverse displacement function \overleftarrow{D} of size $w \times h \times d$, \overleftarrow{D} is sampled at discrete positions $\{x, y, z | x = 0, 1, \dots, w-1; y = 0, 1, \dots, h-1; z = 0, 1, \dots, d-1\}$. This is illustrated in Figure 3.2.

The cut-region in a peel is defined as empty space. In mesh-based and mesh-less rendering [27], the cut-region is handled implicitly, since empty space is not rendered. This is not the case in direct volume rendering since empty space often is rendered as voxels with a value of 0. The cut region must be explicitly defined in the displacement map. In a peel, the cut region is the area between the cutting plane, and the peel. For rendering, this area must be marked as discontinuous.

Handling Discontinuity

Correa et al. suggested several approaches for defining non-continuous areas. One solution is the Out-of-bounds displacement, the cut region can be defined as non-valid displacement. For instance a displacement outside volume-space is a invalid position and can be interpreted as a discontinuous area in the shader. This can create problems due to texture interpolation. Another solution is to use binary α -mask which defines the cut-region. The α -mask is a separate function, $A(\mathbf{p}')$, which can be defined as follows:

$$A(\mathbf{p}') = \begin{cases} 0 & \text{if } \mathbf{p} \text{ is not displaced} \\ 1 & \text{if } \mathbf{p} \text{ is displaced} \\ 2 & \text{if } \mathbf{p}' \text{ is in a discontinuous area.} \end{cases} \quad (3.3)$$

$A(\mathbf{p}')$ can be stored in the α -channel in the vector field texture. The α -mask will then become a representation of the surface of the cut. Since the vector field is not differentiable on the surface of the cut and the resolution of the vector field is quite low, stair-case artifacts might appear. Correa et al. suggests a solution for creating a smooth surface of the peeled structure by smoothening the α -mask and using the gradient of the α -mask, $\nabla A(p')$ [9]. The smoothening process is time-consuming, however and has not been implemented in this thesis in order to achieve interactive framerates.

3.1.2 Ray-Casting through Vector Fields

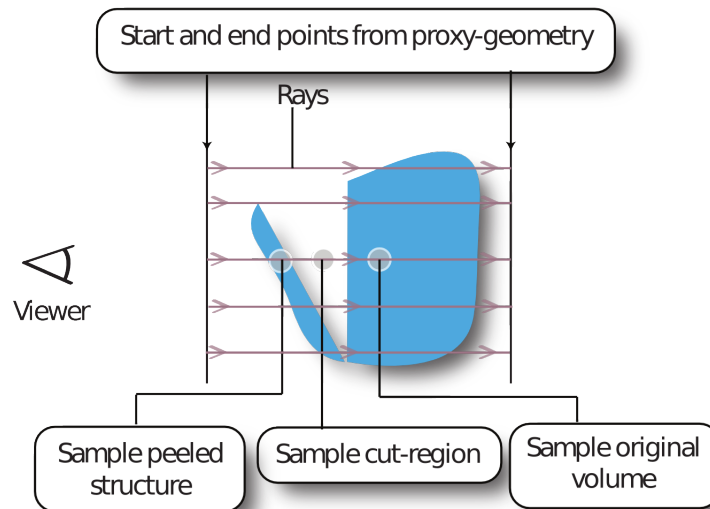


Figure 3.3: The story of a ray; The first step is to acquire the start and end points from the proxy geometry. Then the ray retrieves samples using the extended sampling function which handle the three different cases.

In our approach a basic front-to-back ray-caster is used for volume rendering. Since dedicated graphics hardware is suited for vector calculations, the ray-caster can be efficiently executed on the GPU. A proxy geometry is required for the entry and exit points of each ray. The basic proxy geometry is a geometrically defined structure which contains the volume. As shown in Figure 3.4, when handling deformations the basic proxy geometry is not sufficient. This is because structures can be transformed outside the original geometry. The proxy geometry is then extended to include both the original volume and the displaced structures.

The first step in retrieving a sample is to check if the sample is within the vector field. If the sample is outside the vector field, the sample can be retrieved from the current position. If the sample is within the vector field, another technique must be applied. Below, the general technique for retrieving samples within the peel-template is given.

Every sample is checked for displacement according to the function $A(\mathbf{p})$, see equation 3.3. A volume can be represented by a sampling function $f_s(x, y, z)$. When a dis-

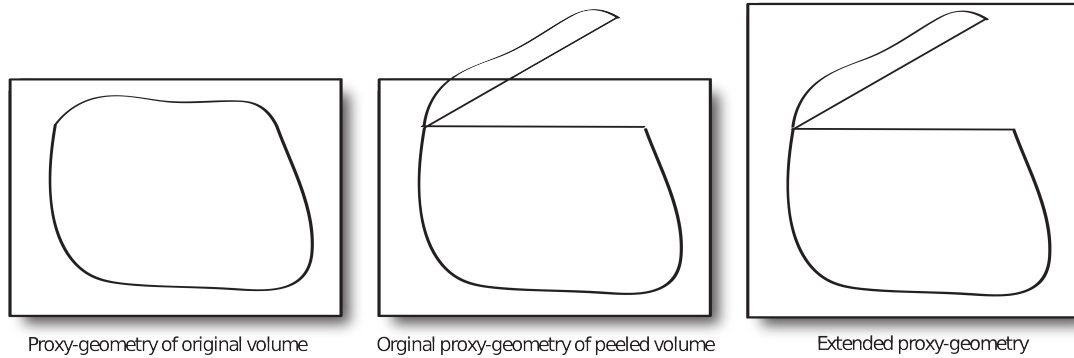


Figure 3.4: The proxy geometry must be extended to contain the peeled structures.

placement map is applied to a scene the sampling function changes to $f'_s(x, y, z)$, where

$$f'_s(\mathbf{p}') = \begin{cases} f_s(\mathbf{p}') & \text{if } A(\mathbf{p}') == 0 \\ f_s(\mathbf{p}) & \text{if } A(\mathbf{p}') == 1 \\ 0 & \text{if } A(\mathbf{p}') == 2. \end{cases} \quad (3.4)$$

The three separate cases are illustrated in Figure 3.3.

The peel can be aligned in two different ways. It can either be aligned with the axis of the vector field, or it can be aligned with a feature of interest. When aligning the peel with the feature of interest, the voxels from the feature should not be displaced. This changes the sampling function to the following:

$$f'_s(\mathbf{p}') = \begin{cases} f_s(\mathbf{p}') & \text{if } A(\mathbf{p}') == 0 \text{ or } \mathbf{p}' \text{ is within the feature of interest} \\ f_s(\mathbf{p}) & \text{if } A(\mathbf{p}') == 1 \\ 0 & \text{if } A(\mathbf{p}') == 2 \text{ or } \mathbf{p} \text{ is within the feature of interest.} \end{cases} \quad (3.5)$$

In volume rendering the normals are often defined as the gradient of the sample function, ∇f_s . If a volume only contains a single scalar and is represented as a 3D texture, the gradient, or normal, can be stored in the RGB channel while the density is stored in the α -channel. However, pre-calculated normals must be recalculated when the volume is deformed. This is a complex task and different solutions must be implemented for different types of peels and peel-alignments.

The central differences algorithm can be used to calculate the gradient on-the-fly in

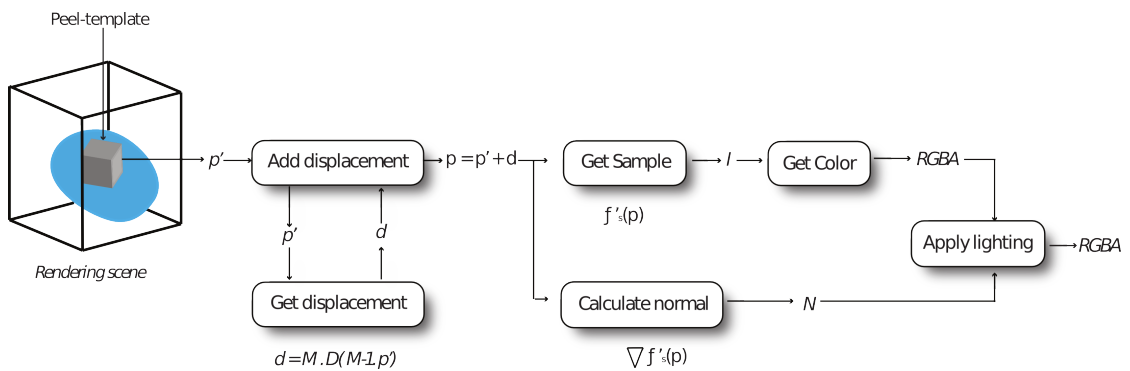


Figure 3.5: The pipeline for sampling within the ray-caster.

direct volume rendering. The central differences approximates the gradient by calculating the difference between the neighboring voxels in the positive x, y and z direction and the neighboring voxels in the negative x, y and z direction, as illustrated in Figure 3.6. Using the central difference calculation to approximate gradients on-the-fly is time-consuming since several samples has to be retrieved for each position in the ray-caster. Due to the increase in computational resources on dedicated graphics hardware, the normals can be calculated on-the-fly while still achieving interactive frame rates. When the gradient is calculated from the sampling function within the ray-caster, the process for normal calculation for both feature-aligned peels and axis-aligned peels becomes the same because the sampling function changes for both processes.

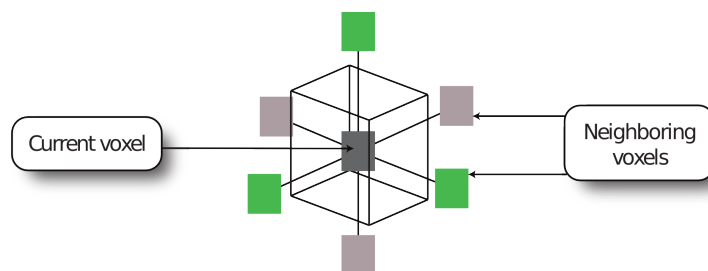


Figure 3.6: The central differences algorithm approximates the gradient by calculating the difference between the values of the neighboring samples in the positive axis-direction (green) and the negative axis-direction (grey).

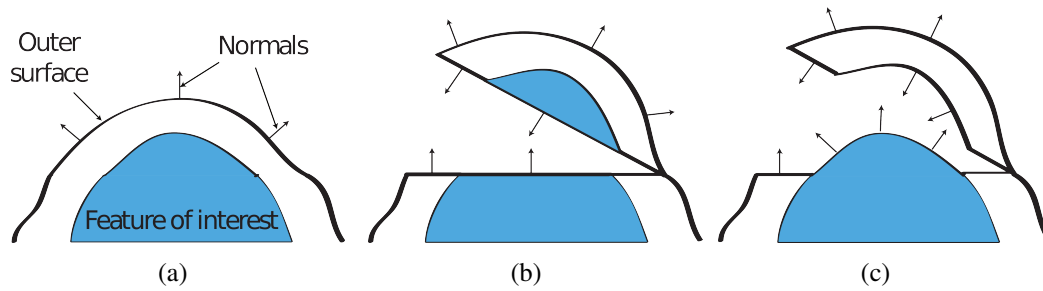


Figure 3.7: Normals change differently when the two types of peel-alignments are applied. Figure (a) show the original volume. Figure (b) show an axis-aligned peel. In Figure (c) a feature aligned peel is applied.

3.2 Interaction

Interaction is an important aspect for exploring volumetric datasets or presenting predefined features in 3D volume rendering, a common type of interaction is altering the orientation of the scene. To investigate the surface of an object, the viewer can observe it from different angles. Another type of scene interaction is altering the representation of the rendering. This is often done with an interaction interface for the transfer function.

For deformation-based exploration, the first step is to position the deformation to a user specified location. In surgical simulation systems, a highly sophisticated interaction technique needs to be implemented to simulate a scalpel making a cut. In a template based deformation scheme, a simpler, less computationally expensive interaction is desired.

The goal is to make an intuitive interaction interface. This means that a user only has to specify the location and the degree of the peel itself, the orientation is then calculated automatically according to the view-point. With an already defined peel-template, the template can be scaled, positioned and oriented with a transformation matrix. Costly re-calculation of the displacement vectors is then avoided.

Although a peel-template only needs to be generated once, animating the peel can provide a stronger spatial relationship between the peeled structures and the structures underneath. Since the resolution of the peel-template is lower than the volume, it is therefore only necessary with a single traversal through the template to generate the displacement vectors. The vectors in a peel-template can be generated on-the-fly. This enables user controlled animation of the peel.

3.2.1 Peel-Template Positioning and Scaling

The process of explorative interaction using pre-defined peel-templates can be outlined in the following way: a peel-template contains a position orientation and a scale. The user can define a position by clicking on the rendered structure. The position is then set to be on the surface of the rendered object. The normal vector for the peel-template is then calculated to be the normalized gradient in the volume, ∇f_s . After positioning the template, the user scales it to a desired size. Translating, rotating and scaling can all be combined into a single transformation matrix, M . Since the peel-template itself is scaled and rotated, the lookup position in the vector field and the displacement vectors need to be scaled and rotated accordingly. The scaled and rotated displacement vector, $\overleftarrow{(D)'(\mathbf{p}')}$, is calculated as follows:

$$D'(\mathbf{p}') = M \cdot D(M^{-1} \cdot \mathbf{p}'). \quad (3.6)$$

A specific problem with peel-aways is that the peel itself can occlude structures underneath. If the peel direction is towards the view point, the peel can cover the cut. The view point must therefore be considered when orienting the peel to ensure the best line of sight to the feature in focus. Figure 3.8 illustrates how a peel can cover the structures of interest.

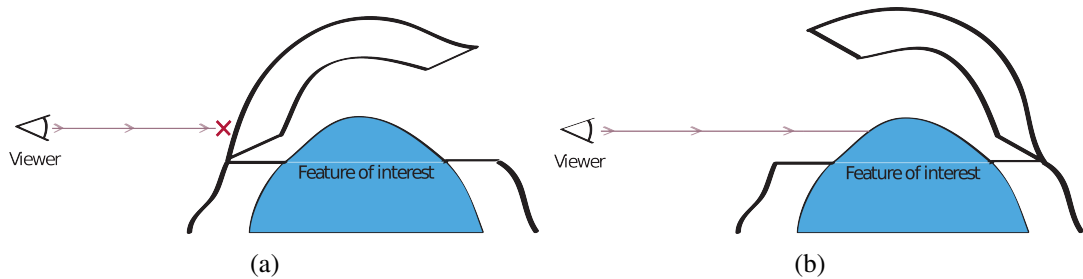


Figure 3.8: As the scene is rotated, the peel occludes the underlying structures (a). Rotating the peel ensures the maximum visibility of the underlying structures while minimizing the transformation of the peel (b).

The peel can always be transformed into a position revealing the features of interest. However, the closer two items are to each other the stronger the correlation between those items become. Rotating the peel to always bend away from the viewer minimizes the transformation needed for exposing the feature of interest. The orientation of the

peel is set to the difference between the inverse view-vector and the normal vector to the peel-template, projected on the plane defined by the normal vector.

3.2.2 Automatic Placement Orientation and Scaling

Automatically generating cut-aways for features of interest have been done for a long time in volume visualization [36, 13]. This provides a fast, easy method for presenting structures within the data. View-dependent cut-aways for illustrative rendering generate cuts from an arbitrary view-point to expose features within the data. The goal is to generate peel-aways which have the same criteria as view-dependent cut-aways. This means that a peel must go as deep as the feature of interest. Since sampling through the peel-template is more costly than sampling in the original volume, the vector field must be scaled down to the minimal size, while still revealing the entire feature.

To automatically position the peel-template, the logical center for the feature of interest must be calculated. The center can be approximated with a three dimensional distance transform [16]. The distance transform generates a map of the shortest distance from any given voxel within a feature to the surface, see Figure 3.9. The center of an object would then be classified as the voxel with the highest distance to the surface of the object.

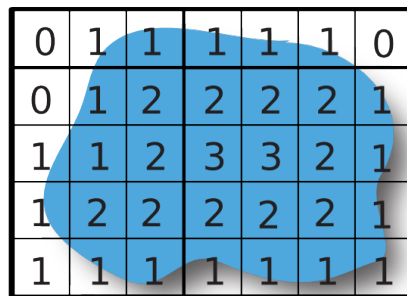


Figure 3.9: Distance transform, the value at each position is the shortest distance to the surface. The position with the highest value can be interpreted as a logical center.

The distance transform approximation is a sufficient approximation for convex structures. In medicinal CT, many structures are close to convex and the chamfer system works well for structures such as a brain or a liver. However, there are important non-

convex structures, such as blood vessels and bone. Using the distance transform for approximating the center on such structures would characterize the thickest area as the center. This is not a good approach for positioning the peel-template, as it would require a larger scale to make the entire feature visible.

Another solution for approximating the center, is using the outer bounds of the feature of interest projected on the screen, and calculate the mean between the extreme x and y positions of the projected image, as shown in Figure 3.10. To generate the projection of a feature onto the screen, a first-hit ray-caster can be used. Since this image is never viewed, shading and other effects can be ignored and faster rendering is acquired. The first-hit coordinates for both the minimum and maximum x -values and the minimum and maximum y -values need to be retrieved in order to calculate the extension of the projected feature in volume space. This can be done by first marking the α -value in the output texture at the current texture coordinate when the feature is hit by the ray and then store the volume coordinate in the RGB-channel. A linear search through the output texture is then performed to determine the minimum and maximum positions for the segment.

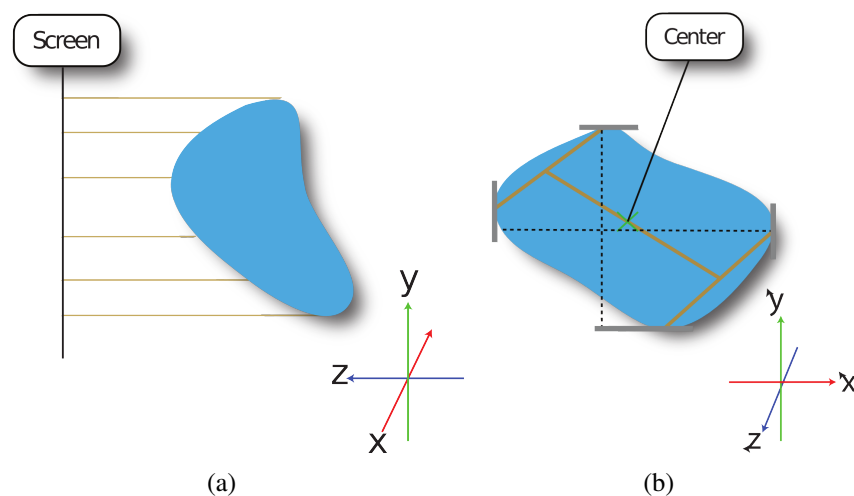


Figure 3.10: A feature are projected onto the screen(a). Then the four extreme positions are combined to calculate an approximate center(b).

To calculate the approximation of the feature center, an average between the four extreme positions are computed. This algorithm is not restricted to convex shaped structures and is a sufficient approximation of the center for automatically positioning the

vector field. The difference between a distance transform center and an outer bounds center is illustrated in Figure 3.11.

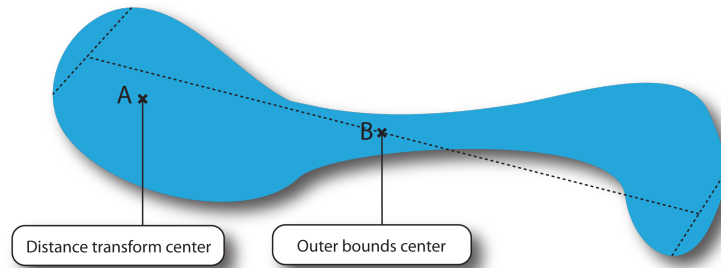


Figure 3.11: Using the distance transform would detect A as the logical center. Calculating the center from the outer bounds would result in an approximate center at B.

After positioning the vector field, the deformation needs to be oriented in such a way such that the segment in focus becomes visible. This is done by setting the up-vector for the template to point towards the view point. The next step is then to rotate the vector field so that the peel always bends out in the same direction. The consistent peel direction provides an easier view of the peel since it remains almost at the same position in the image.

The final stage in creating an automatic peel is to scale the peel such that the entire feature is revealed. To create less distraction from the peel shifting position the peel is oriented to the right of the feature, orthogonal to the cartesian rendering grid. Since the peel is aligned with the grid, the outer positions calculated from the projection of the feature can be used as a measure for the size of the peel in the x and y direction. The distance between the minimum and maximum y -positions are then set to be the y -scale of the vector field. The same process is performed to calculate the x -scale. To keep the peel from becoming distorted the x and z scale must be equal. A problem with this approach is that the feature of interest might be deeper within the volume than the projection is wide. A small peel-template can disappear within the volume and will lead to a non-visible peel. To solve this problem, the depth of the calculated position to the surface of the volume is calculated. The x and z scale is then set to the maximum of the feature projected x -size, and the feature depth. This ensures that the vector field does not disappear within the volume.

In this chapter the general technique for generating view-dependent peel-aways has been presented. In the following chapter, the specific implementation techniques will be presented.

Chapter 4

Implementation

Techniques for implementing deformations, such as peel-aways, varies from different rendering schemes. In this chapter the different techniques and solutions for achieving real-time rendering of view-dependent peel-aways will be described in detail. In Section 4.1 the setup and algorithm for generating a peel-template map is described. In Section 4.2, the process of rendering and sampling using a ray-casting rendering scheme will be presented. Further more, in Section 4.3, the algorithm for projecting the segment

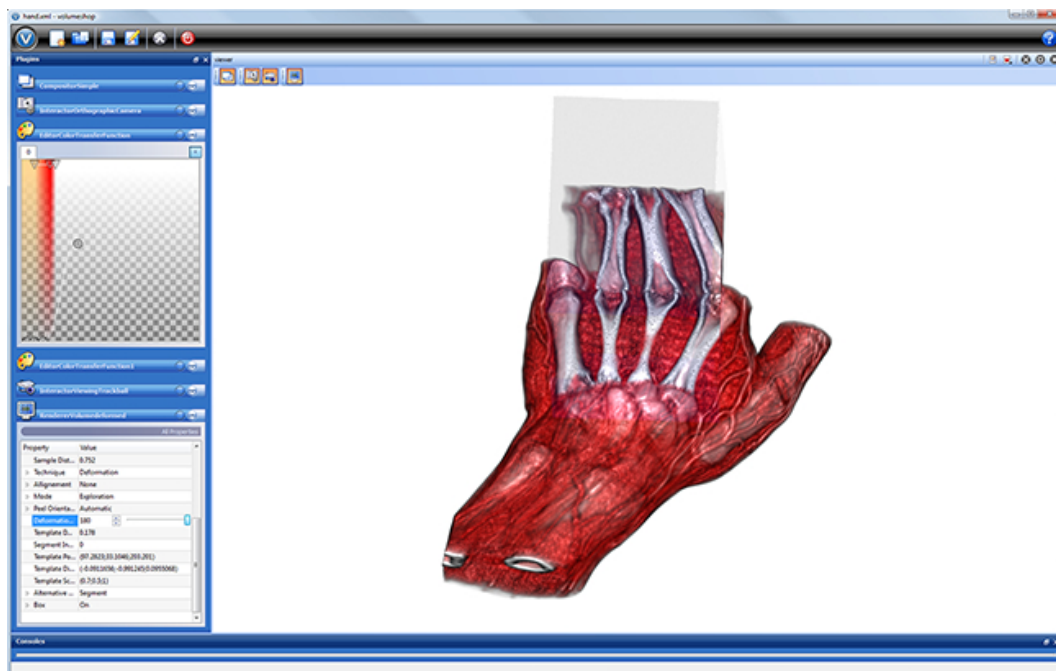


Figure 4.1: Screenshot of the VolumeShop framework.

onto the screen is described and in Section 4.4, the detail for explorative interaction is presented.

The proof-of-concept has been implemented in the C++ programming language, with support of the OpenGL graphics library and integrated into an existing framework called VolumeShop [3]. A screenshot of the framework VolumeShop can be seen in Figure 4.1.

4.1 Building the Peel-Template

The peel-template is contains the inverse transformation of the displaced structures. In essence, it is a vector field containing the displacement vectors back to the original position of the sample. The vector field is a $64 \times 64 \times 64$ set of floating point four-dimensional vectors. This is then represented as a 3D RGBA texture with floating point precision. The displacement vectors are stored in the red, green and blue channels, and the definition of the discontinuous region is stored in the α -channel. The discontinuous region is the α -mask described in Equation 3.3.

To procedurally generate the vector field, an invertible transformation function is required. In this thesis, a rigid peel is chosen. The reason for this is that a rigid peel can be defined as a simple transformation matrix. This is an advantage, because matrices can easily be inverted. For a rigid peel the following transformation function is generated:

$$T_f(x, y, z) = \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4.1)$$

This is the transformation matrix for rotating an arbitrary point around the Y-axis in an euclidean coordinate system. T_f^{-1} , from Equation 3.1 can then be calculated analytically.

A challenge with directly building the inverse displacement map, representing the peel-template, is to define which points the inverse transformation should be applied to. To solve this problem the normal vector of the plane intersecting with the vector field, \mathbf{N} , is defined. Then the transformed normal is calculated by applying the forward

transformation on the plane normal, $\mathbf{N}' = T_f(\mathbf{N})$. Then the inverse transformation is calculated on all the points which are above the transformed plane according to the transformed normal. The discontinuous region is then defined as the points between the original plane and the transformed plane.

With these conditions the peel-template can then be generated procedurally at discrete positions with the following procedure; for all the positions in the vector field, detect if the current position is a valid transformed position. This is defined as true if the dot-product between the vector created from origo to the current position, and the transformed normal of the plane intersecting the vector field is greater than zero. If the current position has an inverse position, displacement vector is calculated as the difference between the current position and the inverse position. If the current position is

Algorithm 1: Algorithm for generating peel-template

Input:Transformation functions, T_f and T_f^{-1} ;Normal of the intersecting plane, \mathbf{N} ; $\mathbf{N}' = T_f(\mathbf{N})$;**foreach** Position $\mathbf{P}' = (x, y, z)$ in the inverse displacement map \overleftarrow{D} **do** **if** $\mathbf{P}' \cdot \mathbf{N}' > 0$ **then** $\mathbf{P} = T_f^{-1}(\mathbf{P}')$ $\overleftarrow{D}(x, y, z) = \mathbf{P} - \mathbf{P}'$ $A(\mathbf{P}') = 1$ **end** **else if** $\mathbf{P} \cdot \mathbf{N}' < 0$ and $\mathbf{P} \cdot \mathbf{N} > 0$ **then**

//Mark position as a discontinuous region

 $A(\mathbf{P}') = 2$ **end** **else**

//Mark position as not transformed

 $A(\mathbf{P}') = 0$ **end****end****Output:**Inverse displacement map $\overleftarrow{D}(x, y, z)$ α -mask $A(x, y, z)$

between the original plane and the transformed plane, the position is marked as a discontinuous region. Finally, if the previous conditions are not fulfilled, current position is marked as a non-transformed position. This procedure is shown in Algorithm 1.

After the peel-template is generated it is then loaded as a 3D texture and bound to a sampler3D in the fragment shader. In the next section we will describe how ray-casting and sampling is done on the GPU when using the generated peel-template to define the deformation.

4.2 Template-Aware Ray-Casting

After the template has been generated an bound to the fragment shader, template-aware rendering is required. For rendering an extended front-to-back ray-caster is used. The main loop in the ray-caster is described in Algorithm 2. The color and opacity accumulation process remains the same for both standard GPU-based ray-casting and template-aware ray-casting.

The rendering scene is defined by bounding structure which stores the entry and exit points for each ray. This structure is often referred to as the proxy geometry.

The process for each ray goes as follows; start and end points for each ray are retrieved from the proxy geometry and the ray traverses through the volume compositing color and opacity. If the current fragment becomes completely opaque, the ray is terminated, since no information beyond this point contribute to the final image.

A problem might occur for deformations outside the already existing proxy geometry since sampling is only performed within the rendering scene. To include displacements outside the normal rendering scene, the geometry of the peel-template is added to the proxy-geometry. The difference between the original and extend proxy geometries are shown in Figure 4.2.

The main difference between the template-aware ray-caster and a standard ray-caster is the sampling function, which is described in Algorithm 3. The standard sampling function retrieves the value from the volume at the current position. Using discontinuous displacement maps the position changes and discontinuous regions must be handled. Aligning the peel according the feature in focus is also handled in the sampling function.

Gradients at each position are calculated on-the-fly using the central differences method and the gradient is used as the normal for each sample.

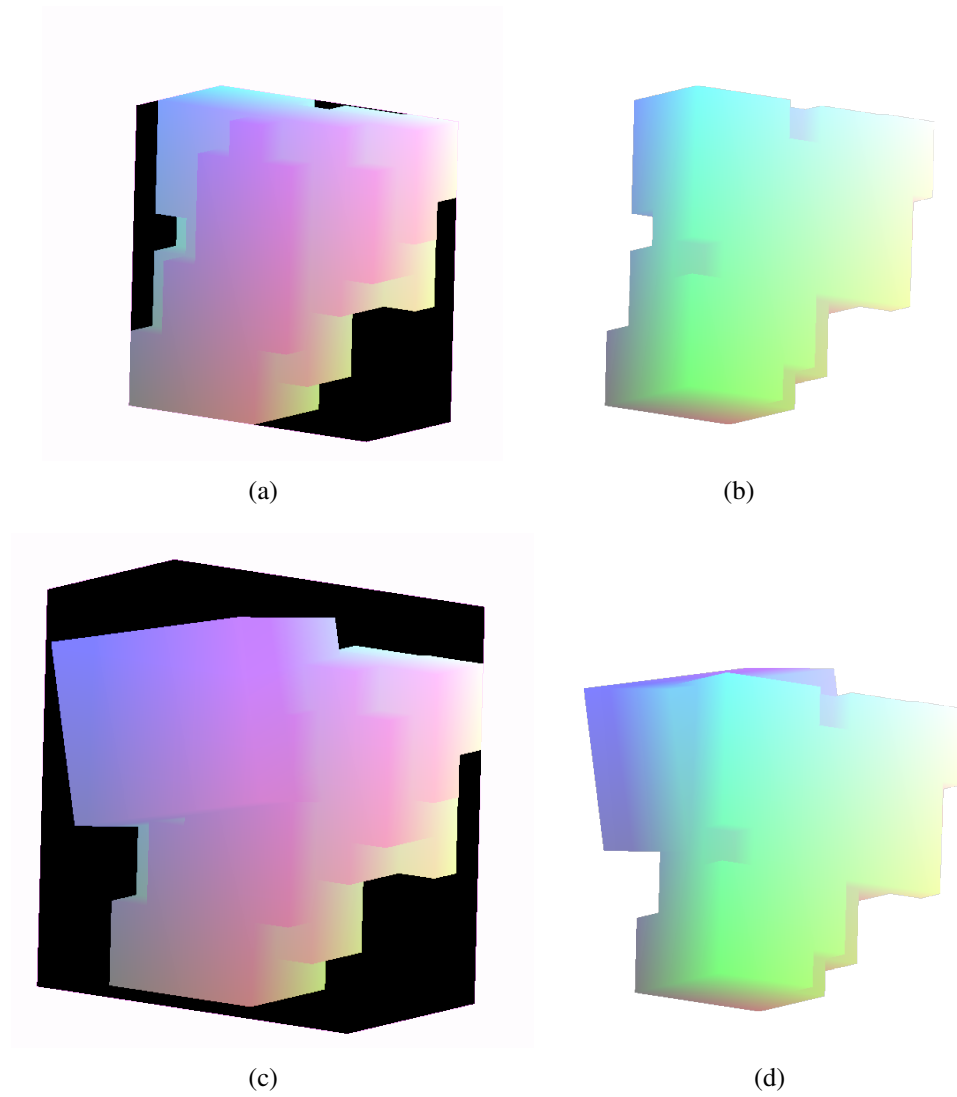


Figure 4.2: Entry and exit buffers of the proxy geometry generated for the hand with and without the peel-template. (a) and (b) show the original entry and exit buffers. (c) and (d) show the extended entry and exit buffers.

Algorithm 2: Main ray casting algorithm

```

while vecRayColor.a < 1.0 and vecRayPosition not at end do
  vecSampleColor = GetColor(vecRayPosition);
  if vecSampleColor.a > 0.0 then
    |   vecSampleColor.a = 1.0 - pow(1.0-vecSampleColor.a,fSampleDistance);
    |   vecSampleColor.a *= (1.0 - vecRayColor.a);
    |   vecSampleColor.rgb *= vecSampleColor.a;
    |   vecRayColor += vecSampleColor;
  end
  vecRayPosition += vecRayIncrement;
end
glFragColor = vecRayColor ;

```

The sampling process goes as follows; the first step is to determine if the current position is within the the peel-template. If this is true, the sample is retrieved from the current position. If this is false the next step is to handle peel alignment, the sample function handles feature alignment by detecting whether or not the current position is within the feature in focus. If this condition is true, sampling is done from the current position. Otherwise, it returns zero, since the feature should not be rendered at a transformed position. For axis aligned peels, the last step is ignored.

The next step has three different cases, the current position is not affected by the displacement, the current position is the transformed position of a voxel, or the current position is in a discontinuous region. These three cases are decided by the value of the α -mask, where 0, 1 and 2 defines the three cases respectively. In the first case, the sampling function returns the value form the current position. If the position is in a discontinuous region, the return value is zero. However, if the current position is the transformed position of a voxel, the displacement vector retrieved from the peel-template and multiplied with the transformation matrix to the peel-template. The vector is then added to the current position to calculate the original position. The sample is then retrieved from the new position. The algorithm for the sampling function is presented in Algorithm 3.

The peel-template transformation matrix are calculated from the scale, position and orientation. In the next section, the main process for calculating the position and scale for automatic feature presentation will be presented.

Algorithm 3: Sample retrieval

Data: peel-template transformation, M **GetSample(vecPosition)****if** *vecPosition* is not within the peel-template **then**

| return sample from current position

end $\text{vecVectorFieldPosition} = M^{-1} \cdot \text{vecPosition};$ $\text{vecVectorFieldSample} = M \cdot \text{texture3D}(D, \text{vecVectorFieldPosition});$ **if** *performing a feature aligned peel* **then**| **if** *vecPosition* is within feature **then**

| | return sample from current position;

| **end**| **if** $\text{vecPosition} + \text{vecVectorFieldSample}.xyz$ is within segment **then**

| | return 0;

| **end****end****if** $\text{vecVectorFieldSample}.a == 2$ **then**

| return 0

end**else if** $\text{vecVectorFieldSample}.a == 1$ **then**

| return sample from transformed position;

end**else if** $\text{vecVectorFieldSample}.a == 0$ **then**

| return sample from current position;

end**end**

4.3 Feature Projection

For the purpose of presenting features of interest, automatically creating a peel which removes the occluding structures is needed. Two steps are required; The first step is to position the peel according to the location of the feature in focus. Second, the peel-template has to be scaled according the size of the feature. Both scale and position are calculated from the projected feature onto the screen. Creating the projection of a specific feature can be done by a simple first-hit ray-caster. In essence, a ray stops traversing through the volume as soon as it hits the selected feature. The position is then stored in the RGB-channel of the output texture. To efficiently generate a texture of the projected segment we implement the simplified ray-caster on the GPU. See Algorithm 4. The result can be seen in Figure 4.3.

4.4 Interaction

Complete manual positioning, scaling and orienting can obtain the same results as the techniques presented in this thesis. A goal is to reduce the number of unnecessary low-level interaction, by suggesting a high-level interaction scheme.

To explore the data using peel-aways, the peel-template must to be positioned within the rendering scene. This is defined by the user with a simple mouse click on the rendered object. To locate the position, a single ray is cast towards the volume, the position is then set to the first surface hit by the ray.

Algorithm 4: Segment projection algorithm

```
while vecRayPosition not at end and feature not hit do  
  | vecRayPosition += vecRayIncrement;  
end  
if feature of interest hit then  
  | glFragColor.rgb = vecRayPosition.xyz;  
  | glFragColor.a = 1;  
end  
else  
  | glFragColor.a = 0;  
end  
;
```

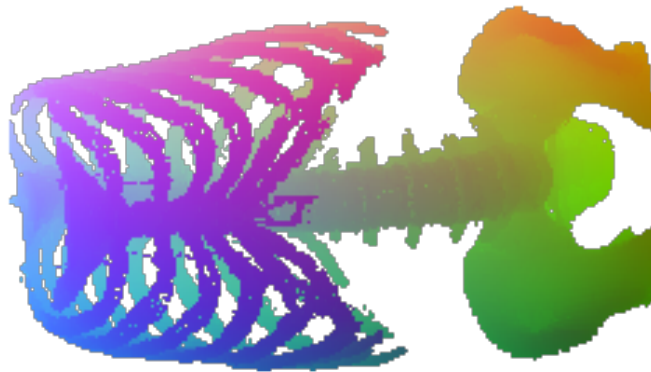


Figure 4.3: Projection of the bone segment from the Monster Study data set. The color encoding is the first-hit positions from the ray-caster.

Animating the displacement can provide a better understanding of the original position of the peel. This is done by updating the amount of degrees which the peel is rotated and recalculating the displacement vectors. The degree of displacement is adjusted with a slider in the graphical user interface. In essence this is the number of degrees the points are rotated around the y -axis.

For exploring the structures underneath the visible surface, the peel needs to be of a desired thickness. After the vector field has been positioned, thickness of the peel is determined by how deep the peel-template is in the volume. The user can adjust the depth according to the normal of the vector field with a slider.

If the feature of interest is larger than the peel-template itself or too deep within the surrounding structure, the peel may need to be resized. The size of the deformation can be adjusted by scaling the peel-template to a desired size. For an intuitive scale interaction, the scaling is done by left clicking on the template with the cursor and scaled by moving the cursor. The scaling direction is then defined by the position clicked on the peel-template and the center of the template. Positive or negative scaling is decided from the dot product between the scale direction and the direction of the cursor movement.

In this chapter, the implementations of the major processes for realizing view-dependent peel-aways on a real-time rendering system has been presented. In the next chapter the results and draw backs from the specified techniques will be discussed.

Chapter 5

Results

In the previous chapters, the techniques for generating peels dependent on the position of the viewer have been presented in both high-level and in detail. These techniques will be, in this chapter, demonstrated on several datasets from the medical domain.

- The hand dataset, see Figure 5.1.
- The Monster Study dataset, see Figure 5.3.
- The Bert dataset, see Figure 5.7(a).

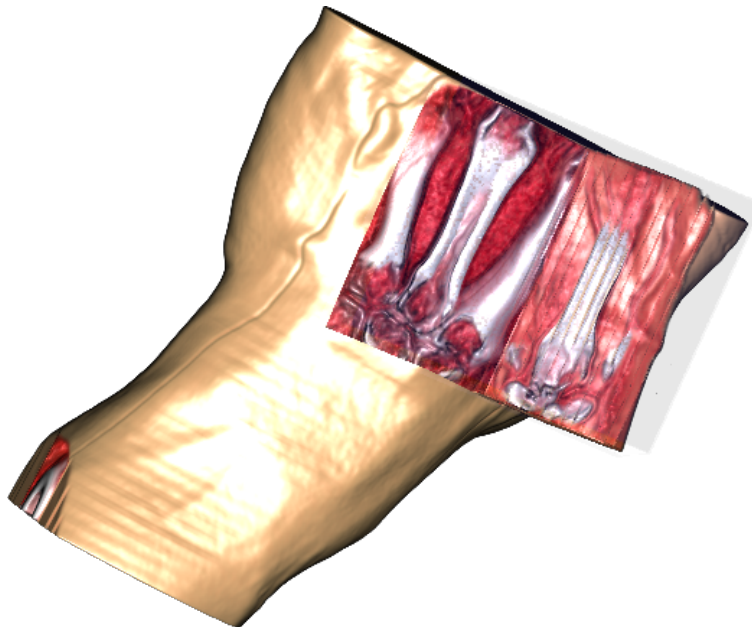


Figure 5.1: Inside the Metacarpal bone of the middle finger.

- The Head dataset, see Figure 5.4.

The hand, Monster Study and Head dataset are in courtesy of the AGFA development team in Vienna. The Bert dataset is obtained from the FreeSurfer distribution.

The purpose for the techniques described in this thesis can be divided into two different scenarios. One scenario is where the user seeks to gain insight into the data and to retrieve information about the structures within. A different scenario is where knowledge about the features within the dataset exists, for instance separate structures represented in a segmentation mask. Fast and easy presentation of these features is then desired.

The images created for the two different scenarios share certain specific characteristics. A blue box is rendered in the final images. This box represents the outline of the peel-template. This is included in the images to provide a better understanding about the position, size and orientation of the peel-template.

Different representations of different structures is useful to separate the structures from each other. For instance, the peel may become difficult to separate from the original volume. To provide an easier perception of the peel, a different transfer function can be chosen to render the transformed structures, as shown Figure 5.2. The same

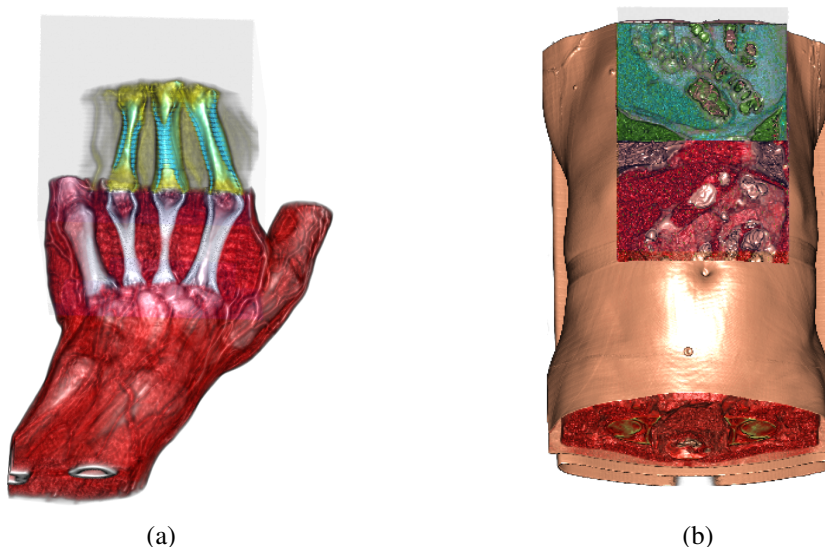
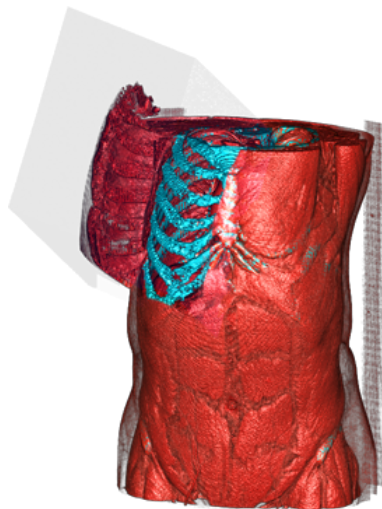


Figure 5.2: Different transfer functions for the peel and the original volume ensures the differentiation between them.



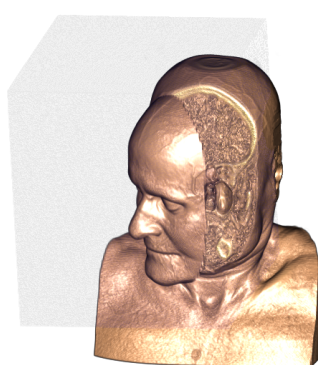
(a) Peeling away the muscles provides a clear view of the bone structures underneath.



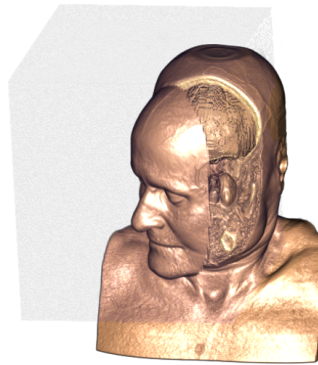
(b) Positioning the peel within the lunge allows the inner surface of the lunge to be visible.

Figure 5.3: Explorative peels.

problem can occur for presenting feature of interest. The feature can blend in with the surrounding structure. Thus, changing the representation of the feature of interest provides a better separation between the object in focus and the surrounding context. In Figure 5.3(a) the different representation of the bone structures, clearly separates it from the surrounding structures.



(a) Axis-aligned peel.



(b) Segment-aligned peel.

Figure 5.4: Different alignments

The alignment of the peel can enhance the perception features selected by the user. In Figure 5.4 the two types of alignment are rendered. While the axis-aligned peel allows the user to cut through the object, the feature-aligned peel enables investigation of both the surface of the object and the surface connected to the object.

5.1 Peel-aways for Exploration

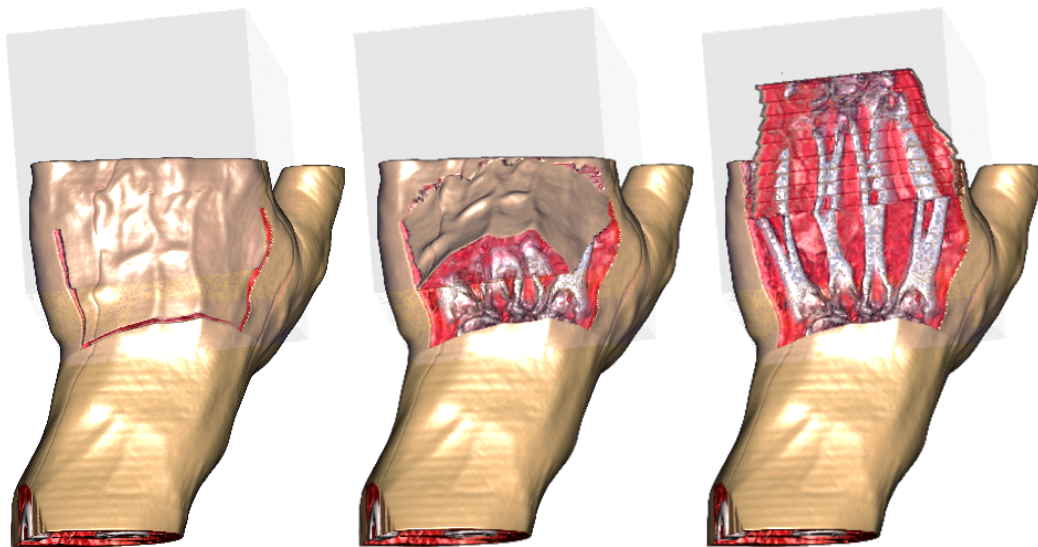


Figure 5.5: Explorative peel of the hand. Animating the peel provides better understanding of the original position to the peeled structure.

Exploration using peel-aways can be compared to exploration using a clipping geometry. The vector field work as a cutting geometry shaped like a cube. An advantage with peel-aways is that the structures clipped out are not removed. The peeled structures are in close proximity of their original position. In Figure 5.1 the inside of the Metacarpal bone in the middle finger is clearly visible. The inside of the bone is difficult to show using a regular transfer function. Also the back side of the cut is visible, which provide a good overview of the bone structure.

Since the resolution of the displacement map is small, the peel can be animated interactively. In Figure 5.5 the peel is animated from a small degree to a large degree. The animations provide a better spatial coherence between the original and transformed position of the peel.

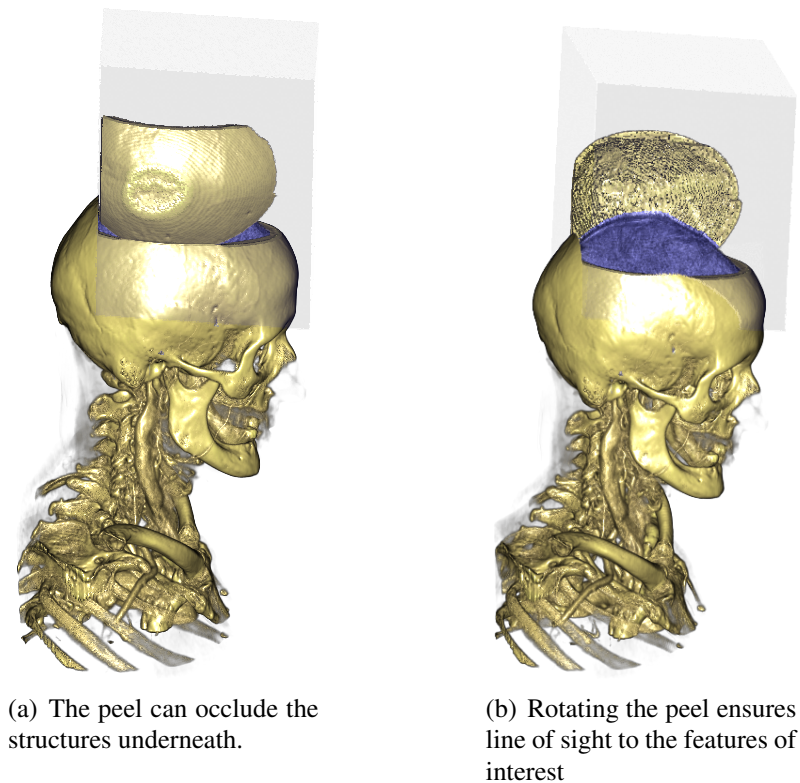
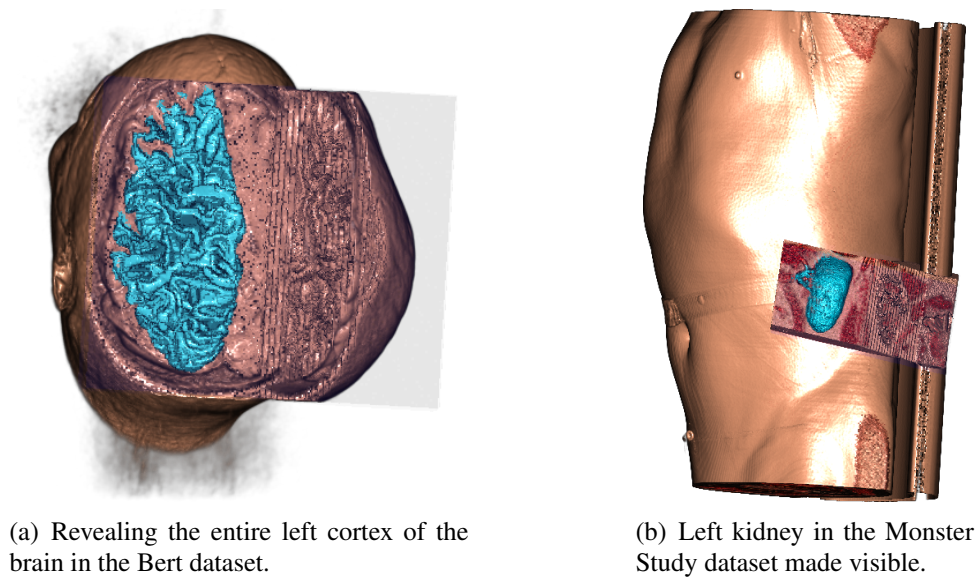


Figure 5.6: Rotating the peel ensures visibility of the features underneath.

With a static peel, changing the view-point can make the peel itself occlude the structures underneath. To prevent this the peel is rotated in such a way that the peel is facing away from the view-point. This is illustrated in Figure 5.6. The peels for both images are at the same position. When the scene is rotated the peel is oriented in such a way that it does not occlude the underlying structures.

5.2 Peel-aways for Presentation

Presenting structures using peel-aways is a method which has been used for a long time in traditional illustration. In scientific visualization, the automatic positioning and orientation of the peel allows for fast presentation of both the feature of interest and the surrounding structures. In Figure 5.7(a) the peel is scaled to include the entire left side of the brain. To show the left kidney a smaller peel is required, see Figure 5.7(b). The scaling function based on the size of the projection of the feature.



(a) Revealing the entire left cortex of the brain in the Bert dataset.

(b) Left kidney in the Monster Study dataset made visible.

Figure 5.7: Automatically generated peels to reveal features of interest.

5.3 Performance

In Section 1.5 one goal was that implementation of the techniques presented in this thesis should obtain interactive framerates. For a system to feel smooth and responsive it should at least obtain a framerate greater than five frames per second (FPS).

The prototype has been implemented on the visualization framework VolumeShop and the benchmark was done on the following system:

- Windows Vista Operating system
- Intel Core 2 Duo 6700, 2.66 GHz
- Nvidia GTX 280 w/1GiB dedicated memory.
- 2 GiB System memory

For this benchmark, rendering with different properties activated has been tested. A combination between the two different peel alignments and the two different interaction modes. Sampling distance was set to 1.0 for the performance testing and the size of the rendered image was set to 1328×852 . For comparison the column on the right

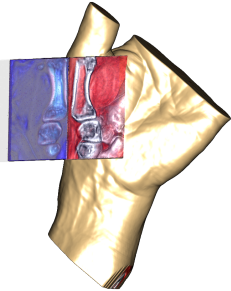
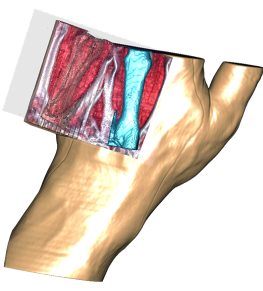
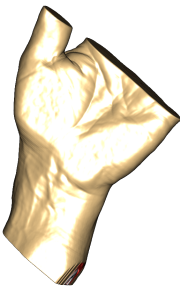
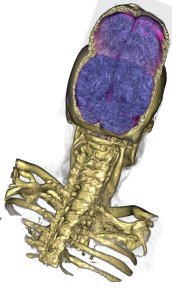
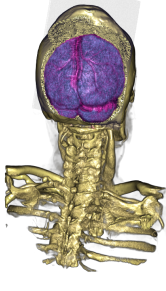
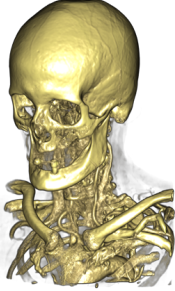
	Axis	Segment	DVR
Hand $244 \times 124 \times 257$			
Framerate	9 FPS	6-7 FPS	21 FPS
Head $256 \times 256 \times 230$			
Framerate	9 FPS	7 FPS	20 FPS

Table 5.1: Benchmark of manually positioned peel-aways.

shows images created using a regular sampling function in the ray-caster. The images in Table 5.1 and Table 5.2 have been resized to fit in the tables.

Table 5.1 show the benchmark results of manually positioning and scaling peels. The scale of the peel-template is set to $(0.6, 0.5, 0.6)$ for all images. This table shows that utilizing reasonable sized peels as a means for exploration can be achieved at interactive framerates. Segment aligned peels provide a good overview of the feature in focus, however it comes with a small reduction in performance.

In the images in Table 5.2, the automatic feature-aligned peel-away technique was used. Sampling within the peel-template is very time-consuming compared to sampling outside the template. This implies that the scale of the vector field has a great impact on the performance. From the table it is clear that scaling down the vector field, interactive framerates can be achieved for smaller features. However, for large features, such as the bone structure, interactive framerates are difficult to achieve due to the large scale of the template.

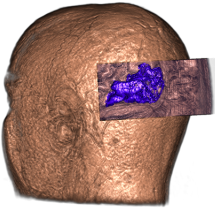
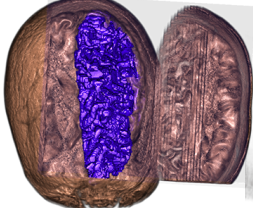
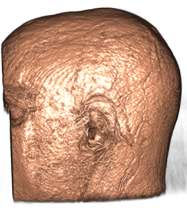
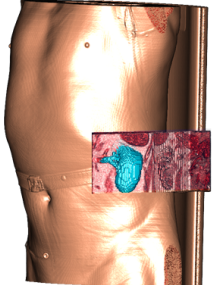
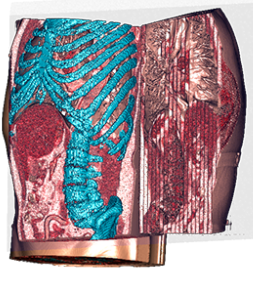
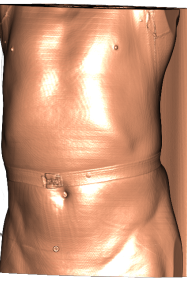
	Small peel	Large peel	DVR
Brain $256 \times 256 \times 256$			
Framerate	5-6 FPS	2-3 FPS	17 FPS
Monster $256 \times 256 \times 557$			
Framerate	5-6 FPS	1 FPS	15 FPS

Table 5.2: Benchmark of automatically generated peel-aways.

Chapter 6

Summary and Conclusions

Traditional illustrators have invented a great selection of tools to better convey the information about 3D structures to the viewer. When displaying 3D structures it is often the case that one structure occludes the other, where as the occluded one can often be of higher interest to the viewer. This creates the challenge of visualizing the features of interest, while providing an understandable representation of the surrounding structures. Techniques, such as peel-aways, cut-aways and exploded views, have been used by illustrators since the renaissance to solve the problem of occluding structures. In Figure 1.1, Figure 1.2 and Figure 1.3 early illustrations describing the human anatomy are shown.

Today, 3D structures can be scanned digitally and investigated on computers. A challenge, with for example anatomical CT scans, is to visualize the information in an easily perceptible way. Volumetric datasets are usually large, and a good overview can be difficult to obtain by 2D slicing. 3D representation of volumetric data can provide a better overview as compared to 2D slice rendering, however when providing a 3D representation of an object, the features inside can get occluded by the outer structures.

In recent years, traditional illustrations have inspired the field of illustrative volumetric visualization to incorporate the metaphors created by illustrators, into the field of scientific visualization to resolve the problem of occlusion. Various techniques in volumetric visualization can be used to explore volumetric datasets or present interesting features hidden within the data. 2D slice rendering is one of the main rendering types used in the field of medical visualization. Extracting a slice out of a CT scan at a given position, reveals all information at the specified area. However, overview is difficult to obtain with regular 2D slice rendering. Providing more information of the surrounding

structures can be helpful for various tasks in the medical scene. Figure 2.1 shows how extended 2D slice rendering can provide a better overview of the surrounding structures.

In 3D rendering, insights can be obtained by adjusting the opacity of the outer structures. Making outer surfaces completely transparent will show structures within the data. A challenge is to provide information about the spatial relationship between the outer surfaces, and the features of interest. In direct volume rendering, a transfer function is often used for mapping opacity and color to each voxel in the volume. With direct control over the opacity, the outer surfaces can be made semi-transparent and both the outer surfaces and structures previously hidden can be made visible.

Investigating the feature of interest can become difficult if a semi-transparent surface is rendered on top of the feature in focus, as shown in Figure 2.4. Other techniques suggests using the orientation of the surface as measure to determine the opacity of a structure. Contour rendering creates a representation of the outline of a surface by only rendering the areas where the normal of the surface is orthogonal to the viewer. This provides a clear view to features underneath the outer surface while providing information about the shape of the outer surface. Much research is done within the field of semi-transparency and contours to provide better insight in the data. Figure 2.3 show how contour rendering can provide information about shape of the surrounding structures, while revealing features inside.

While the use of transparency can gain insight into volumetric dataset, using global parameters for adjusting transparency does not always provide satisfying results. Using geometrical structures can be used to select certain areas in the rendering scene to be represented differently or removed from the image entirely. This process is known as clipping and cut-aways, which are used for exploration and visual communication respectively. Clipping is the process of removing structures, often defined by geometrical shapes, from the final image, as shown in Figure 2.7. Cut-away is the automated process of removing structures to reveal features of interest. The idea is to remove everything in front of the feature, according to the view-point. In Figure 2.9 cut-aways are used to show the liver.

Volume splitting is another technique for handling the problem of occlusion. Splitting is a technique inspired by clipping, where the volume is divided into several different parts. Each part can then be transformed and rendered separately. However, the different parts can still occlude the features of interest. Exploded views utilize the con-

cept of splitting to transform each part so that the feature in focus becomes visible, as shown in Figure 2.12.

Deforming an object can also be used to give focus to or to reveal, interesting areas. Deformation can be divided into to separate types which deal with different problems, continuous and discontinuous deformation. Continuous deformation, also called shape deformation, is distorting the shape of an object to enhance or reveal the interesting features on the surface of the object. Discontinuous deformation extends continuous by introducing cuts as a means to explore features underneath the surface. Realization of the two types of deformation can be divided into two. Object based deformation, in essence restructuring the data before the rendering process is started, and image based deformation, where the altering the shape is done in the rendering process. The latter type of deformation holds an advantage by keeping the original state of the data. Figure 2.14 show how object based deformation is applied to a colon for simulating a colon unfolding.

Defining deformations can be realized in several different ways. A novel approach developed by Correa et al. is using discontinuous displacement maps to define the deformation in the rendering process. Pre-defined templates which represent the deformation are positioned into the rendering scene to illustrate peels, separators and other types of deformation. In Figure 2.20 peeling is performed using a discontinuous displacement map. In this thesis, a template based deformation technique is utilized for fast exploration and automatically generating peels to reveal interesting features. The process is inspired by automatic cut-aways, where the feature of interest is automatically made visible from a user selected viewpoint.

Occlusion in volumetric rendering is handled with the use of various techniques inspired by traditional illustration methods. In this thesis, the peel-away metaphor is applied to volumetric datasets to explore and present features within the data. The two main goals are to make an intuitive method for exploring the data using peel-aways, and generating automatic view-dependent peel-aways for visual communication.

The peels are represented as pre-defined peel-templates. This is advantage for creating view-dependent peels compared to a physics-based peeling system, since time-consuming physics calculations are avoided. The peel-templates can be positioned, oriented and scaled to create the desired peel, without recalculating the displacement.

Visualizing peels using a pre-defined template can be divided into three major steps.

The first step for visualizing peels is to generate the peel-template. Second, the template needs to be positioned, scaled and oriented in the virtual environment, and third, the peeling must be handled in the rendering process to create the final image.

The peel-template is in essence a vector field which contain the inverse transformation of the peeled structure, as shown in Figure 3.2. To generate an inverse vector field, an invertible transformation function is required. The vector field is sampled at each point with the inverse transformation function according to Equation 3.2. For defining discontinuous regions in the rendering process, a binary α -mask is generated. The displacement vectors are then stored in the RGB-channel and the α -mask is stored in the A-channel of a 3D RGBA texture. Algorithm 1 describe how the process of generating a template representing a rigid peel.

The second step is to position, scale and orient the peel. This is divided into two parts; manual positioning and scaling for explorative peel-aways, and automatic positioning and scaling for presenting features of interest. The features of interest are defined by a pre-generated segmentation-mask which are stored as a separate volume.

For manual positioning the user positions and scales the peel with simple mouse gestures. However, to ensure a clear view of the peeled structures and the structures underneath, the peel is oriented in a way to always bend away from the viewer. This is to ensure that the peel itself does not occlude the interesting areas, as shown in Figure 3.8.

To reveal features of interest, the user does not control the position, scale or the bend of the peel. The user selects a feature of interest and the peel is automatically positioned and scaled according to the position and size of the feature. Size is determined by a projection of the feature, see Figure 4.3 onto the screen, and the position is calculated out of the extreme x and y positions of the projection.

The final step to visualize view-dependent peel-aways is rendering. The rendering is done by using a front-to-back ray-caster with an extended sampling function. Regular sampling retrieves values from the volume at the current position. For template-based peel-aways, the sampling function must consider the displacement of the peel. The sampling function first looks up the value of the α -mask for the current position in the peel-template. The value says if the current position has a valid inverse displacement. If the displacement is valid, the sample is retrieved from the original position, if the α -value suggests a discontinuous region, empty space is rendered. In Figure 3.5 the pipeline of the sampling function is described.

The sampling function also deals with peel alignment. The surface of the peel has two types of alignments. It can be aligned with the axis of the peel-template, or it can be aligned with a pre-defined feature. Axis aligned peels enable the user to cut through objects and examine the inner structures, while feature aligned peels provide a better overview of the structure in focus.

In volumetric visualization normals are used for shading 3D representations. The normal are approximated by the gradients. Gradients are calculated on-the-fly within the ray-caster using the central differences method. Previously this have been considered to be too time-consuming. However, due to the increase in performance of dedicated graphics hardware, the gradients can be calculated on-the-fly while still achieving interactive framerates. These three step combined enables visualization of peel-away, where the position, size and orientation of the peel consider the view point.

In this thesis, the peel-template represents a rigid peel. The rigid peel can be considered as plane cutting through the volume, and rotating all points around an axis. The rigid peel has an advantage with not preserving the spatial relationship for the structures within the peel. There are certain limitations with the approach described in this thesis. Stair-case artifacts appear on the cut-surface of the peeled structure. These artifacts appear because of the low resolution of the peel-template compared to the volume and because the discontinuous region is defined by a binary α -mask. Correa et al. suggested a solution for aliasing, by smoothing the α -mask. However, smoothing the mask is time consuming and reduce performance.

Because of the low resolution of the template, a template can be generated on-the-fly. This means that the peel can be animated. Animating the peel provides a better correlation between the peel and its original position, as shown in Figure 5.5.

Automatic peeling provides fast and easy presentation of features of interest, selected by the user. Figure 5.7 show peels created automatically to reveal the features selected by the user. While explorative peels provide a simple exploration technique for both the features of interest and surrounding structures.

Performance is an topic for consideration when creating interactive visualization systems. The general performance for explorative peels are high enough for a smooth interaction. For automatic peels, there is a reduction in performance due to the calculation of both the size and the position of the feature in focus.

6.1 Conclusions

The work presented in this thesis was done to satisfy the goals presented in Section 1.5. Techniques for generating illustrative peel-aways in volumetric visualization has been presented. The major focus has been to create peels which consider the position of the viewer. In the field of medical education, illustrative renderings are helpful to understand the human anatomy. Locating and investigating structures within the human body can be difficult using basic rendering methods.

The displacement method in the presented system utilizes discontinuous displacement maps to generate peels within the rendering process. Since this work did not aim at simulating physical interaction with the rendered objects, the pre-defined peel-template suites the purpose of positioning and orienting peels according to the view-point. The template based peeling has been proved to be capable of visualizing reasonable sized peels in direct volume rendering at interactive framerates.

The template used in this system represents a rigid peel. A rigid peel is generated by transforming the points around the y -axis. The forward transformation can be represented by a simple transformation matrix. In a GPU-based template-aware ray-caster, the inverse transformation is required for sampling from the original position in the volume. The transformation matrix can be inverted and the inverse transformation can be calculated. Because of the simple transformation calculation, and the low resolution of the peel-template and the corresponding α -mask, the peel-template can be calculated on-the-fly. Interactively generating the peel-template enables animation of the peel.

Due to the discrete binary α -mask for defining discontinuous regions, and the low resolution of the peel-template, stair-case artifacts appear at the surface of the peeled structure. Stair-case artifact distorts the view of the peeled structure, however it clearly conveys that the peel was artificially transformed.

The create more advanced and refined deformations additional work is required. Smoothing of the peel-surface and deforming the peel itself are fields which can improve the visual results for explorative and automatic peeling. In the next section, possible future extensions of the presented system are discussed.

6.2 Future Work

The techniques presented in this thesis provides a basis for exploration and visual communication using illustration techniques defined as templates. Although a rigid peel was selected in this thesis, different templates can be defined and applied using the same positioning and scaling system, and thus making it a very flexible illustration technique.

The work presented in this thesis does not support more than one peel at a time. A future endeavor would be to extend the system to handle multiple displacement templates.

For animation, the size of the template needs to be small enough to be able to generate the displacement vectors interactively. Today's hardware contains of usually of more than one CPU core, and generating templates is a parallelizable process since the displacement vectors does not influence each other. This means that a multi threaded process of generating templates can be implemented for larger, more sophisticated displacement templates.

Acknowledgments

I would like to thank the AGFA development team and Athinoula A. Martinos Center for Biomedical Imaging, for providing the datasets which have been used to demonstrate the techniques presented in this thesis. A special thanks to Ivan Viola, who supervised this thesis, for the great support and inspiration. I would also like to thank Stephan Bruckner, for providing technical support on the source code of VolumeShop.

Bibliography

- [1] A. V. Bartrolí, R. Wegenkittl, A. König, and M. E. Gröller. Nonlinear virtual colon unfolding. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 411–420, 2001.
- [2] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving volume rendering. In *Eurographics / IEEE VGTC Symposium on Visualization*, pages 69–76, 2005.
- [3] S. Bruckner and M. E. Gröller. Volumeshop: An interactive system for direct volume illustration. In *IEEE Visualization*, 2005.
- [4] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transaction on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.
- [5] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.
- [6] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions-A unified approach to specifying deformation in volume modeling and animation. In *Proceedings of Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 35–44, 2003.
- [7] R. L. Cook. Shade trees. *Computer Graphics Forum*, 18(3):223–231, 1984.
- [8] C. Correa, D. Silver, and M. Chen. Illustrative deformation for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327, 2007.
- [9] C. D. Correa, D. Silver, and M. Chen. Discontinuous displacement mapping for volume graphics. In *Proceedings of Eurographics / IEEE VGTC Workshop on Volume Graphics*, pages 9–16, 2006.
- [10] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M. E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3), 2001. ISSN 1067-7055.

- [11] D. DeCarlo, A. Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. volume 22, pages 848–855, 2003.
- [12] J. Diepstraten, D. Weiskopf, and T. Ertl. Transparency in interactive technical illustrations. *Computer Graphics Forum*, 21(3):317–326, 2002.
- [13] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. In *Proceedings of Eurographics 2003*, volume 22 of *Computer Graphics Forum*, pages 523–532, 2003.
- [14] M. Doggett and J. Hirche. Adaptive view dependent tessellation of displacement maps. In *HWWS '00: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics hardware*, pages 59–66, 2000.
- [15] M. C. Flannery. Artists and anatomists. *The American Biology Teacher*, 56(1):55–58, 1994.
- [16] D. M. Gavrilu. A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(8):1408–1421, 2007.
- [17] S. F. Gibson. 3D chainmail: A fast algorithm for deforming volumetric objects. In *SI3D*, pages 149–154, 195, 1997.
- [18] S. Gumhold and T. Hüttner. Multiresolution rendering with displacement mapping. In *HWWS '99: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 55–66, 1999.
- [19] S. Islam, S. Dipankar, D. Silver, and M. Chen. Spatial and temporal splitting of scalar fields in volume graphics. In *In proceedings of IEEE Volume Visualization*, pages 87–94, 2004.
- [20] S. Islam, D. Silver, and M. Chen. Volume splitting and its applications. *IEEE Transaction on Visualization and Computer Graphics*, 13(2):193–203, 2007.
- [21] Y. Kurzion and R. Yagel. Space deformation using ray deflectors. In *Eurographics Rendering Workshop 1995*. Eurographics, 1995.
- [22] Y. Kurzion and R. Yagel. Continuous and discontinuous deformation using ray deflectors. In *Proceedings of GRAPHICON'96*, pages 102–110, 1996.
- [23] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

- [24] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [25] M. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization*, pages 401–408. IEEE Computer Society, 2003.
- [26] J. Mensmann. Real-time deformation for illustrative medical volume rendering. Master’s thesis, Westfälische Wilhelms-Universität Münster Institut für Informatik, 2006.
- [27] A. Nealan, M. Müller, R. Keiser, E. Boxermann, and M. Carlson. Physically based deformable models in computer graphics. In *STAR Proceedings of Eurographics 2005*, pages 71–94, 2005.
- [28] C. Paloc, F. Bello, R. I. Kitney, and A. Darzi. Online multiresolution volumetric mass spring model for real time soft tissue deformation. *Lecture Notes in Computer Science*, 2489:219–??, 2002.
- [29] P. Rautek, S. Bruckner, M. E. Gröller, and I. Viola. Illustrative visualization: new technology or useless tautology? *SIGGRAPH Computer Graphics*, 42(3):1–8, 2008.
- [30] C. Rezk-Salama and A. Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum*, 25(3):597–606, 2006.
- [31] F. Schulze, K. Bühler, and M. Hadwiger. Interactive deformation and visualization of large volume datasets. In *GRAPP (AS/IE)*.
- [32] Florian Schulze. Direct volume deformation. Master’s thesis, University Koblenz-Landau, 2006.
- [33] K. Singh and E. Fiume. Wires: A geometric deformation technique. In *Proceedings of SIGGRAPH*, pages 405–414, 1998.
- [34] C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauß. Enhancing slice-based visualizations of medical volume data. In *IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 123–130, 2006.
- [35] F. Vallejo-Manzur, Y. Perkins, J. Varon, and P. Baskett. Andreas vesalius, the concept of an artificial airway. 2003.
- [36] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *IEEE Visualization*, pages 139–146, 2004.

-
- [37] S. J. Walton and M. W. Jones. Volume wires : A framework for empirical non-linear deformation of volumetric datasets. 2006.
- [38] S. J. Walton and M. W. Jones. Interacting with volume data: Deformations using forward projection. In *MEDIVIS '07: Proceedings of the International Conference on Medical Information Visualisation - BioMedical Visualisation*, pages 48–53, 2007.
- [39] D. Weiskopf, K. Engel, and T. Ertl. Volume clipping via per-fragment operations in texture-based volume visualization. In *IEEE Visualization*, 2002.
- [40] Q. Zhu, Y. Chen, and A. E. Kaufman. Real-time biomechanically-based muscle volume deformation using FEM. *Computer Graphics Forum*, 17(3):275–284, 1998.