TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

MASTERARBEIT

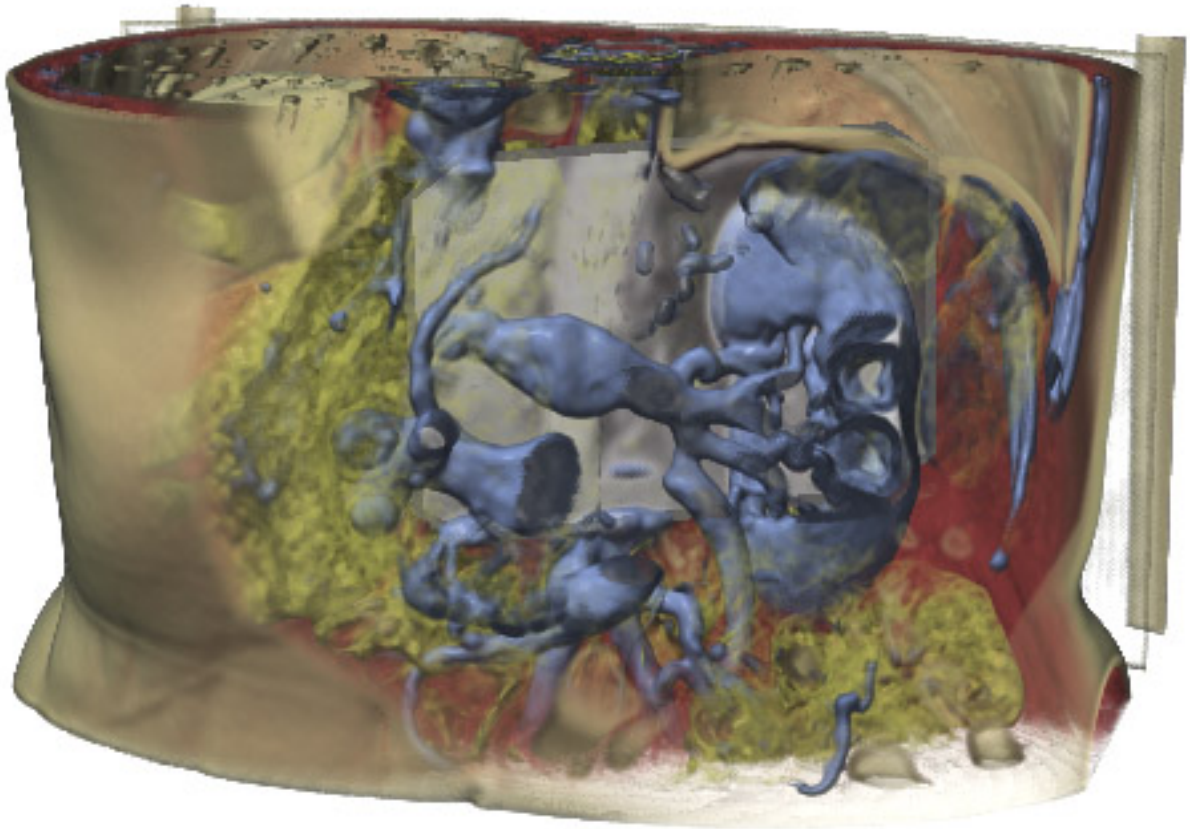
Importance-Driven Rendering in Interventional Imaging

Ausgeführt am Institut für
Computergraphik und Algorithmen
der Technischen Universität Wien,

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller,
Institut 186 für Computergraphik und Algorithmen,
und
Dipl.-Ing. Dr.techn. Ivan Viola,
Universität Bergen, Norwegen,

durch
Martin Haidacher,
Matrikelnummer 0025416,
Oberer Markt 11,
5661 Rauris, Österreich,
geboren am 24. März 1980 in Zell am See.

Wien, am 27. August 2007



Abstract

In this thesis a combined visualization of dense clinical data like 3D CTA (Computed Tomography Angiography) combined with co-registered real-time images of medical intervention applications is presented. The main challenge here is to provide a merged visualization that allows sufficient spatial perception of the important parts, as derived from the pre-operative data, while not occluding the information in the real-time image embedded within the volume.

This work presents a new approach of importance definition for volumetric data and how this importance can be used to create a feature-emphasized visualization. Furthermore the viewpoint and the position of the intervention image is used to generate a contextual cutaway which influences the density of the visualization to avoid an occlusion of the real-time image by less important parts of the volumetric data.

Kurzfassung

Diese Arbeit präsentiert eine kombinierte Visualisierung von dichten volumetrischen Daten, wie etwa von einer CTA (Computed Tomography Angiography), kombiniert mit co-registrierten Echtzeitbildern welche live während eines Eingriffes aufgenommen werden. Die Hauptaufgabe dabei ist es, eine Visualisierung zu erzeugen, welche eine ausreichende räumliche Wahrnehmung der wichtigsten Teile von beiden Eingangsdaten erlaubt. Es ist auch erforderlich, dass keine wichtigen Teile durch weniger wichtige Teile verdeckt werden.

Um das zu erreichen, wird ein neuer Ansatz für die Definition der Wichtigkeit im volumetrischen Datensatz vorgestellt. Es wird gezeigt wie diese Wichtigkeit verwendet wird um die wichtigen Teile hervorzuheben. Weiters wird die Position des Echtzeitbildes verwendet um daraus einen kontextabhängigen Ausschnitt zu erzeugen. Dieser Ausschnitt wird verwendet um die Dichte der Visualisierung zu beeinflussen damit das Echtzeitbild nicht von weniger wichtigen Teilen verdeckt wird.

Contents

Chapter 1

Introduction

One picture is worth a thousand words.

Fred R. Barnard

This thesis belongs to the research field of medical visualization. The general goal in this research field is to visualize data sets coming from medical screening devices. Such an imaging system can be, e.g., an X-ray. The algorithm for a medical visualization takes the raw data coming from the imaging system as input and produces a visualization which can be used by a physician to complete a certain medical task. The kind of visualization algorithm which is used for this task is, thereby, strongly dependent on the needs of the physician and on the used imaging system.

The more complex a data set is and the more information is included the more difficult it is to produce a visualization which points the doctor's attention to the relevant parts for a certain medical application. The complexity and the information content are highly related to the used imaging system. X-ray, e.g., produces a single two-dimensional image of a part of the body by projection. More recent medical imaging systems acquire three-dimensional data sets from internal structures of the human body. Such imaging systems are, e.g., computed tomography (CT) or magnetic resonance imaging (MRI). These three-dimensional data sets contain more information than only a two-dimensional plane. Therefore it is possible that some parts, which are important for a doctor, are occluded by other parts. Due to that reason it is necessary to find a visualization which shows all important parts to the physician without any occlusion.

In the last decade the complexity of the data sets has increased a lot. Even low-dimensional imaging systems, such as ultrasound, are nowadays used to produce three-dimensional data sets, or also called volume data sets or volumetric data. Besides the extension to the third dimension also the density of the data has increased. This development needs more than ever before good visualization techniques to support physicians in their medical tasks. In Chapter 2 an overview of recent visualization techniques for volume data sets is given. Besides the visualization, or also called rendering, of volumetric data there are a lot of other research topics in the field of medical visualization.

The topic of this master thesis, importance-driven rendering in interventional imaging, combines two different research topics of medical visualization. The importance-driven rendering is a visualization technique that gives visual prominence to special parts in contrast to the rest of a data set. The overall goal is to produce a visualization where the important parts are not occluded by the less important parts. A more detailed introduction to this research field and an overview over existing methods can be found in Section 2.2.

The second part of the master thesis title is about interventional imaging. Interventional imaging is the name for a medical intervention with a live image of the field of activity. The live image can be from an ultrasound probe, from a CT scan, or from MRI. Which imaging technique will be used depends on the kind of intervention and how deep the field of activity is in the body. Interventional imaging is normally used for minimal invasive surgeries. In such a surgery the physician is operating only through small cuts in the patient. Therefore he is not able to see the field of activity directly but with the imaging system he can see what he is actually doing. The benefit of a minimal invasive surgery for the patient is a faster healing of the surgery wounds because they are much smaller as compared to a conventional surgery. On the other hand the physician has a penalty because he has a limited sight on the field of activity through the imaging system. So it is very important that the imaging system shows him all the important parts for his task without any time delay.

Besides the imaging during an intervention some additional information is used to avoid any injuries of the patient. For this reason the intervention is planned in advance by using other imaging systems which cannot be used during the intervention. The data used for this planning is also called pre-operative data because it is recorded before the operation. So all steps which are necessary for a minimal invasive surgery are: The recording of a pre-operative data, the planning of the surgery on this data, and the execution of the intersection with the knowledge from the surgery planning and the usage of the interventional imaging.

The recent developments in the last few years in hardware and software brought some new possibilities for interventional imaging. The graphics hardware is much faster so even

complex visualizations can be done now in real time. Furthermore there were some algorithms developed which can register the image from the interventional imaging system to the pre-operative data in real time as well. These developments can be used together to merge the pre-operative data with the interventional imaging and show both together during the intervention in real time. The topic of this thesis is how this can be done in detail and which algorithms are needed to get a visualization that can be used during the intervention.

The aim of the combination of both input sources is to quicken the planning task because in a merged visualization the information from the pre-operative data can be seen directly during the intervention. To use this benefit it is important that there are no additional penalties coming with the steps needed to create the merged visualization. First of all it is important that a physician is able to see the most important parts of both sources at once. Less important parts from the pre-operative data should not occlude the image coming from the imaging system used during the intervention. Furthermore all steps to get the final visualization during the intervention should be applied on the input sources without the need of a user interaction. This is very important for a practical use because physicians are not interested in spending time for adjusting, e.g., parameters of an algorithm.

In the following work algorithms are introduced which lead to the desired visualization with avoiding any additional user interaction in a practical use apart from employing traditional devices such as the ultrasound probe. With a kind of importance-driven rendering all important parts of both input sources will be visible at once in the final visualization. Before concentrating on the algorithms the next chapter will introduce a special application for interventional imaging because the approaches in this thesis are mainly designed for this particular application.

1.1 Liver Intervention

Interventional imaging is used for many different surveys and interventions. The approaches in this thesis concentrate on interventions in the liver. As imaging system during a liver intervention typically ultrasound is used. Only in a few cases X-ray or MRI is used when the field of activity is too deep inside the body to see it on ultrasound. If it is possible to see the field of activity in an ultrasound then it is the best choice because ultrasound does not expose the body to radiation as compared to X-ray. Furthermore it is easier to handle in contrast to MRI and X-ray because the acquisition of the data is done by a small probe. This makes the handling for a physician very comfortable and allows a movement in every arbitrary direction. Because of these benefits and the fact that it is very rare that a field of activity in the liver

is outside the range of ultrasound this thesis considers ultrasound as imaging system during the intervention. Nevertheless it is no problem to apply the approaches to any other imaging system besides ultrasound.

There are mainly two applications for which this kind of interventional imaging is used in the liver. These two applications are the biopsy and the radiofrequency ablation of liver tumors. In both cases the physician uses a needle which has to be placed in a certain region inside the liver. In the case of the biopsy a tissue specimens is being taken from a suspicious mass. It is the only way to make an accurate diagnosis. The radiofrequency ablation is used after a tumor has been found to destroy the tumor tissue through high temperature at the top of the needle. The interventional imaging is needed to guide the physician while he tries to place the needle in the suspicious tissue.

The most crucial part during this intervention is the insertion of the needle. On the way to the target area the needle must not injure any vessel inside the liver. If the physician wounds a vessel during the insertion then an emergency surgery is necessary to stop the bleeding.

Figure 1.1 shows a schematic visualization of the liver and its position in the body. The liver is the largest inner organ. It has a number of functions such as drug detoxification or glycogen storage. Most of the blood supply of the liver comes from venous blood and therefore from other organs such as stomach and pancreas. Only 25% is coming from the heart directly. Overall almost a third of the total blood within the body travels through the liver each minute. This is the reason why it is very crucial to not injure any vessels during the intervention.

The needle is normally introduced on the right side of the body (see Figure 1.1) below the ribs or between the ribs. This intervention is a minimal invasive intervention. In the usual case no general anesthesia is needed and the overall duration of the surgery is only a few hours. Most of this time is used for waiting to see if there is an internal bleeding caused by the intervention. Agrafiotis *et al.* [1] write in their work that 0.4% of patients with liver cancer can develop a fatal hemorrhage due to puncturing of the blood vessels supplying a tumor. With a better guidance of the physician during the intervention this rate could be lowered.

During an intervention the physician has only the information from a single image plane coming from ultrasound. By moving the ultrasound probe over the region of the actual position of the needle the physician is able to see the position of the needle in the body and the vessels which are around the top of the needle. As mentioned in the introduction, typically the data of another imaging system is used for planning the surgery in advance. In this data for a path through the vessels is searched. In the intervention itself the physician

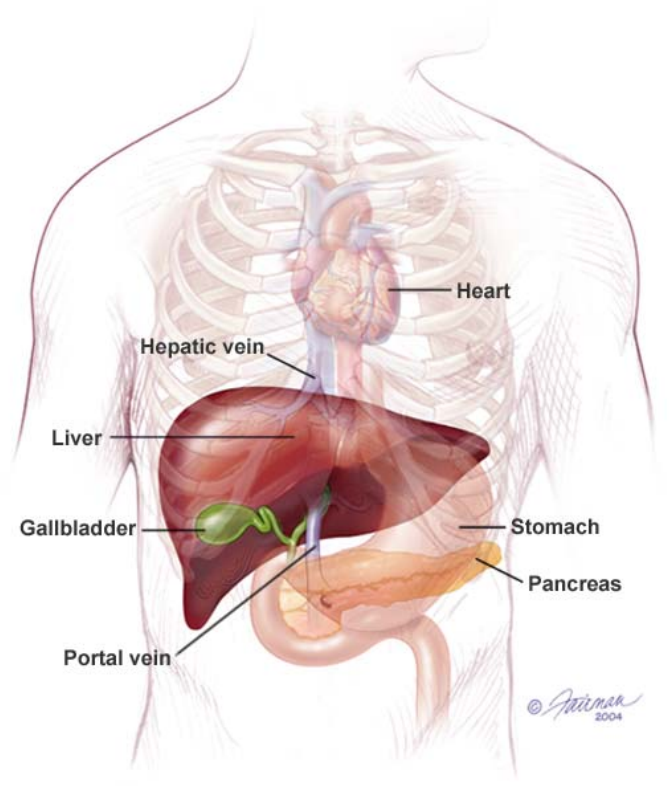


Figure 1.1: Schematic visualization of the liver and its position in the body (Source: Massachusetts General Hospital Cancer Resource Room, Boston, MA)

tries to insert the needle along the planned path with the support of the ultrasound. As pre-operative data for liver intersections normally a volume data set is used as described in Agrafiotis *et al.* [1] and Glombitza *et al.* [16]. In these works they describe techniques how the volume data set can be visualized for an easy needle path planning by emphasizing the vessel structure and the tissue of interest such as a tumor.

There are mainly two different imaging systems which are able to produce volume data sets from the liver where the vessel structure can be extracted. One of these techniques is MRI. It has the benefit that it has no unhealthy radiation. The penalties of MRI are a low contrast in the resulting data for tissues which do not contain much fluid and the acquisition of a data set with MRI is more expensive as compared to other techniques. The second imaging system which can be used to record the pre-operative data for liver surgery is computed tomography angiography (CTA). CTA is a often used technique which uses X-rays to visualize blood flow in arterial or venous vessels. It can be used for several kinds of examinations such as investigating blood vessels in the brain or in the kidney. To highlight the vessels in the recorded data set an additional contrast agent is needed. This contrast

agent is injected in a vein of the patient. After the injection it takes a moment before the contrast agent reaches the region of interest. This period is depending on the vessel length from the injection point to the region of interest. For a liver surgery it lasts approximately 30 seconds when the injection is done through the arm. In the computed tomography which is done after this time period the arterial vessels in the liver are highlighted.

Due to the fact that CTA is more often used than MRI the approaches are designed for data sets which are recorded by CTA. Nevertheless they also work with MRI what can be seen in the results in Chapter 5.

In the following Table 1.1 both input sources which are used for the entire liver intervention are compared. As one can see the benefits of one technique are somehow penalties of the other technique and vice versa. So joining the information from both techniques combines the benefits of them and avoids the penalties which each of the technique would have if it is being used alone.

	Ultrasound	CTA
Benefits	• Very fast, high refresh rate • No unhealthy radiation • Easy and flexible to use	• Three-dimensional view of the body • Can be very accurate • Vessels are enhanced • Shows the entire body at once
Penalty	• Low contrast through perturbations • Occlusions through reflections of the ultrasound signal • Limited depth range in the body	• Unhealthy radiation • The preparation and recording takes a while • Use of a contrast agent which can cause allergic reactions in rare cases • Static image

Table 1.1: Benefits and penalties of ultrasound imaging and CTA

So the usage of CTA as pre-operative data and ultrasound as imaging system during

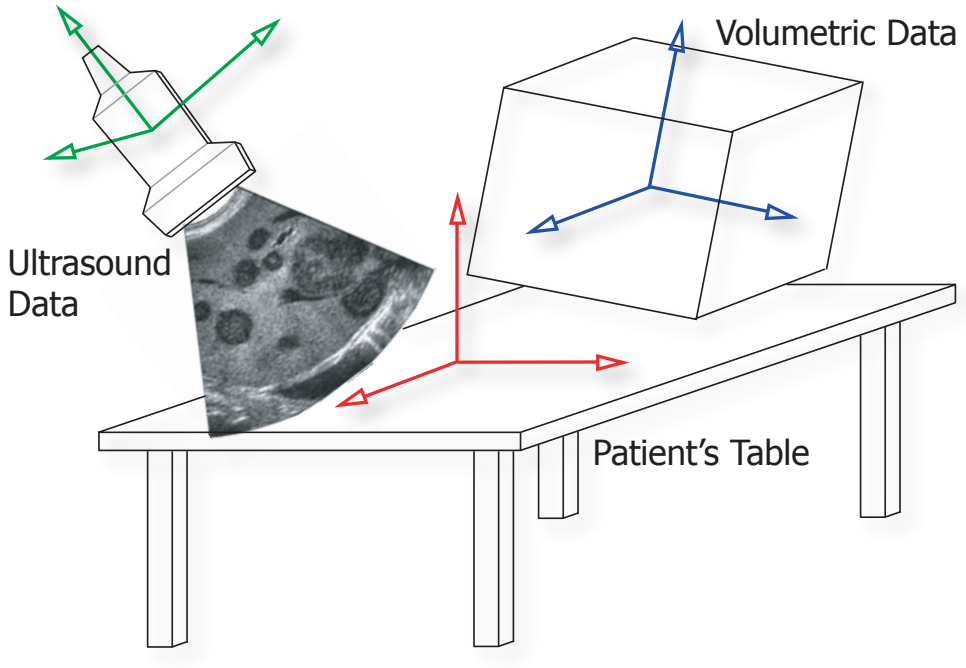


Figure 1.2: Different coordinate systems which have to be merged in the registration step.

the intervention is a quite good combination. The only problem in the entire process is the planning of the surgery before the intervention. The planning is done with the volumetric data coming from the CTA. With a good visualization of the CTA data it should not be a problem to find a straight path through the vessels to the suspicious tissue which has to be examined. The greater problem is to reproduce this path in the human body during the intervention because besides the ribs there is no real reference point between the body and the volumetric data.

This is the point where the new approach can be brought in. As mentioned before the idea of the approach in this thesis is a fused visualization of the ultrasound image together with the volume data set. This fused visualization can be used directly during the intervention. The significant benefit with this is that there is no need for a separate needle path planning because the path can be seen directly in the fused visualization. The first step for creating such a visualization is the registration of all parts to have them in a corresponding relation. Figure 1.2 shows all parts which have to be merged for creating such a fused visualization.

All parts in the figure have their own coordinate system. Without a registration there will be no positional relation between these parts. The goal of the registration is to bring the volume data set in accordance with the ultrasound image and the patient's position. The registration used for this thesis is based on the approach of Wein *et al.* [45]. This approach

uses a magnetic tracking system for the ultrasound probe. With this tracking it is possible to bring the coordinate system of the ultrasound probe in the coordinate system of the patient's table. This setup is only done once for a single operation environment. The registration of the volume data set is done for each patient separately. For this reason a sweep of ultrasound images from the patient is taken when he is lying on the patient's table. The algorithm developed by Wein *et al.* takes these images and aligns them with the volume data set. From the position of the images in the volume data set and the information of the position of the probe when these images were taken it is possible to register the volume data set to the coordinate system of the patient's table. So, finally, both input sources, the ultrasound image and the volumetric data, are in the same corresponding coordinate system which matches with the coordinate system of the patient's table. This alignment of the volume data set is done once for each patient. Besides the recording of the ultrasound sweep no further interaction is needed from a physician for this alignment.

After this registration we know for each position of the ultrasound probe the corresponding position inside the volumetric data. The volume data set from the CTA can be seen in this scenario as fixed and the ultrasound image is moving around in this data. This setup leads to completely new possibilities for the visualization during the intervention. With the registration not only the information inside the ultrasound image is changing, also the position of the ultrasound plane itself is changing. When now the volume data is visualized with the ultrasound plane then the physician has the possibility to virtually explore the volumetric data with the ultrasound probe. Besides the possibility of a new visualization it also brings a complete new possibility for the user interaction. A physician is then able to see the position of the ultrasound plane on the screen together with the information in the ultrasound plane. Therefore the visualization can be used for the navigation of the ultrasound probe on the patient's body.

In this thesis we will only concentrate on the new possibilities for the visualization which is possible through the registration of the input sources. As mentioned earlier the information which a volume data set contains is enormous. For the needle path planning the position of the target such as a tumor and the vessels inside the liver are really important. The target, thereby, is marked by a physician before the surgery. The rest of the volume data set is not that important for the task but it is needed to get an impression where the ultrasound plane is positioned inside the volume data set. Therefore the goal of the visualization is to show the most important parts with keeping an overview of the less important parts. To find the right visualization for this medical task it is necessary to know which parts are the most important ones during the intervention. Figure 1.3 shows an importance ranking of all parts for a liver

intersection.

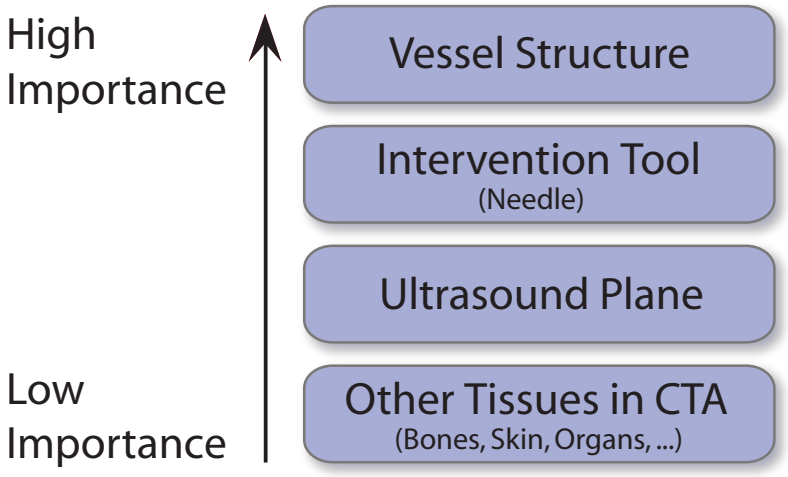


Figure 1.3: Importance ranking of tissues and import sources

The most important part during the intersection is the vessel structure inside the liver. It should be always visible and not occluded by other parts. The intersection tool is the second most important part. For a liver intersection the tool is typically a kind of needle. In the further work the visualization of the needle is not taken into account because it does not affect the visualization technique and it is no problem to add it in the final visualization process. During the intervention it will be also tracked by a magnetic tracker to register it to the coordinate system of the patient's table. In the importance ranking the ultrasound plane follows the intersection tool. In the further work the ultrasound plane will also be referred to as region of interest or object of interest to recall that the algorithm can also be used for other imaging systems beside ultrasound. The least important parts are the tissues around the vessels in the volume data set.

Now everything is defined which is needed for the development of the new approach. The following section will close the introduction chapter by giving an overview on the structure of the further work.

1.2 Structure

In the following Chapter 2 an overview on volume rendering is given. The most important algorithms in this research field are briefly shown. In Section 2.1 volume rendering algorithms are shown which can be used for real-time rendering. Common techniques are shown which make the rendering faster to get higher frame rates. This is also important for the new

approaches in this thesis because the final visualization has to be in real time for a practical usage. Section 2.2 presents the recent developments in rendering techniques to highlight important parts in a volume data set. The most important developments are briefly explained.

The entire Chapter 3 concentrates on the new techniques which are needed to get the fused visualization of the two input sources. Sections 3.1, 3.2, 3.3, and 3.4 explain all the processing steps which are applied on the pre-operative data. In Section 3.5 a new cutaway technique is introduced which is used to cut away the parts which might occlude the region of interest inside the volume. Section 3.6 combines the new visualization technique of the pre-operative data with the cutaway structure to get a fused visualization which shows only the most important parts.

In Chapter 4 a short overview on the implementation of all algorithms is given. It shows the creation of the cutaway view and which rendering technique has to be used. For both of these steps it is very important to be fast to achieve the real-time request. The usage of graphics hardware plays an important role thereby.

The final results after applying all the processing steps from Chapter 3 on data used for liver intersections are shown in Chapter 5. It compares the result from some different data sources and shows how the new approach can be used in practice.

In the final Chapter 6 a summary of the entire thesis is given. Furthermore the chapter shows what can be done in the future to improve the algorithm. It also gives an outlook how this new approach can be brought to a daily practical use.

Chapter 2

State of the Art in Volume Rendering

If we knew what it was we were doing, it would not be called research, would it?

Albert Einstein

In this chapter an overview over different volume rendering methods is given. In Section 2.1 a closer look on hardware-based rendering techniques is done. This is necessary because a fast visualization is mandatory in the interventional imaging application. Furthermore in Section 2.2 an overview over recent research in the importance driven rendering is given. First some basics in volume rendering or more general in volume visualization are described. Before even starting to write about visualization and rendering the volume itself and its origin will be considered.

The so called volumes which are used as input for this kind of visualization can be generated through different methods and by different tools and devices. Kaufman [22] wrote that the primary sources of volumes are three: sampled data of real objects or phenomena, computed data produced by a computer simulation, or modeled data generated from a geometric model. For all different sources the spatial dimensionality is three such as in the physician's definition of a volume. If there is exactly one value at a single position in the volume space then it is also called scalar field in contrast to a volume where two or more different values are assigned to a single position. If so then it is denoted as vector field. An example for a vector field is a meteorologic data set where temperature and air pressure is given at one position. A second big difference is that input sources can be either discrete or continuous. A

continuous data set is typically based on a function and is generated synthetically. It is being used for phenomena which does not exist or are hard to simulate in reality. This can be for example a generated weather model. The discrete volumes are mostly from a measurement of an existing phenomenon. The measurement is done on discrete points. Normally there is no knowledge about the values between measured points and, therefore, these values must be estimated.

In volume visualization all different kinds of volumes can be used. In the special case of medical volume visualization typically discrete scalar fields are used as input source. For this reason, only this kind of input data will be considered in this thesis. Nevertheless there are still several methods how to generate this kind of volume. The most common procedures in medicine are:

- Computed Tomography (CT)
- Magnetic Resonance Imaging (MRI)
- Positron Emission Tomography (PET)

All three methods are generating a physical value for a three dimensional position. The meaning of this value is different. For CT the value is equivalent to the physical density of a point position in the body. For additional information about this topic see Novelline [36].

The scanning procedures do not create a volume directly from the scanning process. There are some steps necessary to reconstruct a volume from a medical imaging device. The reconstruction, thereby, is also dependent on the used technique. In the case of CT the volume is reconstructed from cross sectional images which are recorded through the X-ray technique during a rotation around the patient. The reconstruction process can also include filtering and resampling to reduce the errors from measurement and reconstruction and to get well-distributed points within the volume. For more information see Hounsfield [21]. To get a volume with this procedure more than one scan is necessary. For this reason the patient or the scanning device is moved forward or backward a small distance to do another scan. This produces a stack of images along the patient's body. One of these images is also called a slice. The slice number in the stack of images is the third dimension of the volume besides the x and y direction in a single image.

In this kind of medical data set the points are mostly arranged in a regular grid. This means that from each point in the volume the vectors to the adjacent neighbors in the three dimensions x, y, and z create a Cartesian coordinate system with axes parallel to the Cartesian

coordinate system of the entire volume. A second condition for a regular grid is that the distance between two neighboring points is constant in each of the three main directions.

A single point in the volume is also called voxel. This name comes from volume element according to pixel as short version of picture element. The region between eight adjacent voxels is called cell. In the literature two different definitions for the size of a voxel and a cell can be found (see Elvins [10]). One definition considers the size of a voxel not only as point but as a region of constant value around the corresponding measurement position. In this case the size of a cell is zero because the voxel regions are filling the entire volume space. In the second definition the voxel is not a region it is just the point at the sampling position. A cell is in this definition the space between eight neighboring voxels. The difference between these two definitions is the way how to handle the space between sample points. In the first definition where a voxel is considered as a region the value is constant in the entire region. In the second definition where cells are between voxels the value is not defined. An interpolation can be used to calculate values for points inside a cell. Which of the two definitions is used depends on the used visualization method. In the further thesis the voxel will always be considered as a single point and the cell is the area between.

This definition brings the possibility for the interpretation of values between the voxels. The assumption in this model is that the values in the space between two neighboring voxels are continuous. For some visualization techniques the value for a point between voxels is needed. Therefore different kinds of interpolations are used to calculate the values for these points. The usage of interpolation for these points will provide a continuous change of the values for points between voxels. Some typical kinds of interpolations are briefly described later in this chapter.

Besides the interpolation the assumption of a continuous change of the values between voxels brings another possibility: extraction of derived properties. Such derived properties are, e.g., surface orientation or curvature. The surface orientation which is expressed by the normal vector is often used for calculations such as light reflection in volume visualization. This is something which is used by a lot of visualization algorithms. Curvature and other higher-order features are used more rarely.

Now we know where the volume comes from. The next step is the visualization of this volume. Due to the fact that a standard screen is only able to display a two-dimensional image a technique is needed which generates a two-dimensional image out of the volume data set. This technique is called volume visualization or volume rendering. In general the entire process can be seen as a mapping from the three-dimensional space to the two-dimensional space. The three-dimensional space is also called object space and the two-dimensional space

is also called image space. The result image after the projection is dependent on the position of the image space in relation to the object space, the projection settings, and the used visualization method.

Table 2.1 shows a classification of volume visualization methods. The subdivision in volume rendering and surface rendering is the most significant difference in the methods. Surface rendering, or sometimes also called model rendering, uses a certain data representation (e.g., polygonal) generated from the volume data as basis for the rendering. This model represents a specific iso-surface of the object. There are several algorithms which have been developed to create such a surface model.

Volume Visualization		
Surface Rendering	(Direct) Volume Rendering	
	Object-Order	Image-Order

Table 2.1: Classification of volume visualization methods

One of the first invented methods for surface rendering was the *contour-connecting algorithm*. It is a very simple but powerful method which is still in use in several research areas to create three dimensional models from contours. It was introduced first by Keppel [23] and improved later by Ekoule *et al.* [9] and Fuchs *et al.* [13]. The basic idea is to find closed contours in each slice of the volume and afterwards connect them between the slices with geometric primitives. The finding of the contour works usually with a threshold value. First the algorithm tries to find a point with a certain threshold value in one slice. The first found point is then used as starting point for finding a contour with values close to the threshold. This contour finding can be done either by hand or automatically. An automatic contour finding does not always work well especially if the contrast of the image is very low.

When these contours are found in each slice then the next step is to connect them between the slices. For this reason geometric primitives such as triangles are used. The gradient of cells can be used to find the point in the next slice which should be a neighbor on the resulting surface.

Another older algorithm for surface rendering is called *opaque cube* or *cuberille*. It was introduced by Herman and Liu [20] and has been optimized by many others. In general the user has to select a threshold first. The entire data set is traversed with this threshold. If one corner value of a cell is lower than the threshold value and another corner value is higher then the cell is selected for rendering. For this reason six quads are generated for each face of the cell. If there is more than one threshold value then the quads can be drawn in different

colors and opacities for each different threshold value. This makes it possible to see differences between tissues. The resulting surface for one threshold is also called iso-surface because it represents a surface of cells with almost the same value. The penalty of this algorithm is that small features in the data are hard to visualize and the resulting surface doesn't look very smooth because it consists of cells.

A third and more recent method for surface rendering is called *marching cubes*, introduced by Lorensen and Cline [34]. It is a more accurate method than the two introduced algorithms before because the size of a single primitive can be much smaller. For the generation of this model four adjacent slices must be loaded in the memory. For all grid points in the middle two slices the gradients are calculated. After that, all cells between these two slices are traversed. If one corner value is higher and another is lower than the threshold value then one or more triangles are fitted into this cell regarding on the gradient directions and values at all eight corner points. All other cells will be discarded.

For each edge of the cell the threshold can be in the value range between the two belonging corner points or not. This results in 256 possibilities to cut a cell by triangles for all combinations with the 12 edges of a cell. This number of possibilities can be reduced to 15 cases if rotation and reflection can be applied to a cell. If it is assumed that one edge can only be cut once by a triangle then the maximum number of triangles for one cell can be four. The cut position of a triangle on the edge can be calculated by interpolation between the two appropriate corner points and the threshold. Three neighboring cut points in a cell form one triangle. The gradient of the vertices of a triangle can be calculated by interpolation of the gradients of the cell vertices.

The problem of the marching cube algorithm is that the fitting of triangles can be done in more than one way. This is why the algorithm can cause holes in the surface. A second problem is the size of a single triangle. It happens often that triangles are smaller than a pixel on the screen. For this reason some improvements of the algorithm check the size of a cell on the screen first before the triangles are created. If the size of the cell is smaller than one pixel on the screen then the entire cell can be rendered instead of dividing it further.

All these surface rendering algorithms have the benefit that the generated model consists of primitives which can be visualized very fast and effectively using graphics hardware acceleration. A penality, on the other hand, is the creation of the surface model from the volume data set because this is a non-trivial and computationally expensive task. So for each change of the threshold a new model has to be calculated. Another penalty is that for the creation of the model an underlying structure has to be assumed. This is not always possible for some data sets.

But besides these benefits and penalties something has changed in the last 10 to 15 years what makes surface rendering less significant than volume rendering in medical volume visualization. This change was caused by the growing power of graphics cards and the higher resolution of the volume data sets coming from medical imaging devices. Faster graphics cards with more memory made it possible to handle much bigger data sets in real time. So it is now possible to load entire volume data sets in the graphics card memory. Therefore volumes can be rendered directly in real time without a time consuming generation of a surface model before. The second problem for surface rendering was the raising resolution of the volume data. This leads in a lot of cases to very small primitives if they are generated in the cell space. Therefore most of the primitives are smaller than a pixel on the screen. To combine several small primitives to a bigger primitive is time consuming and therefore not very efficient.

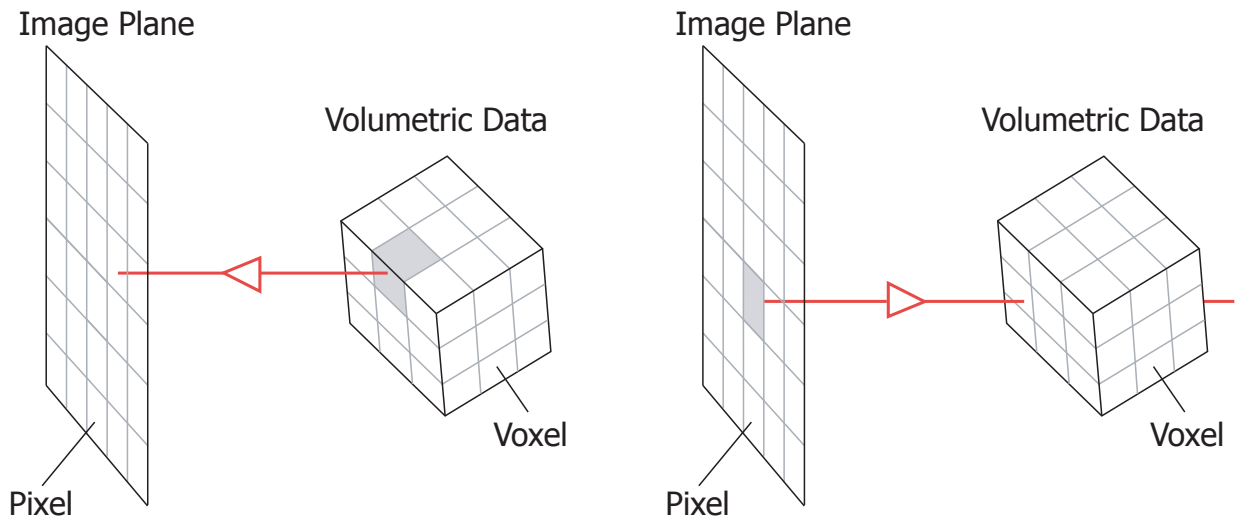
These recent developments bring us to the second group of rendering methods, the so called volume rendering techniques. Unlike in the surface rendering no intermediate representation of the volume data is needed in volume rendering. The image is directly generated from the volume data. This is the reason why this technique is often referred to as direct volume rendering.

Initially volume rendering was introduced for a better visualization of amorphous features such as clouds, fluids, and gases. Such amorphous features can also be used for medical volume data sets. For example it can be useful to visualize the skin semi-transparently to see through it to the bones. In this case the skin and other tissue in front of the bones are amorphous features.

The basic idea in generating images directly from the volume is blending together voxels projecting to the same pixel in the image plane (see Fuchs *et al.* [14]). How and in which order the voxels are combined and blended is dependent on the used algorithm. In most of the cases the algorithm is a simplification of the light transport formula. In this physical model each particle in the volume can have one of three interactions: emission, absorption, or scattering. Dependent on this interaction the particle affects the passing light. For volume rendering of medical data normally only absorption and emission are considered for a single particle. The scattering is not considered to simplify the procedure. For further reading and explanations of the way how the light transport model can be simplified for volume rendering see Engel *et al.* [11].

In general there are two basic ways how to combine voxels to a resulting image, object-order and image-order methods. Sometimes they are also referred to as forward mapping and backward mapping (see Westover [48]). Figure 2.1 shows the basic difference between these

two ways of volume rendering. The object-order approach in Figure 2.1 (a) finds for each voxel the pixel which belongs to it. In the image-order approach in Figure 2.1 (b) each pixel is traversed and all voxels are found which affect one pixel.



(a) Object-order rendering

(b) Image-order rendering

Figure 2.1: Difference between object-order and image-order volume rendering.

In contrast to surface rendering, where the data values are only used for finding a surface, in volume rendering the values are displayed directly. This is on a common display device not possible because it cannot interpret the physical values in the volume for direct displaying. So an intermediate step is needed which maps these values to optical properties which can be used by the display device. This step is called classification and is usually done with a transfer function which assigns a color and opacity to each value in the volume. A more detailed description of this process can be found in Section 3.1. For now it is enough to assume that the classification is already done.

One of the first object-order algorithms is called *splatting*. It was introduced by Westover [48]. It performs a front-to-back traversal of all voxels in the volumetric data set. The method is called splatting because it can be compared to throwing a snowball from each voxel position on the screen. Like a snowball after being thrown on a wall also the voxel is spread around the center of the projection. This distribution on the image plane is called footprint. The form and the size of the footprint is dependent on the used function. Mostly a Gaussian distribution function is used for this. For all pixels affected by the footprint a color and opacity is calculated depending on the underlying distribution function. If a color and

opacity for a pixel is already set from previous voxel footprints then the resulting color and opacity is calculated by compositing the new and the old optical properties. One benefit of this method is that a user can watch the image as it gets more refined.

Another object-order volume rendering is called *shear warp factorization*. It was first introduced by Lacroute and Levoy [31]. The algorithm consists of two steps. In the first step a 3D shear of the slices is being done along the stack of slices. The shear has the effect to change the direction of the viewing rays through the volume. After the shear all rays should pass the sheared stack of slices perpendicular to the first one in the stack. When this is done the slices are blended together from front to back with the blended result from the slices before into a result image. This image is still distorted because in general the first slice will not be parallel to the image plane. For this reason as second algorithm step a 2D warp is applied to produce an undistorted image on the image plane. To save memory both transformations can be applied directly on each slice in a front-to-back traversal and the resulting distorted slice is blended with the already existing image in the image plane.

The shear warp factorization is a very fast algorithm but the result is not that good compared to other volume rendering methods. To view the volume from all sides it is also necessary to produce three stacks of slices for each main direction. Which stack is used can be determined by calculating the dot product between the viewing direction and each of the main directions. The stack for the main direction with the highest dot product value should be used. The penalty of these view dependent stacks is a higher memory need.

The finally mentioned method is *raycasting*. It is an image-order method and the most used algorithm in medical volume visualization in the recent past. It was proposed by Levoy [32]. In an image-order algorithm all pixels of the image plane are traversed. In the raycasting algorithm a ray is shot into the scene from each pixel. This ray intersects the volume and all values along the ray are blended. The intersection position and angle of the ray through the data set is thereby dependent on the viewing position and the used camera model.

The quality of the resulting image can be controlled by the sample distance between two adjacent steps on a ray. A small sample distance produces a more accurate image than a larger distance. The quality is also influenced by the calculation of the value at a sample position. Different kinds of interpolation can be used for this. The nearest neighbor interpolation just takes the value of the voxel closest to the sample position. This is the fastest interpolation method but also the least accurate. An often used interpolation is linear interpolation. In three-dimensional space it is also called tri-linear interpolation because one interpolation consists of a combination of three linear interpolations in all three dimensions. Sometimes also higher order interpolation methods are used. An example is cubic interpolation. In

general the following rule is valid: the higher the order of the interpolation the more accurate is the result but the more voxels are used for a single interpolation.

Raycasting can produce very high quality images but the calculation time can be very high with a small sample distance. There is always a compromise between the image quality and the rendering speed. In the recent past new possibilities with new graphics cards came up. It is now possible to do the rendering entirely on the graphics card which makes it much faster as compared to a software solution. The following section gives a short introduction to various rendering techniques which are performed on the graphics card.

2.1 Hardware-based Volume Rendering

The development of hardware-based volume rendering is closely related to the development of graphics hardware. The first graphics card, or also called video card, was built in the 1960s, when printers were replaced by screens. The video card was needed to create the image which was displayed on the screen. The first graphics card with a personal PC was developed at the beginning of the 1980s. In the following years until the beginning of the 1990s the display quality was stepwise enhanced to higher resolution and more colors. In these years this enhancement was the main force for the development of the graphics cards.

The development of new graphics cards rapidly accelerated in the 1990s when computer games became popular. In 1995 the first video card was introduced which incorporated 3D functions. In the following years until now the graphics cards were constantly improved to handle more and more complex scenes from computer games or other graphical applications. Since then the average growing rate for the speed of graphics cards was higher than two each year.

With the development of more realistic and more complex computer games also a higher memory size on the video card was needed to store all the textures directly on the graphics card. For hardware-based volume rendering the size of this memory is especially important because the volume data set has to be stored on the video card. If the entire data set does not fit in the memory of the graphics card then it is not possible to execute the entire rendering process on the graphics card. A volume data set, e.g., with a size of $256 \times 256 \times 256$ where each voxel value is stored with 16 bits has an overall size of 33.5 MB. For a current graphics card a memory size of 512 MB is not unusual. So it is no problem to store this and even larger volumes directly on the video card.

Besides the memory size of the video cards the development of user-programmable *shaders* was very important for the implementation of volume rendering algorithms on the hardware.

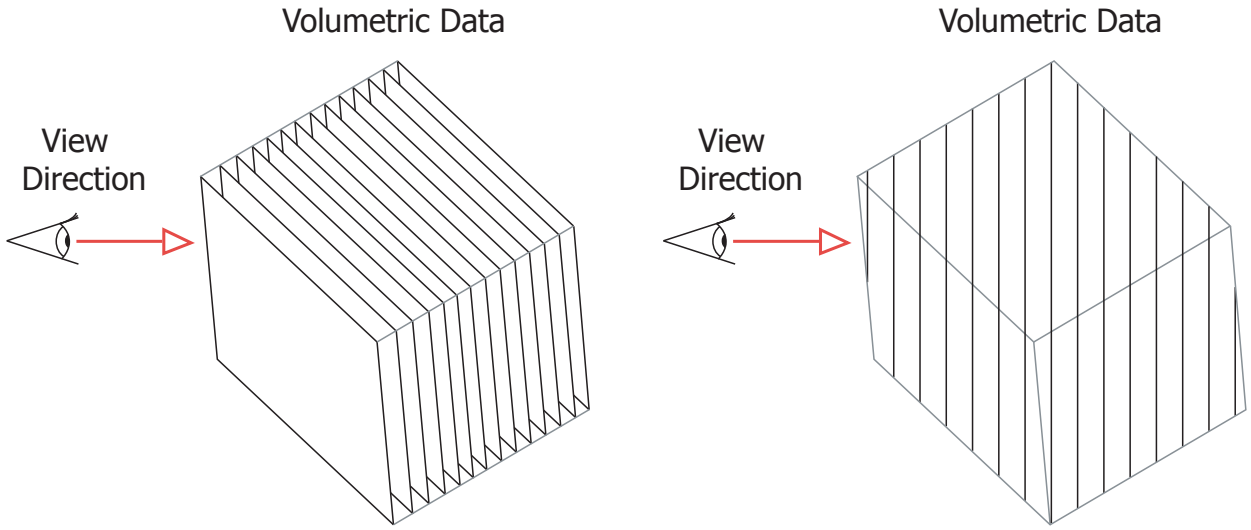
A shader is a set of software instruction to perform certain operations during the rendering pipeline. A common PC graphics card uses two shaders in the pipeline. The *vertex shader* performs its operations on each vertex of the incoming data structure. It is used for geometrical calculations. The *pixel shader* or *fragment shader* is traversed after the vertex shader. It is used to calculate the resulting color of each pixel. In an older video card design the routine of these two shader programs was fixed and could not be changed by the user. With the introduction of OpenGL 1.5 and DirectX 8 the user was able to replace both shader programs with his own programs. This allowed to perform even complex volume rendering algorithm directly on the graphics processing unit (GPU).

In the following two algorithms for hardware-based volume rendering are described in more detail. One of these algorithms is called *texture slicing* and the other one is called *hardware-based raycasting*. Nevertheless there are many more volume rendering algorithms which can be performed on the graphics hardware. As examples the hardware-based shear-warp algorithm (see Yin Wu *et al.* [49]) and the hardware-accelerated splatting (see Ren *et al.* [37]) are mentioned here.

The first texture-slicing algorithm was introduced by Cullip *et al.* [7]. It was implemented on a special graphics engine because graphics cards for personal computers were not powerful enough back then. A more recent technique which already works on common video cards was introduced by Rezk-Salama *et al.* [38]. In both approaches first the volume data set is stored as texture in the memory of the graphics hardware. By mapping this texture on a stack of planes the volume becomes visible.

The stack of planes can be arranged in two different ways. Both possibilities are shown in Figure 2.2. The object-aligned slicing in Figure 2.2 (a) aligns the stack of planes with one of the principal axis of the volume. The planes are always aligned with the main direction which points in the viewing direction. When the volume is rotated then the alignment can change from one main direction to another. This causes some artifacts. Figure 2.2 (b) shows a viewpoint-aligned slicing. Thereby the planes are always oriented in the direction of the viewer. So no artifacts are caused by a rotation of the volume but it is harder to create the geometry of the slices because they are not all quads such as in object-aligned slicing.

In most of the cases such as CT or MRI the voxel values cannot be used directly for rendering because they do not define a color and opacity. Therefore the volume has to be converted to have a color and opacity value for each voxel. For this reason a transfer function is used which classifies the voxel values in colors and opacities. This classification can be done on two different stages of the rendering pipeline. One possibility is to classify the volume before it is transferred to the graphics card. The 3D texture of the volume can then be directly



(a) Object-aligned slicing (b) Viewport-aligned slicing

Figure 2.2: Different possibilities for slicing.

used. This technique is called *pre-classification*. In contrast to this in *post-classification* the volume is transferred to the video card without assigning a color and opacity to each voxel. Additionally to the volume a texture is stored on the graphics hardware which contains the lookup table of the transfer function. The actual classification is then done in the fragment shader. For each fragment a lookup in the transfer function texture is done to acquire its color and opacity.

The penalty of pre-classification is the increased texture requirement because more memory space to save the color and opacity for each voxel instead of only the single voxel value is needed. Furthermore for each change of the transfer function the entire volume has to be classified new and transferred to the graphics hardware again. This takes some time and, therefore, it cannot be done in real time. In the post-classification technique only the lookup table has to be changed when the transfer function is changed which can be done easily in real time. The pre-classification also produces a fuzzier result because the interpolation of color and opacity values is not as accurate as the interpolation of the voxel values before the classification.

For hardware-based raycasting both classification techniques can be used. But in contrast to texture slicing the result image here is created by passing rays through the volume. The entire process is done on the GPU which is much faster as compared to a traditional software-based raycasting. The main step of the algorithm is done in the fragment program. For each

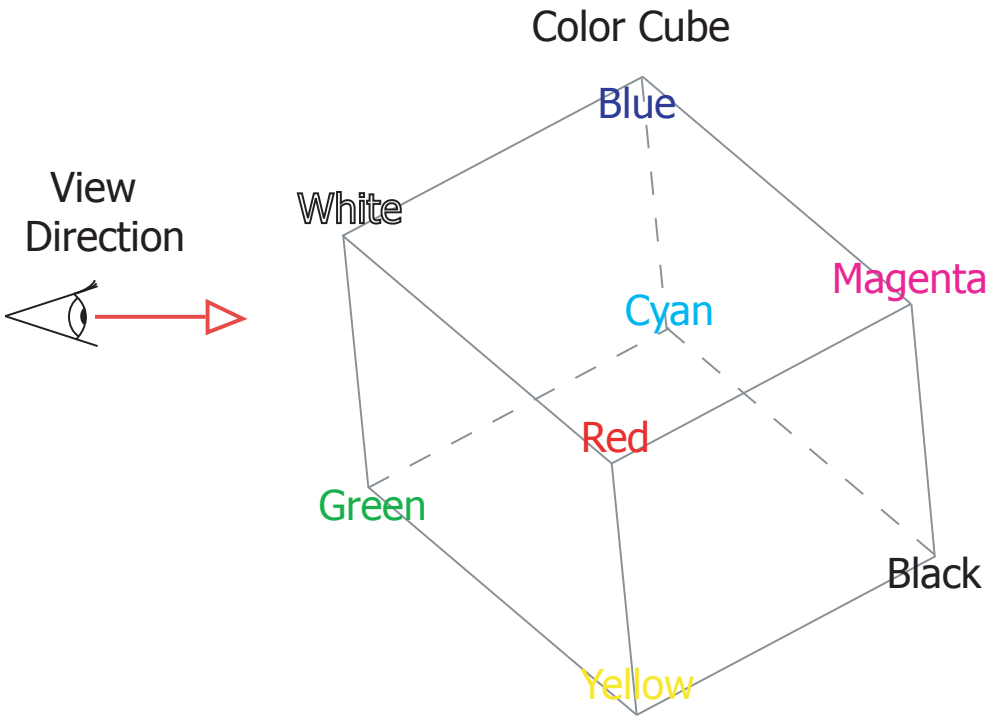


Figure 2.3: The image shows how a color cube is used for hardware-based raycasting.

fragment a ray is passed through the volume and the colors and opacities at the sample points along the ray are composited. This is done by a loop in the fragment program where in each cycle the position along the ray is moved forward a small distance.

Before the rays can be passed through the volume the entry point of the ray and the ray direction through the volume has to be acquired for every pixel. This is done by rendering a cube with the same size and position as the volume. The vertex colors of the cube are defined as shown in Figure 2.3. This assignment of colors to the vertices as for a color cube has the effect that the RGB color of every point on the surface can be directly used as position on the surface. The vertex with a red color has an RGB value of $(1, 0, 0)$. This value can be equally used as x, y, and z position on the surface of the cube and furthermore on the volume.

To get the entry point and the direction of the ray there are more rendering steps necessary. In the first step only the faces of the cube are rendered which are pointing away from the viewer. This rendering result is stored in a texture. In the second rendering step only the front faces are rendered. This rendering result is also stored in a texture. In the final rendering step only a quad is rendered which covers the entire screen. This causes that every pixel runs through the fragment shader. In the fragment shader a lookup is done in the stored textures.

If they contain a color and opacity then the ray starting from this pixel intersects the volume cube. The color in the textures is used to determine the entry point and the exit point of the ray. These two points, furthermore, are used to calculate the ray direction. With the entry point and the ray direction the loop can be started to determine the final color of the fragment.

To further increase the speed of the algorithm some additional improvements can be used. Early ray termination stops the loop in the fragment program when the composite opacity is higher than a certain threshold. With this technique some processing steps can be saved. Another improvement is called empty space skipping. Thereby parts of the volume can be skipped when they have no impact on the resulting color along a ray. The improvements can speed up the rendering by a factor of 2 to 5. For further information see Wei Li *et al.* [33] and Krüger and Westermann [30].

Besides the improvement of the speed there are also several approaches for the improvement of the quality. In pre-integrated volume rendering introduced by Engel *et al.* [12] the color along a ray is determined by integration between two sample points. This results in much smoother images. The problem is that it only works for one-dimensional transfer functions. Kniss *et al.* [25], [26] shows how multi-dimensional transfer functions can be used for hardware-based volume rendering.

The described methods are examples how the rendering can be done faster. For the application in this thesis the speed is very important because it is very hard to perform a task based on a visualization which is not in real time. Besides the speed of the method it is also important that the rendering method is flexible enough to create a merged visualization of both input sources.

2.2 Emphasized Volume Rendering

In this survey about volume visualization, so far, various methods were shown which are used to generate an image from a volume data set. In this section some approaches are shown which are designed to emphasize certain parts in the result image. The goal of this emphasis is to highlight the most important parts for a given task. This is necessary because the dense volume data sets contain too much information. In the final result the observer should be automatically attracted by the most important parts.

The distinction between the more and less important parts can be done either discretely or continuously. In both cases a value is assigned to each part which indicates it as more or less important. This value is also denoted as importance. If the distinction is done discretely

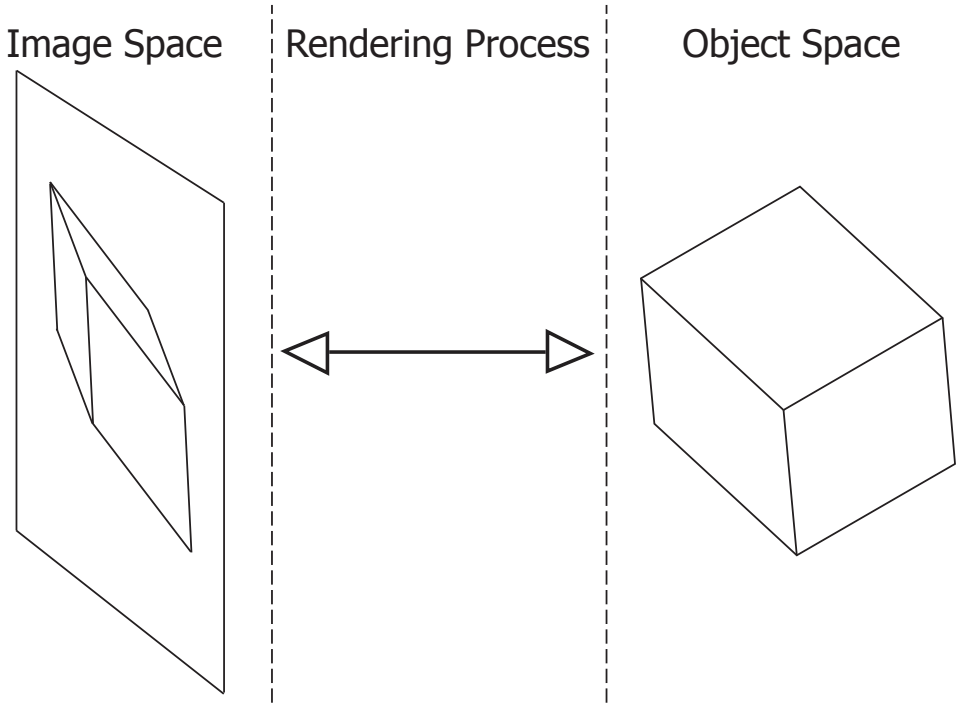


Figure 2.4: Three different stages where the emphassizing can be done.

such as binary then there is a clear difference between the less important parts and the more important parts. In such a case the more important parts are often referred to as *focus* and the less important parts are referred to as *context*. This notation comes from the fact that the less important parts are only used as context to convey the relation between them and the more important parts. If the distinction between less important parts and more important parts is continuous then the notation with focus and context is not applicable. But even with a continuous definition of the importance the same rule is valid that more important parts are highlighted in contrast to less important parts.

The emphasis of important parts can be done in different stages of the volume visualization. It can be done either in object space directly in the volumetric data, by modifying the rendering process, or by emphasizing the most important parts in image space. In Figure 2.4 the three possible stages for the emphasis are shown. The rest of this section describes the benefits and penalties of the emphasis in each stage and shows some existing approaches.

The simplest way of emphasizing is done by modifying the resulting image after the rendering process. It is the simplest way because the modification is only done in a two-dimensional space. After the result image is generated there is no more information of the inner structure of the volume left. Therefore the partition of the image into more and less important parts can only be done in the two-dimensional space. For this reason most of the approaches for

such an emphasis divide the two-dimensional space in two distinct regions. One region is the focus region which contains the most important parts for a certain task and the other region is the context region.

Depending on the approach the shape of these regions and the technique for the emphasis of the focus region is different. One of these approaches which can be applied on such a result image was introduced by Furnas [15]. Thereby a *fisheye* is used to show a region in the two-dimensional space larger than the rest of the image. The region which is displayed larger is the focus region. The rest of the image is the context region. Based on this work a lot of other techniques have been developed which are using different methods to emphasize the focus region.

The problem with all these techniques in volume visualization is that they are not able to show important parts which are, e.g., occluded by less important parts of the volume. Overlapping parts cannot be removed in the final image without a knowledge of the inner structure of the volume. So these approaches can only be used when all important parts of the volume are already visible in the result image. In most of the cases this is not true. Therefore the emphasis has to be done in an earlier stage of the visualization pipeline where the information of the inner structure of the volume is still available.

Another possibility to emphasize the most important parts is to change the volume data set itself in object space. The final data set after the modification contains only the most important parts. The less important parts are removed from the volume data set by setting the according voxels to a zero-value. For the extraction of the most important parts a segmentation algorithm is needed. The used algorithm is thereby dependent on the part which has to be segmented. Kirbas *et al.* [24], e.g., describes different methods which can be used to segment vessels in a volume data set.

The penalty of modifying the volume data set itself is that the less and more important parts can only be classified in a binary manner because a single point in the volume can only be kept or discarded. Furthermore the less important parts completely vanish after the segmentation and, therefore, they cannot be visualized anymore. For most applications it is necessary to visualize the less important parts as well to get a better impression of the position of the more important parts. In such a case the emphasis has to be done directly in the rendering process.

To emphasize the more important parts directly in the rendering process it is necessary that each voxel is classified as more or less important during the rendering process. Therefore an additional processing step has to be done to classify volumetric structures by importance. This makes the emphasis during the rendering process a bit slower as compared to the emphasis

in the volume data set where no additional processing step is necessary during the rendering. On the other hand the emphasis in the rendering process is more flexible and no information is lost before the rendering.

In general the identification of the importance during the rendering process can be done in two different ways. One way is to classify the importance of each voxel in a pre-processing step. This is mostly done by a segmentation of the volume. The second way is to classify the importance during the rendering process. Which of these two ways is used is closely related to the application for which the emphasis is needed.

A possibility to classify the importance directly in the rendering process is given when the position of the important part is known or the emphasis is used to explore the volume data set. Zhou *et al.* [50], [51] use a lens-like geometry to divide the volumetric data into two parts. The inner part of the geometry is considered as more important and, therefore, rendered completely opaque. The other part of the volume is rendered very sparsely to not occlude the important part. In contrast to that Mullick *et al.* [35] do not divide the volume rather they clip away a layer of the volume which might occlude important parts behind it. The thickness of this layer can be defined by the user.

Further techniques which do not need a segmentation are context-preserving volume rendering (Bruckner *et al.* [2]) and opacity peeling (Rezk-Salama *et al.* [39]). Both of them are generating a special kind of cutaway view. These views should cut away less important parts to clear the view on more important parts. These concepts, however, do not incorporate an importance assignment mechanism so the object emphasis and suppression mechanism is harder to specify.

All methods which emphasize important parts not based on a pre-segmentation have to be very fast because they are mainly used to explore the data set and, therefore, a real-time rendering is important for a useful user interaction. When the segmentation is done in a pre-processing step then the different important parts are already known before. With an existing segmentation the challenge of the rendering process is to provide a clear understanding of the complex data and guide the user to the most relevant information.

In the visualization of 3D flow data a degree of interest function (DOI) has been introduced by Hauser *et al.* [18] to include the user interest in the visualization. The DOI then affects previously defined optical properties to modulate the opacity or change the color hue or saturation. In the volume visualization of scalar data two-level volume rendering introduced the notion of objects in the data where each object can be represented with a different rendering technique (see Hauser *et al.* [19]).

In the above mentioned cases the viewpoint position is not considered. The assignment

of a dense visual representation to the most important focus region does not automatically guarantee a clear view at the most important part. To achieve this, optical properties of a less important region that occludes the focus have to be adapted. Several smart visibility techniques have been developed for this purpose, often inspired by traditional illustration techniques (see Viola *et al.* [43]).

One of the approaches, which is using the data relevance to automatically generate cutaway visualizations, is importance-driven visualization (Viola *et al.* [44]). Here several view-dependent operators have been introduced to suppress occluding context areas in order to increase the visual prominence of the most important parts. A single continuous importance value assigned to each segment in the volume is thereby used to control the visualization.

Occlusion of most important information is currently a very active research area. Many interactive techniques for cutaway (or ghosted) (Bruckner *et al.* [3], Krüger *et al.* [29]), peel-away (Correa *et al.* [6]) and exploded visualizations (Bruckner *et al.* [4]) have been developed. Smart visibility techniques have been applied in the visualization of the oil and gas data (Ropinski *et al.* [41]). In the domain of medical imaging, cutaway visualizations have been applied for the visualization of peripheral arteries in lower extremities (Straka *et al.* [42]). Another example from the medical imaging domain is neck dissection planning for enlarged lymph nodes removal (Krüger *et al.* [28]).

The different mentioned methods which are using a pre-segmentation for the emphasis are able to produce visualizations where the more important parts are clearly highlighted in contrast to the less important parts. The only problem thereby is the needed algorithm for the segmentation because almost all of these algorithms need a user interaction to set, e.g., some seed points. For a clinical use it is important to avoid user interactions because physicians are not interested in setting seed points or something similar. Therefore the approach in this thesis for emphasizing more important parts is designed to work without any user interaction.

So far the in this section mentioned approaches to emphasize important parts only use a single input source. For the application in this thesis two input sources are used. In the literature there are not many works which address the visualization of multimodal medical data. In one of these works magic mirrors have been applied for multimodal visualization of the human brain (König *et al.* [27]). Another way of multimodal visualization is to combine multiple modalities or properties through smooth brushing and linking of scatterplots (Doleisch *et al.* [8]). The product of this multimodal selection is mapped to a one-dimensional DOI classification and the DOI is directly mapped to a visual representation.

This chapter gives an overview on different rendering methods which can be used for volume rendering. Furthermore techniques were shown which can be used to emphasize

important parts. The following chapter describes the new approach and how important parts are emphasized there. Most of the parts of this new solution are based on our previous work (Burns *et al.* [5]).

Chapter 3

Importance-Driven Rendering in Interventional Imaging

A penny will hide the biggest star in the Universe if you hold it close enough to your eye.

Samuel Grafton

In the previous chapter several methods were mentioned which are able to visualize a single data set from a medical imaging device. For the scope of this thesis we have two different data sets from different imaging devices. The goal is to create a merged visualization which shows both data sets together at once, whereby the most important parts of both input sources should not be occluded by less important parts. For this reason new techniques have to be introduced which are able to create the desired visualization.

In this chapter all the steps to create an importance-driven rendering in interventional imaging are described. Figure 3.1 shows an overview of this pipeline. The top three nodes are the input sources and the subsequent nodes are processing steps described in the following sections.

The base inputs are the volumetric data and the object of interest, which occupies a region in the volumetric data set. For this thesis mainly a CTA data set is used as volumetric data and an ultrasound plane is the object of interest. The main goal of the approach is to have a combined visualization of the volumetric data with the ultrasound plane while the ultrasound

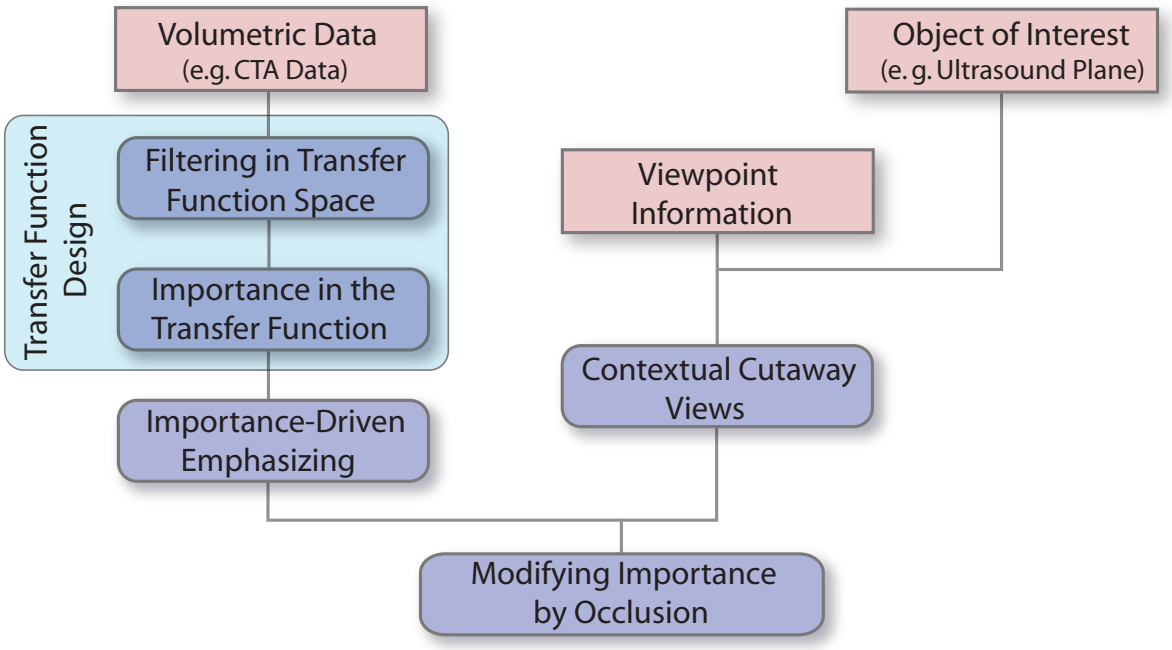


Figure 3.1: Processing pipeline to produce an importance-driven rendering in interventional imaging from the input data.

plane should not be occluded by less important parts of the volumetric data set.

The pipeline can be basically split into two parts. The nodes in the left part use the information from the volumetric data for the processing steps. The nodes on the right side operate on the region of interest data together with the information of the viewpoint. Finally the two parts are combined to a fused visualization which uses the pre-computation steps of both parts. Besides the origin of the data for both parts of the pipeline also the time base is different. The volumetric data is only recorded once before the surgery and therefore static during the intervention. Instead the region of interest or in particular the ultrasound plane is time dependent and therefore non-static during the intervention. This makes a big difference in the processing speed for each node. The first two nodes on the left side of the pipeline and the *Transfer Function Design* are operating on the volumetric data and are only applied once before the intervention. Therefore these processing steps need not to be as fast as the nodes in the right part of the pipeline.

The following sections will describe all processing steps of the pipeline. Section 3.1 will start with the transfer function design in general and the two-dimensional transfer function design in particular. It will be described how the additional information of the gradient magnitude in the two-dimensional transfer function space can be used for the design of a new

transfer function.

The results of the first section are needed for the first two nodes in the pipeline. The first processing step of the volumetric data is filtering. In Section 3.2 a new concept of filtering is introduced. The information of the transfer function is used as filter mask for different kind of filters. The goal of this processing step is a smooth filtered volumetric data set without the loss of important information.

The following processing step, described in Section 3.3, assigns an importance value to sample points in the volumetric data set based on their position in the transfer function space. Section 3.4 shows how this value can be used to emphasize more important parts in contrast to less important parts. The final result of the processing steps in the left part of the pipeline should be a smooth visualization where parts of the volume are emphasized or de-emphasized according to a single importance value for each sample point.

The right part of the pipeline addresses the region of interest. As mentioned before, the region of interest is a data set recorded during the intervention in real time. The position of this data is registered to the volumetric data set. The final goal is a fused visualization with the volumetric data where a physician can see the important information from both input sources at once. In Section 3.5 a new concept of a cutaway view is introduced. For the creation of the cutaway the position and size of the region of interest as well as the information about the viewpoint is needed to avoid occlusion of the object of interest from the observer's position.

Finally the occlusion of a given sample point, calculated by the cutaway view, is used to modify the importance value and furthermore the optical properties. In Section 3.6 the formula for this modification and its effect on the final result is shown.

3.1 Transfer Function Design

Transfer functions are a principle component of volume rendering. A transfer function describes a mapping from a data value to optical properties. This is necessary because normally the data values in a volume data set cannot be directly displayed on a screen as the value range does not fit into the color scale of a display without any modifications. However, this is not the only gain of a transfer function it also brings much more flexibility. The following formula describes the mapping which is done by a transfer function:

$$g(\vec{h}(x, y, z)) \rightarrow (\vec{p}) \tag{3.1}$$

In this expression the function g describes the transfer function. The function vector $\vec{h}$ consists of different functions applied on a three-dimensional position (x, y, z) in the volume data set. The dimension of the vector is directly related to the complexity of the transfer function. If the function vector has the dimensionality n then the transfer function is shortly named n-dimensional transfer function. The vector $\vec{p}$ describes a bundle of optical properties which are manipulated by the transfer function.

The simplest and most common transfer function is a one-dimensional transfer function. It uses the density value as single input function. In Equation 3.2 the function f returns the density for a sample point in the volume data set for a given position (x, y, z). The most common optical properties which are determined by the transfer function can also be seen in this equation. The vector $\vec{c}$ describes the color value which typically consists of the three color components red, green, and blue. Besides the color also the opacity α is specified by the transfer function.

$$g(f(x, y, z)) \rightarrow (\vec{c}, \alpha) \tag{3.2}$$

A one-dimensional transfer function is most often used in direct volume rendering. For most of the applications it is the best choice in respect to flexibility and simplicity. The density value of a sample point is used as single input value to classify it with the transfer function. For each value the transfer function returns a specific color $\vec{c}$ and opacity value α. To design a one-dimensional transfer function is much easier compared to higher-dimensional transfer functions. The most common design technique is trial-and-error. Thereby the transfer function is created by assigning color values and opacities to the density values of the sample points. If the result looks good enough then take it otherwise try to find a better transfer function. The frequency distribution of the density values of all sample points in the volume data set is normally used as support for this task.

The result of a well designed transfer function should be a classification of the sample points which distinguishes different tissues in the volume. This can be done with different colors and opacities for different tissues. The problem here is that tissues are often in the same area of the transfer function space. For one-dimensional transfer functions the density is the only input value for the classification. However different tissues might have the same density values. So it is often not possible to assign different colors and opacities to different tissues. The best possibility to reduce this overlapping of tissues in the transfer function space is to use more than one dimension for the classification. This extends the transfer function space to a higher dimension and reduces the overlapping.

As information for further dimensions of the transfer function higher order features such

as the gradient or the curvature can be used. Equation 3.3 shows a two-dimensional transfer function where the magnitude of the gradient is used as second dimension besides the density. This transfer function assigns a color $\vec{c}$ and opacity value α to each pair of density value and gradient magnitude.

$$g\left(f(x,y,z), \left\| \frac{\vartheta f(x,y,z)}{\vartheta(x,y,z)} \right\| \right) \rightarrow (\vec{c}, \alpha) \tag{3.3}$$

The gradient magnitude is also often referred to as $f'(x,y,z)$. The transfer function can be extended to higher dimensions with the additional use of other features. The use of higher dimensions can lead to less overlapping of different kind of tissues in the transfer function space but only when the features are distinctive enough. As mentioned before a lower dimension makes it easier to design a good transfer function, so the goal is to use only the most distinctive features for a given task.

For this thesis the particular problem is to find a good transfer function for a volume data set which comes from a CTA scan. A good transfer function for such a data set should be able to classify different important tissues with different optical properties. The most important parts in the particular case of the liver biopsy are the vessels. The problem is that the range of the density values for contrast-enhanced vessels is the same as for inner organs and soft parts of the bones. Therefore a one-dimensional transfer function is not enough to distinguish the vessels from other parts. The use of the gradient magnitude as second dimension for the transfer function makes it possible to find a transfer function which differentiates between vessels and other tissues.

The extension of the transfer function space into a second dimension has also penalties. The largest penalty is the fact that it is typically harder to find a good transfer function in two dimensions compared to only one dimension. The often used trial-and-error technique for one-dimensional transfer functions is much harder to use for a two-dimensional transfer function because the second dimension gives much more possibilities for error cases. In the one-dimensional case the frequency distribution of the density values gives the user an overview of the volume data set and therefore makes it easier to find a good transfer function. For a two-dimensional transfer function a scatterplot of all pairs of density value and gradient magnitude in the data set is used to support the design of the transfer function.

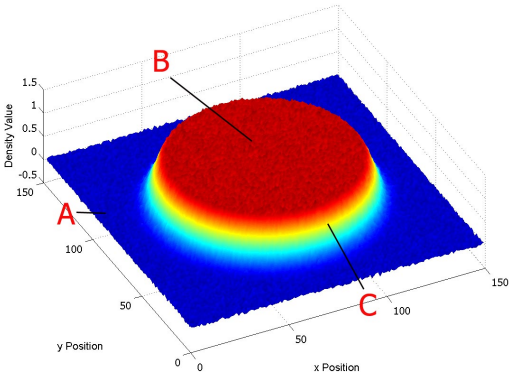
Depending on the scanning system and on the kind of scanned object specific patterns can be seen in the scatterplot (see Kniss *et al.* [25]). The properties of the scanning system have the highest influence on the appearance of the plot. MRI and CT flatten edges between different tissues because both systems are band limited. So high frequencies of sharp edges cannot be recorded by the imaging system. A second component which influences quality is

the noise which emerges during all the steps of the scanning process.

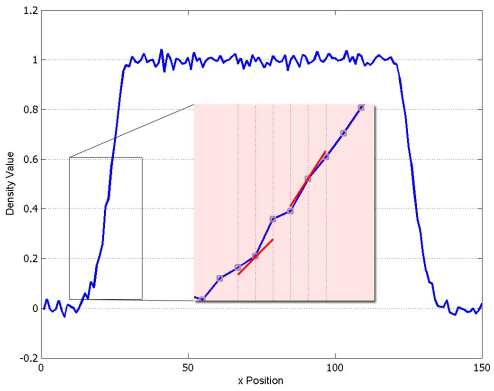
Figure 3.2 shows where these patterns come from. In Figure 3.2 (a) a two-dimensional scalar field is shown. The scalar field consists of two areas with different values. In a circle around the middle of the field the values are higher than for the rest. The edge between the higher and lower values follows a Gaussian curve such as in a real volume data set recorded by, e.g., CT. Additionally some noise is randomly added to all points in the scalar field to simulate the noise from the scanning. Figure 3.2 (b) shows the values of the points in the scalar field along a single line through the center of the field. In this diagram the Gaussian curve at the edge between the low and high values can be seen. The zoom-in area shows a small part of the edge with some gradients for single sample points. It can be seen that the gradients vary a little bit because of the noise in the data. Finally in Figure 3.2 (c) the scatterplot of this scalar field in the two-dimensional transfer function space is shown. The density values are on the x-axis and the corresponding gradient magnitudes are on the y-axis.

The red letters in the figure mark different regions in the scalar field. The letter A marks the region where the density values are low and the letter B marks the region where the values are high. The letter C marks the edge between these two regions. In the scatterplot these areas are also highlighted. The scatterplot shows a kind of arc which connects two regions. The region A on the left side of the transfer function space belongs to the values with a constant low density value. The region B on the right of the diagram in Figure 3.2 (c) belongs to the values in the scalar field with a high density value. Both of these regions have constant density values all over the entire area. Therefore the gradient magnitude for these points is very low. Only the noise added to all points in the scalar field causes small values for the gradient magnitude. Otherwise these regions would only be a single point in the transfer function space. The edge area between the two constant regions causes higher gradient magnitudes. The scattered points marked as region C are these points of the edge area. They connect the two neighboring regions with an arc in the scatterplot. The arc shape comes from the fact that the edge starts and ends smoothly near the regions with constant density values. This is also the reason why the gradient magnitude close to the constant regions is lower than in the middle between the two regions where the edge is steepest.

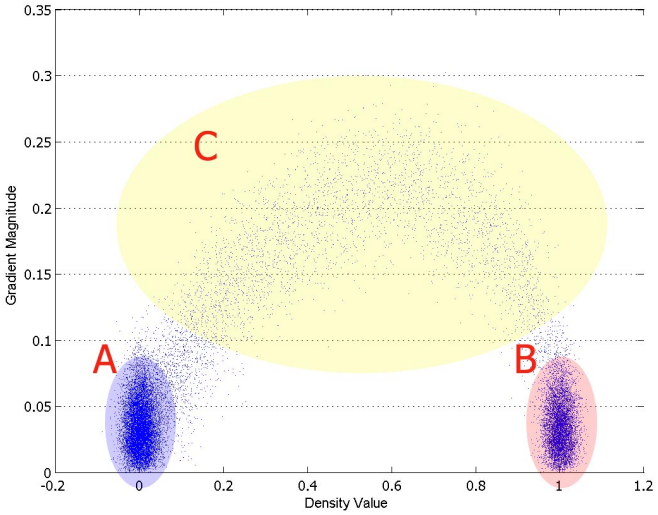
Figure 3.2 demonstrates how the points from a scalar field are distributed in the two-dimensional transfer function space. The result in the scatterplot is also the same for a three-dimensional scalar field such as a volume data set recorded by CT. Due to several different regions with constant density values several arcs are drawn in the scatterplot. Each of these arcs belongs to an edge between tissues in the body. For the proposed application in this thesis the borders of the tissues are important because the needle during the intervention



(a) Two-dimensional scalar field



(b) Density values through the center of the scalar field

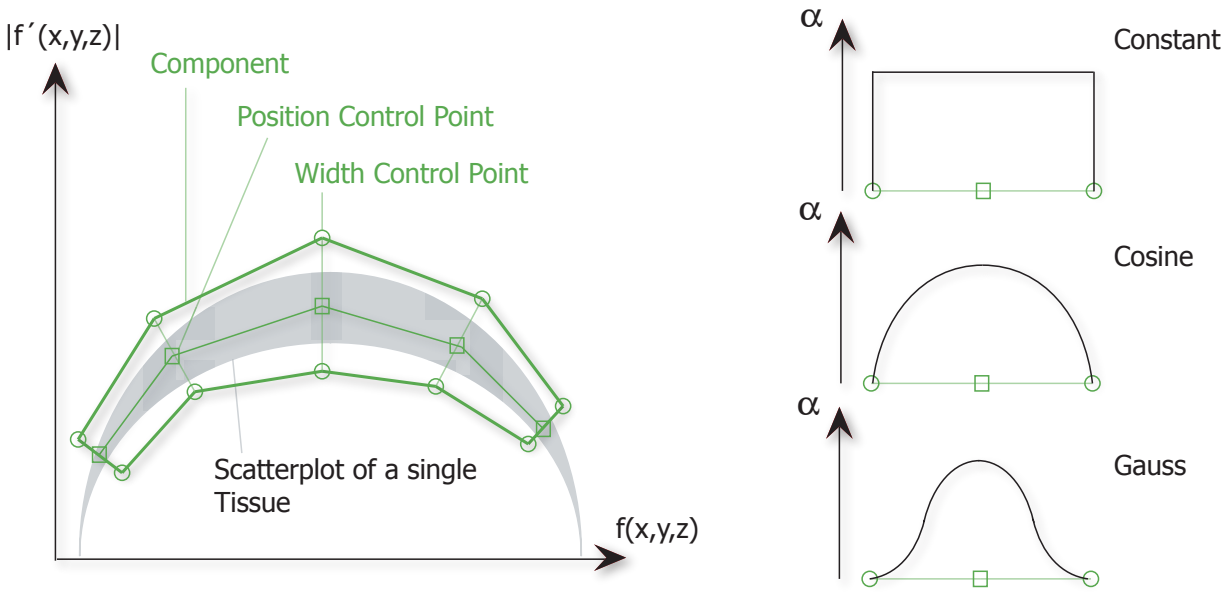


(c) Distribution of the scalar field in the transfer function space

Figure 3.2: Two-dimensional scalar field together with a scatterplot of all values of the scalar field in the two-dimensional transfer function space. The different density regions as marked in (a) can also be seen in the transfer function space (c).

should not hit any of the edges between vessels and the liver. Inner parts of tissues are not that important. This fact means that the transfer function should be able to make inner parts of tissues mostly transparent and highlight the edges between the tissues.

For the transfer function design this means that only arcs of important parts should be colored. Regions with a low gradient magnitude should have a very low opacity. For an easier transfer function design the color and opacity for each tissue of the volume data set should be assigned independently from other tissues. One region for the coloring of a certain tissue is called component. A component should be able to cover an arbitrary region in the transfer function space. It should be especially able to cover an arc.



(a) Component of the transfer function

(b) Opacity functions

Figure 3.3: In (a) an example of a single component of the transfer function is shown. The curves in (b) show different functions for the opacity distribution perpendicular to the center line of a component.

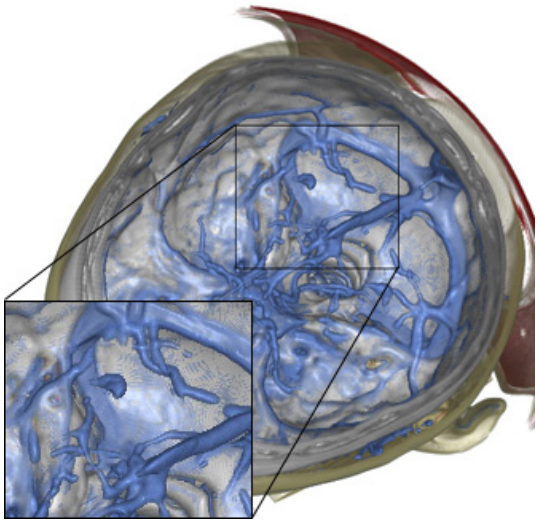
In Figure 3.3 (a) a single component is shown. The gray arc symbolizes a scatterplot of a tissue interface. The component does not cover the entire area of the arc because sample points with a low gradient magnitude are not so interesting for the final result. For the transfer function designer it should be easy to change the position and shape of a component. The two kinds of control points make it easy to change the shape and position of the component in an accurate way. The *position control points* can be moved to change the position of the center line. Each of these points can be moved independently. A component consists of at least

three of such control points. Two *width control points* always belong to one *position control point*. With these control points the width of the component perpendicular to the center line at the corresponding *position control point* can be changed. For each *position control point* the user can define a color and opacity. The color between two *position control points* on the center line is calculated by interpolation. All points along a perpendicular line to the center line are colored with the same color as the point on the center line.

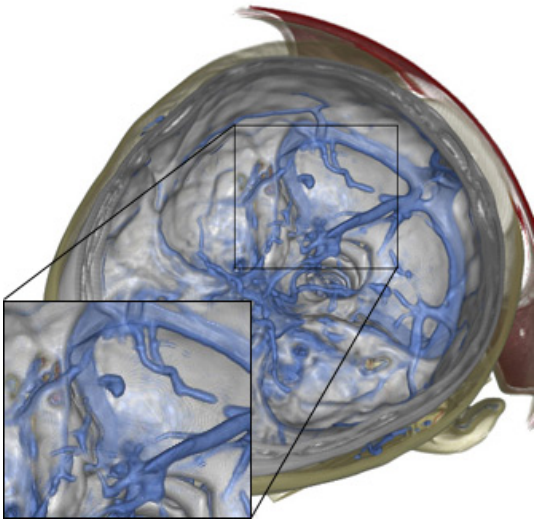
Experiments have shown that the opacity function along the line perpendicular to the center line has an important impact on the rendered result of a tissue. Figure 3.3 (b) shows three different functions of the opacity. The opacity level on the center line is calculated by an interpolation of the opacity values of the neighboring *position control points*. With a constant function the opacity level stays on the same level towards the border of the component. With a cosine and Gauss function the opacity is reduced towards the border. In Figure 3.4 a head is rendered by using the three different opacity functions for the component of the vessel structure.

In Figure 3.4 (a) a constant function was used for the component of the vessels. In comparison to the result images in Figure 3.4 (b) and (c) more artifacts can be seen. Especially on the bone of the skull more regions are colored with the color of the vessel component. This comes from the fact that a region for a tissue in the two-dimensional transfer function space may overlap with other tissues. In this special case the scatterplot of the bone structure is very close to the region of the vessel structure. In such a case the component for one tissue will also cover some parts of the other tissue. A constant opacity function produces thereby sharp edges of the opacity function on the border of the component and therefore also points in this border region are visualized with a high opacity. This causes more artifacts if two tissue edges are nearby or overlapping in the transfer function space. With a cosine or Gauss function the opacity decreases closer to the border of the component so the rendering result of the tissue itself is smoother and artifacts resulting from other tissues are reduced. Mostly there is no big difference between using the cosine and Gauss function in the resulting image. For the design it is a bit easier to use the Gauss function because the component is less sensitive to minor changes in the width of the component.

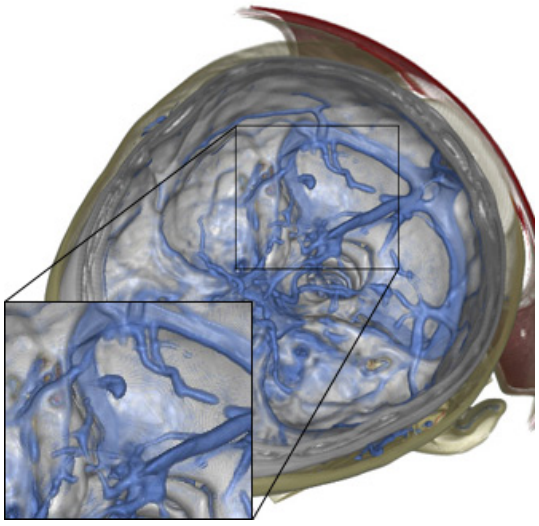
Even with all the knowledge about the distribution of points in the scatterplot it is not easy to find a good transfer function. The millions of sample points in a volume data set make it very hard to recognize the arcs from the different tissue edges. Therefore it is important to have an additional tool to explore the transfer function space easier. Such a tool should highlight the region in the scatterplot where sample points of a certain tissue are located. With this information it is much easier to design a component for this tissue.



(a) Constant function



(b) Cosine function



(c) Gauss function

Figure 3.4: The three rendering results show a comparison between the different opacity functions for the vessel component in the transfer function (blue color). The constant function in (a) produces more artifacts because the transition between the vessel component and surrounding tissues in the transfer function space is not that smooth compared to using the cosine or Gauss function.

To get points of a specific tissue the user has to select an area within this tissue in the volume data set. Figure 3.5 shows how this can be done. The clipping plane is used to cut through the considered tissue. All sample points in front of the clipping plane are clipped away to get a better view on the cut. The position of the clipping plane, thereby, can only be moved in the viewing direction for an easier navigation. If the clipping plane is at the appropriate position where it cuts the tissue then the user can move a so called pick area on this clipping plane to an area which belongs to the considered tissue. The pick area is a half-sphere through the volume behind the clipping plane. The size of the pick area can be modified by the user. If the right position and size of the pick area is found then randomly distributed points in this area are taken. For these points their positions in the two-dimensional transfer function space are calculated and are drawn in the scatterplot.

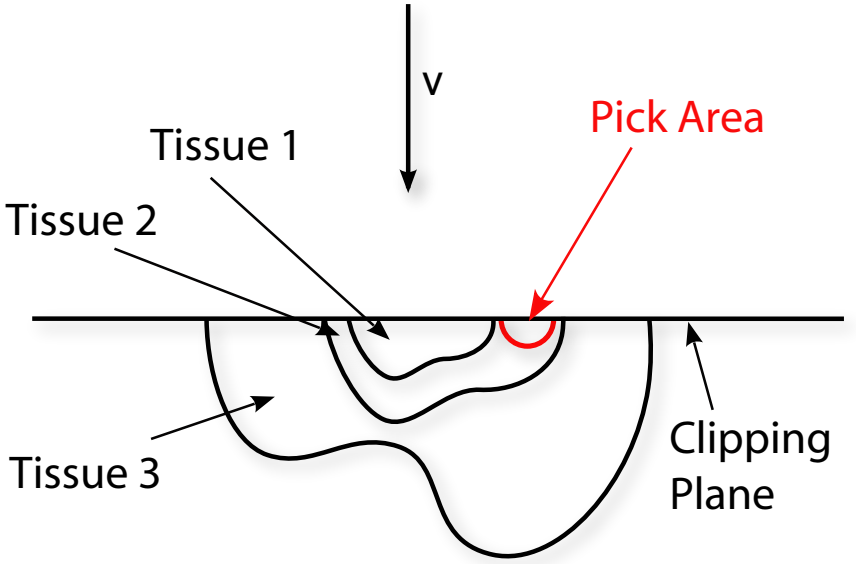


Figure 3.5: The graphics shows how the pick tool can be used to pick points from a certain tissue.

Figure 3.6 shows an example how this tool can be used. The tissue of interest in this case is the bone structure in the head. For this reason the clipping plane is placed at a position where it cuts a bone structure. The pick area is then located on a part of the spine. In the scatterplot on the right side the picked points can be seen as red crosses. The distribution of the picked points is similar to an arc. With this background information it is easier to define a component for the bones.

In the two-dimensional transfer function space it is also possible that the scatterplot of two different tissue edges is overlapping in some regions. So it is also possible that two

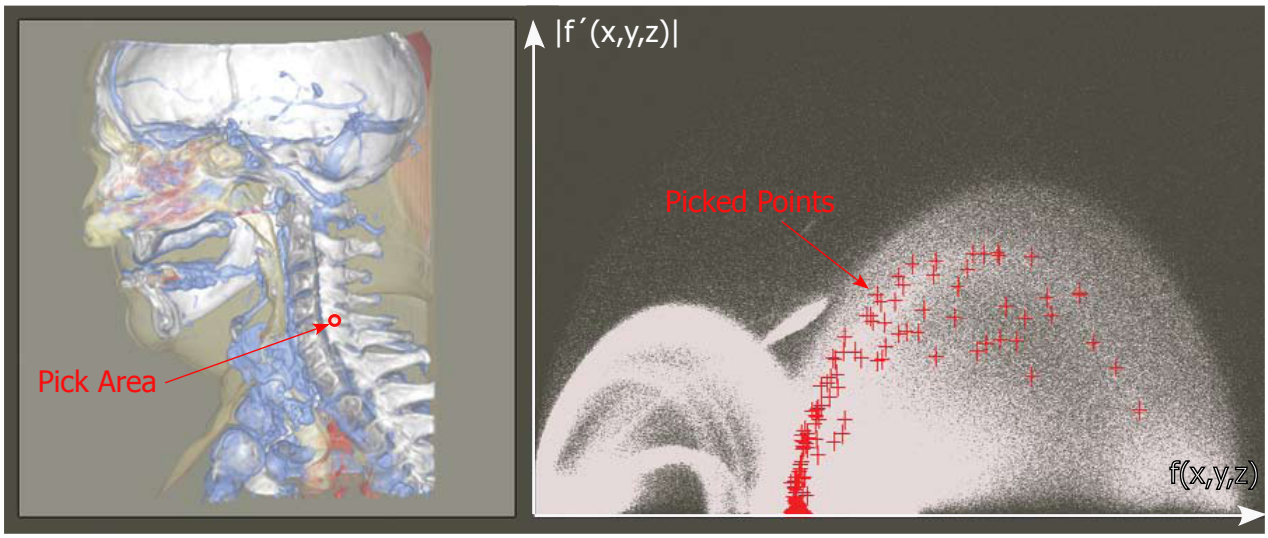


Figure 3.6: On the left side the pick area in the bone region of a head is shown. In the scatterplot on the right the distribution of sample points in the pick area can be seen.

components are overlapping but for the further rendering process a single two-dimensional color and opacity lookup table is needed. To get this the components must be blended together over the entire transfer function space. The Formula 3.4 describes this blending for each point. The variable N_{comp} stands for the absolute number of components used for the design. The resulting color $\vec{c}$ for a point in the transfer function space is calculated by a weighted and normalized sum of all colors from different components. The weighting is done by the corresponding opacity value α_i of a component. As final opacity value the maximum opacity value of a component α_i for a given point is used.

$$g\left(f(x,y,z), f'(x,y,z)\right) \rightarrow \left(\frac{\sum_{i=1}^{N_{comp}} \vec{c}_i * \alpha_i}{\sum_{i=1}^{N_{comp}} \alpha_i}, \max_{i=1..N_{comp}} \alpha_i \right) \tag{3.4}$$

The introduced concept with the individual components and the pick tool makes it easier to find a good transfer function but it needs still some experience and interactive trial-and-error adjustments. Equation 3.4 shows how the different components can be combined to use them directly for assigning colors and opacities in the rendering step. For the next two sections we still need the concept of components in the transfer function space. First the components are used for filtering and then for defining an additional importance value.

3.2 Filtering in Transfer Function Space

Each measurement process causes some errors. The source of these errors is depending on the kind of measurement. For example in an X-ray scan the absorption of X-rays by the tissues in a body is measured. A CT scan uses the same principle but instead of making only one measurement many measurements are made along a spiral around the body. All measurements are combined by cross correlation to a final volumetric data. For an MRI a magnetic field is used instead of X-rays. These are just some examples how medical images are acquired. The measurement methods might be completely different but they have all in common that they measure some physical properties. The errors in all these measurements are a combination of environmental influences combined with errors from the recording process itself such as, e.g., jitter. In Figure 3.7 a volume data set is shown produced by a CTA. In this rendering result the errors, or also called noise, can be seen especially in regions with high opacity such as the vessel structure.

The reason why regions with a higher opacity look noisier than other regions comes from the fact that a single sample point with a high opacity has more influence in the resulting image. For regions with low opacity, such as the skin in Figure 3.7, a single sample point does not influence the rendering very much. Only a higher concentration of these sample points in a region of the volume produces a dense visualization in the result. A single noisy point with a low opacity has only a low impact in the final image. Therefore a simple and often used solution for reducing the noise in a volume data set is the reduction of the opacity for all sample points. With this method the result is in most of the cases sufficiently good. The largest penalty of this technique, though, is the loss of sharp edges. So this solution for the noise problem cannot be used for the special task of a liver biopsy because the vessels in the visualization need to have sharp edges for a better three-dimensional perception and a safer navigation of the needle through the liver.

In general filtering is a commonly used tool for each kind of data. The origin of the filter technology lies in signal and image processing. The goal of the filtering is an enhancement of the incoming data for a given task. This goal is the same for volume data sets. The kind of enhancement is related to the needs of the filtered data set. In general there are two different tasks for which filters can be used:

- Noise reduction
- Segmentation / Feature extraction

A filter for the noise reduction should find and correct corrupt points. Filters for segmentation

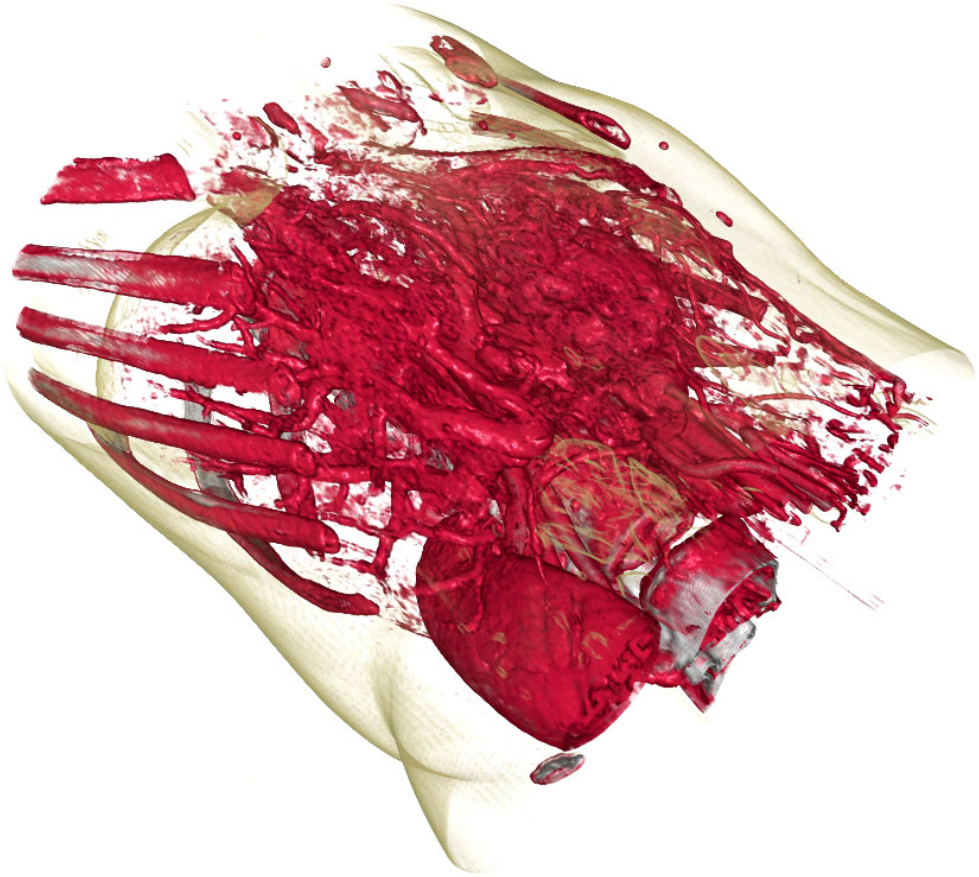


Figure 3.7: Image of an unfiltered volume data set of the abdomen. (This and the following images in this section are scaled by an factor of 1.7 in z-direction for an easier recognition of the differences.)

and feature extraction should find particular regions in the volume data set and extract them from the rest of the data set.

For the given task of this thesis only two kinds of filters are needed to get a sufficiently good result. One filter is needed to reduce the noise especially for the vessel structure. A second filter is used to remove parts of the bones which occlude the vessels. In Figure 3.7 these bones can be seen. Especially the ribs are a problem because the standard viewpoint of a physician is the same as shown in Figure 3.7. To remove these parts they have to be segmented. First we will concentrate on the reduction of the noise in the volume data set.

Noise in data has normally a high spatial frequency because the noisy points are small spikes with sharp edges to the surrounding points. Therefore these points produce high frequencies in the frequency space. A noise filter should remove these high frequencies in

the frequency space to smooth the edges between the spikes, coming from the noise, and the surrounding points. The filtering can be done either in the image space, directly on the data points, or in the frequency space. In volume rendering the filtering is mostly done in the object space using convolution with the filter operator matrix. The filtering result is depending on the type of the used filter kernel. For the discrete case, such as for volume data sets, an average filter such as mean, median, or Gaussian filter can be used to eliminate the high frequencies. All of these three filters smooth the result by adapting a considered point to the values of its neighboring points.

The only problem with using these filters is the fact that the vessel structure has very fine branches in some parts. These fine branches also cause high spatial frequencies. Therefore the proposed filters would also eliminate some of these branches. Due to the fact that there is no difference in the spatial frequency of the noise and fine branches, it is impossible to find another filter kernel, besides the normal smoothing filters, which eliminates the noise without eliminating the fine branches. So the only solution for this problem is a segmentation of the critical parts of the vessel structure and applying the filter only on the rest of the volume data.

Typically the segmentation is done in three-dimensional space. The algorithms for this kind of segmentation have some drawbacks which are adverse for the proposed application in this thesis. One of these drawbacks is the need of a user interaction for most of these algorithms. It also takes some additional time for completing the segmentation task, and finally these algorithms are only able to segment the entire vessel structure. For the proposed application in this thesis it is enough when only the fine branches are segmented, because the rest of the vessel structure can be smoothed together with the rest of the data set.

To avoid the mentioned disadvantages of a usual segmentation, the new approach does not use the spatial position of a sample point as base for the segmentation rather it uses the position of the sample point in the two-dimensional transfer function space to decide if it belongs to the segmented region or not. Expression 3.5 formalizes this concept. Function h is the segmentation function. For each pair of density value and gradient magnitude it returns a single binary value b. If it is 0 the sample point does not belong to the segmented area, if it is 1 then it belongs to the segmentation region.

$$h\left(f(x,y,z), f'(x,y,z)\right) \rightarrow b \tag{3.5}$$

For an easier design the concept of components introduced in the previous section can be used. The design of such a segmentation component works the same way as the design of a component for the transfer function. The pick tool introduced in the previous section can be

used to pick some sample points from a region which should be segmented. The position of these points in the transfer function space can then be used as orientation for the design of the segmentation component. It should be possible to design such a segmentation component only once and use it then for all data sets which are recorded for a given medical application. So the entire filtering step needs no user interaction by a physician.

Normally the segmentation region can be defined with a single component because the shape of the component can be changed very flexibly. If there are more components needed, e.g., when two not connected regions in the transfer function space belong to the same segmentation then these components have to be blended together. This blending can be seen in Expression 3.6. For each point in the two-dimensional transfer function space the binary values b_i of all components are combined by a logical OR function. The final blended function will return 1 for all points where at least one segmentation component is defined.

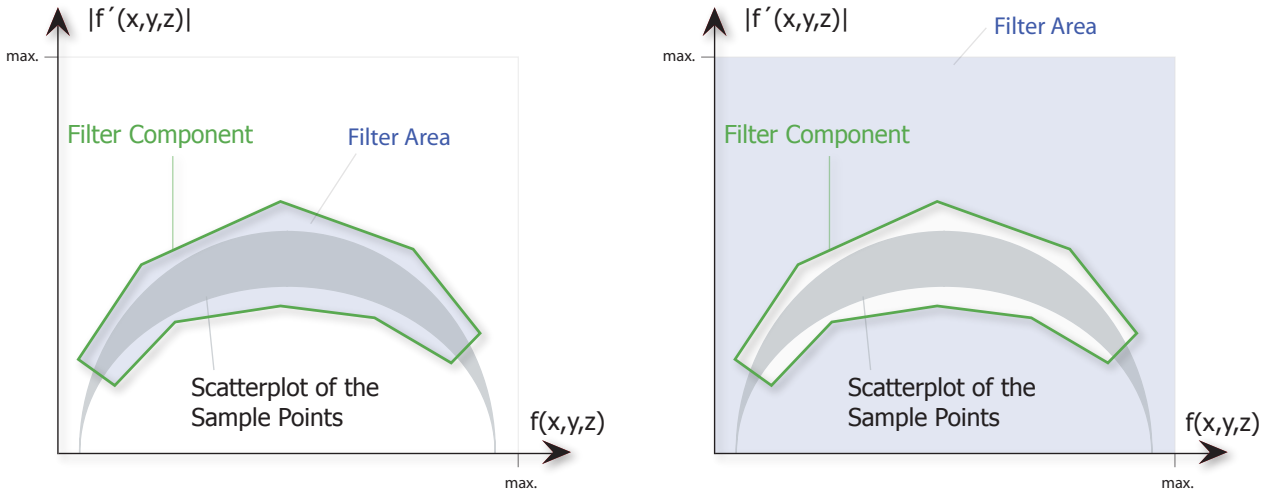
$$h\left(f(x,y,z), f'(x,y,z)\right) \rightarrow \left(\bigvee_{i=1..N_{comp}} b_i \right) \tag{3.6}$$

For the filtering this segmentation can be used in two different ways. Figure 3.8 shows these two different possibilities. In Figure 3.8 (a) only points in the segmentation component are used for the filtering step. This can be used when the filter should only be applied on points inside a certain segmentation component. In Figure 3.8 (b) the filter is applied on all points except the points in the segmentation region.

With a given filtering function h the filter step is quite easy. For each sample point the position in the two-dimensional transfer function space is acquired. Depending on the position and on the segmentation option, as shown in Figure 3.8, the filter kernel is applied on the considered sample or not. For the mentioned application of noise reduction of the vessel structure, a segmentation component is designed for sample points which belong to the fine branches of the vessel structure.

Figure 3.9 shows the difference between filtering of all sample points as compared to filtering of sample points which are outside of a well defined segmentation component. As filter mask for both results a 3×3×3 sized mean filter was used. The zoom-in area highlights a part where small branches are almost completely removed in Figure 3.9 (a). In contrast to this, in Figure 3.9 (b) the fine branches are still there. If we compare the result in Figure 3.9 (b) with Figure 3.7, a difference can be seen concerning the smoothness of the result but the fine branches are still there after the filtering. Figure 3.9 (a), in comparison, looks much smoother but almost all of the fine branches are gone.

The segmentation of the fine branches is easier to do in the transfer function space because



(a) (b)

Figure 3.8: Possibilities for using a segmented region. In (a) only the points inside the segmentation component are used for filtering. In (b) all points except the points inside the component are used.

the gradient magnitude of these fine structures is lower than the gradient magnitude of thicker branches and other parts in the vessel structure. It would be much harder to segment only the fine branches by their location in the volume instead of the usage of the transfer function space. But there are a lot of other tasks where a segmentation in the transfer function space is not possible. The ribs, e.g., cannot be distinguished from parts of the vessel structure by their position in the two-dimensional transfer function space because they have the same density and gradient magnitude as the vessels.

As mentioned before these parts of the ribs which are in the same component with the vessel structure may occlude some vessels. The goal is to remove at least a part of these bones to get a better view on the vessels in the liver. Due to the fact that the segmentation of the ribs by their location in the transfer function space does not work, we have to find another solution for this problem. Typical segmentation algorithms cannot be used because they usually need some user interaction and this is something we want to avoid.

In the segmentation task for the filtering only the information of the sample point itself is used. This information might not be enough for all segmentation tasks. To get more information about a sample point in the volume often the neighboring points are considered as well. Together with the neighboring points it is possible, e.g., to calculate the gradient or the curvature. The neighboring points can also be used to detect which kinds of tissues are surrounding a particular sample point. The transfer function space, again, can be used to

(a) (b)

Figure 3.9: Image (a) shows a filtering result by applying a mean filter to all sample points. In (b) the filter was only applied to sample points outside a segmentation component.

find out which tissues are around a considered point.

The diagram in Figure 3.10 shows the distribution of a sample point and its neighboring points in the transfer function space. In this example the neighboring points of the considered sample point are part of and between two tissues. The two components in this graph are two possible components for these tissues. Some of the neighboring points belong to the area of the lower component whereas the considered sample point is part of the upper component. If there are more of the neighboring points in another component then it means that the considered point is close to this tissue.

This neighboring information can now be used in the following way. If we know there is a tissue close to parts which should be segmented then such a neighboring test can be used to find out if the sample point is close to this tissue. For this reason two different segmentation functions are needed. We will call them h_1 and h_2. Each of these functions look exactly the same as the function in Expression 3.5. The function h_1 is applied on the considered sample point. If this function returns true then the function h_2 is applied on all neighboring points in a certain surrounding of the considered sample point. The more often h_2 returns true for the neighboring points the higher is the probability that the sample point is close to the tissue expressed with h_2.

In Figure 3.10 two segmentation functions are shown. Each of these functions consists of a single component. The first segmentation component belongs to the function h_1 the

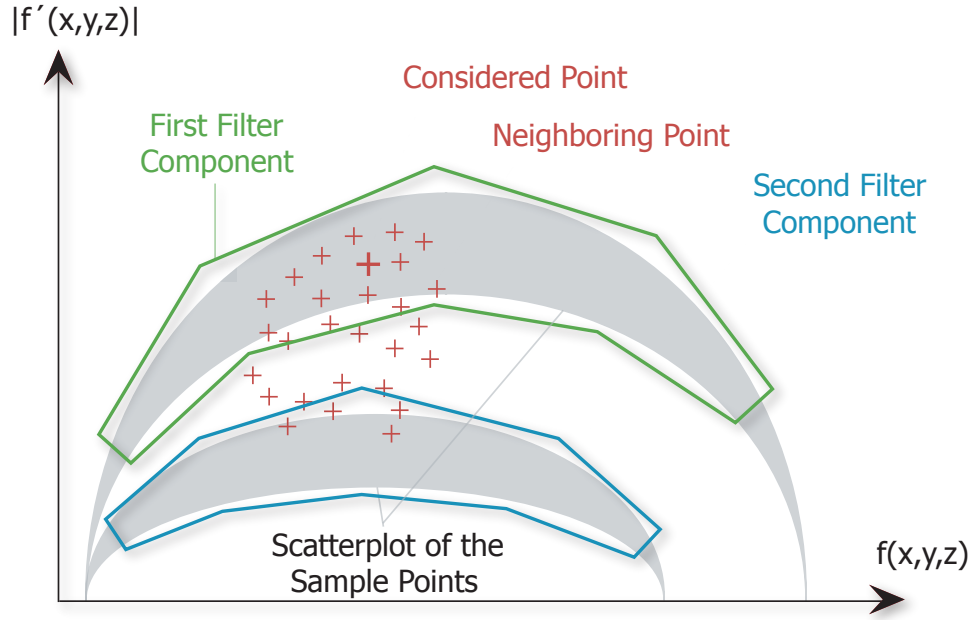


Figure 3.10: Neighboring points of a single sample point can be used in the two-dimensional transfer function space for segmentation purposes.

second segmentation component belongs to h_2. In this example the sample point lies in the first segmentation component. So h_1 would return true for this point. If h_2 is applied to all neighbors it would return a total count of four *true* responses. With defining a threshold for this total count it is possible to segment sample points. The threshold is dependent on the total number of neighboring points used for the segmentation.

For the given task, on which this thesis is based, we are interested in removing parts of the ribs which might hide some vessels in the liver. These parts of the ribs are in the same area of the two-dimensional transfer function as the sample points of the vessel structure. So there is no possibility to distinguish them in this space. But the sample points in the ribs are surrounded by other bone structure. This fact can be used to segment these sample points by their neighborhood to this bone structure. For this reason the function h_1 consists of the same component used for the transfer function of the vessel structure. The function h_2 returns true for all points which belong to the bone structure.

In Figure 3.11 the result of this segmentation can be seen. For the segmentation a $3\times3\times3$ window size was used. If a sample point returned true on the function h_1 then the function h_2 was applied on all 26 neighboring points in this window. If the total number of *true* responses was higher than a threshold value of five then the considered sample point was removed. In

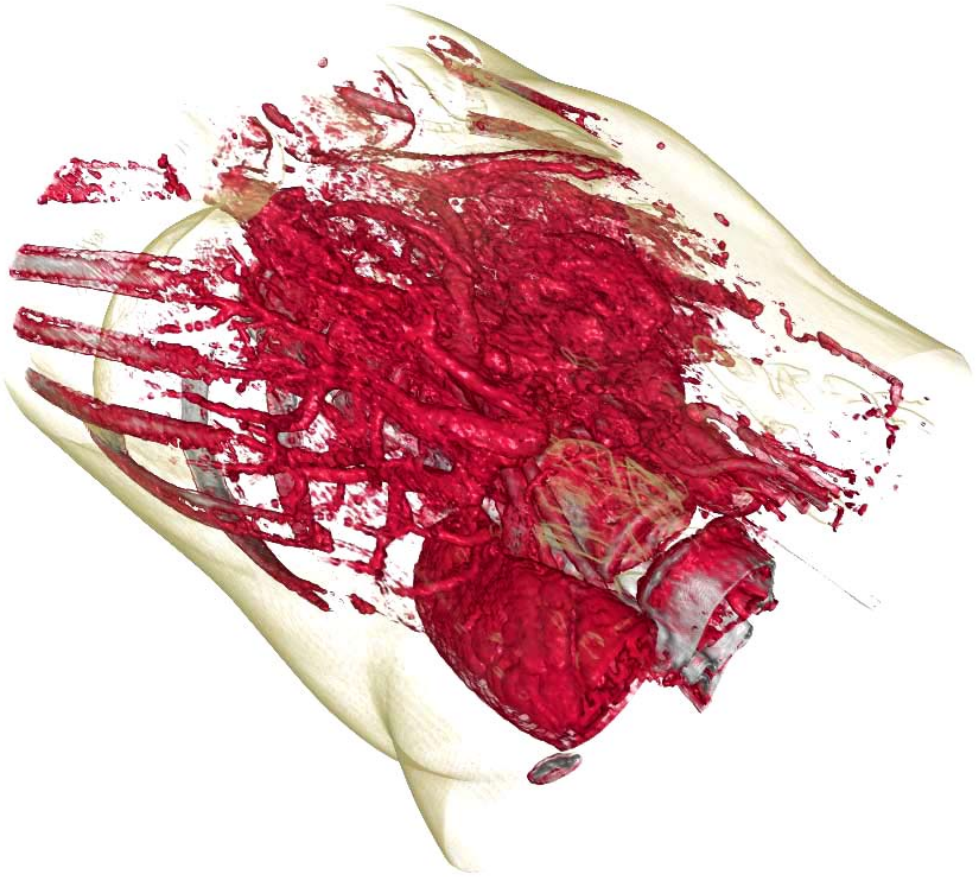
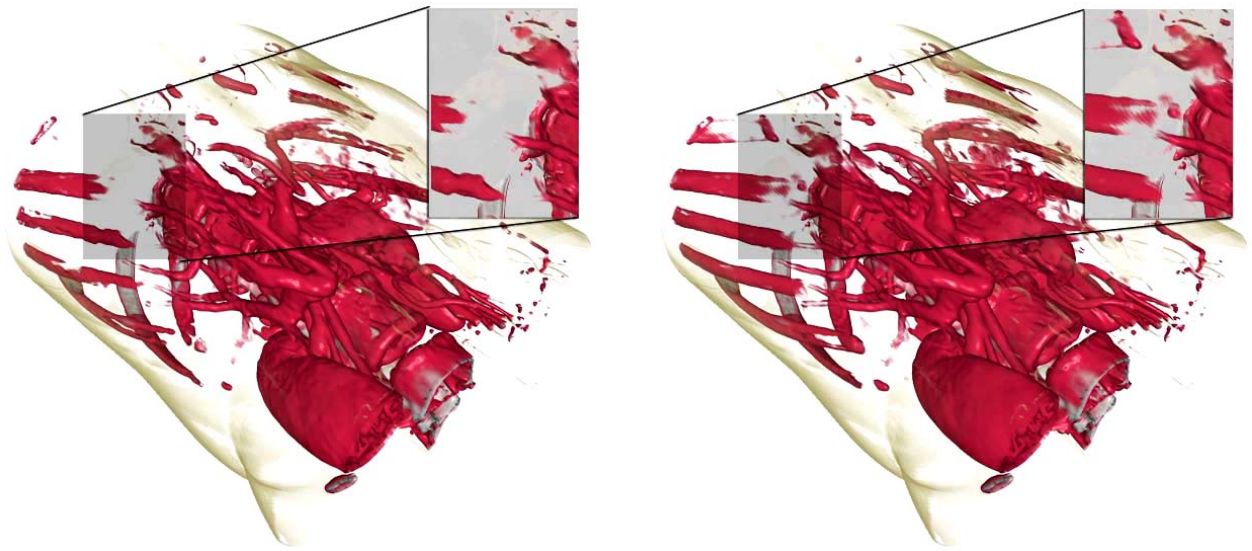


Figure 3.11: Result after applying a neighbor-filter to the vessel component.

the result image it can be seen that a lot of sample points have been removed especially from the ribs area.

The result after this removal looks very noisy. So an additional smoothing step has to be done afterwards. The smoothing is done as described earlier in this section. In Figure 3.12 (a) the final result after the bone removal and the smoothing can be seen compared to the smoothed data set without the bone removal in Figure 3.12 (b). It can be seen that some parts from the end of the ribs are removed by the introduced algorithm. These are the most important parts to remove for the given medical task because some vessels in the upper part of the liver could be hidden by them.

With applying the introduced algorithm on the volume data set we get a smooth result data set without any overlapping parts of the vessel structure. The entire filtering is done as a pre-processing step. So it is not time critical and does not have to be in real time. For CTA scans the components which are needed for the algorithm only have to be designed once for a



(a) (b)

Figure 3.12: Image (a) shows the final effect of the bone removal after smoothing the result compared to the result without applying the bone removal in image (b).

given medical task and for a given imaging system. They can be used almost unchanged for all data sets from the same imaging system because the density value of the contrast agent remains almost on the same level even for different patients. Therefore the user interaction of a physician can be reduced to a minimum.

After applying the filter steps on the data set the volume is ready for visualization. Before we concentrate on the volume visualization itself we have to find a technique which can be used to highlight important parts. In the following section an importance value is defined which can be used for this task.

3.3 Importance in the Transfer Function

Due to the amount of information in a single volume data set a representation is needed which directs the observer to the most significant parts of the data set. For this reason a kind of segmentation is needed which splits the volume into more and less important parts. The used visualization technique should then be able to highlight the more important parts in addition with an overview visualization including less important parts. The overall goal is to make the cognition of important parts fast and easy while keeping an impression of the position and the relation of these parts with respect to the rest of the data set. In the final visualization a physician should be able to recognize the most important parts and to see their positions in

the three-dimensional space.

The idea of highlighting more important parts in contrast to less important parts is not new in volume visualization. Because of the dense volume data set such a technique is needed for most of the visualizations. For the simplest and most often used technique only the transfer function is used to suppress the less important parts. This can be done by assigning a low or zero opacity to these parts. This approach needs no further processing steps because every modification is only done in the transfer function. The penalty of this approach is the lack of flexibility because only the opacity can be changed. Furthermore it is often very hard or impossible to highlight special parts only by the transfer function because they might be in the value range of other parts of the volume data set.

To have more flexibility a segmented data set can be used. The segmentation splits the entire volume data set into smaller parts. For medical applications each of these segments typically belongs to a tissue in the body. Depending on the importance of a tissue a separate visualization method for each segment can be used. This brings more flexibility than only modifying the opacity value of each tissue. The penalty of this method is the segmentation task. The segmentation of a volume data set in spatial areas is a time consuming and computationally intensive task which mostly also needs some user interaction. The last issue is very important because for a practical use of the method it is not desired by a physician to, e.g., spend some time in selecting seed points for a segmentation algorithm. For an overview of algorithms for vessel extraction see Kirbas *et al.* [24].

In this thesis a new approach for highlighting more important parts is introduced which avoids the penalties of the commonly used techniques. In difference to a segmentation by the position in the three-dimensional space the segmentation for the introduced technique is done in the transfer function space. This has the benefit that the segmentation can be pre-defined once for a given kind of medical task on a certain imaging device and no user interaction is needed. It works especially well for data sets which are acquired with the additional use of a contrast agent because tissues which are highlighted by the contrast agent in the volume data set are easier to segment in the transfer function space.

For the segemtation in the transfer function space the idea of components is used as introduced in Section 3.1. Each component in the transfer function space belongs to a tissue. For a usual transfer function the final combination of all components leads to the Expression 3.2 which assigns a color and opacity to all points in the transfer function space. As mentioned before this technique has a lack of flexibility for the highlighting of more important parts because only the color and opacity can be used for modifying the visual result. To have more flexibility an additional value is assigned to each point in the transfer function space. This

value is called importance value. In Expression 3.7 the transfer function with this additional importance value I can be seen.

$$g(\vec{h}(x,y,z)) \rightarrow (\vec{c},\alpha,I) \tag{3.7}$$

The function vector $\vec{h}$ consists of different functions applied on a three-dimensional position (x,y,z) in the volume data set. The vector $\vec{c}$ is the color and α is the opacity. These two optical properties are independent of the importance and can be defined independent from it. The value range of the importance value I is from 0 to 1. A value of 0 stands for a very low importance and a value of 1 indicates a very high importance.

As mentioned before the idea of components in the transfer function space is well suited for the importance assignment. In Section 3.1 each component covers a region in the transfer function space. The color and opacity are distributed within a single component in a well defined way depending on the assigned values at the *position control points*. In contrast to this the importance value is defined only once for the entire component area because a tissue which is covered by the component in the transfer function space should only have a single importance value.

Normally a transfer function consists of more than one component. So it is possible that two or more of them are overlapping in some regions of the transfer function space. In such a case the components must be blended. In Expression 3.4 in Section 3.1 this blending is done with the color and the opacity values. Now with the additional importance value the blending of the components is done as given in the following expression:

$$g(\vec{h}(x,y,z)) \rightarrow \left(\frac{\sum_{i=1}^{N_{comp}} \vec{c}_i * \alpha_i * I_i}{\sum_{i=1}^{N_{comp}} \alpha_i * I_i}, \max_{i=1..N_{comp}} \alpha_i, \max_{i=1..N_{comp}} I_i \right) \tag{3.8}$$

I_i is the importance value of component i. N_{comp} is the number of all components overlapping at a single point. The RGB color components are combined in this formula in the vector $\vec{c}$. In contrast to Equation 3.4 the color is here additionally weighted by the importance value. This is necessary because the color of more important parts should have a higher impact in the resulting color than a color of a less important part. For the final opacity α as well as for the importance value I the maximum value of all components overlapping on a given point is taken. The maximum opacity and importance value at a given position is not influenced by lower values from other components.

The blending is only done once before the rendering and results in a lookup table which assigns each point in the transfer function space to a quintuple consisting of RGB color values, an opacity value, and an importance value. After this blending we have everything ready for

the rendering. We have a filtered volume data set and we have a lookup table which returns for each sample point of the data set a color, an opacity and an importance. All algorithms and steps introduced so far can be done in a pre-operative step. Starting with the next section all steps have to be done in real time.

The following section explains how the additional importance value can be used for the visualization to highlight more important parts in contrast to less important parts.

3.4 Importance-Driven Emphasizing

For a standard rendering method a traditional transfer function is used which assigns a color and an opacity to each sample point in the volume. The colors are weighted by their opacity and blended together through the entire volume. In Figure 3.13 a head is shown which is rendered with such a standard method. It might be a good visualization for some applications because several tissues can be seen at once. Therefore this visualization gives a quite good overview of the head. For some applications, however, one or more tissues in a volume data set are more important than others. If this is the case, it makes more sense when these more important parts are highlighted in contrast to the less important parts. So the attention of an observer is automatically focused on the more important parts. Moreover more important parts should also not be occluded by less important parts. In Figure 3.13 the bone structure, e.g., is occluding the structures in the brain.

The goal of the importance-driven emphasizing is to create a result image which brings out the regions of interest without loosing the overview on the rest of the volume. All algorithms for emphasizing will be applied to the head data set to see the difference to the standard rendering method in Figure 3.13. The vessel structure will be considered as most important part of the head data set.

As starting point for the importance-driven emphasizing we only have the additional importance value for each sample point. This is the only difference to a standard rendering method. As mentioned earlier the vessel structure is considered as most important part, so the importance value is high for sample points which belong to this structure. For the rest of the sample points the importance value is lower. The resulting color of a sample is commonly a combination of the color coming from the transfer function manipulated by several optical properties. Such optical properties are, e.g., the opacity of a sample point or the shading method. Besides the opacity which results together with the color from the transfer function these optical properties are typically defined globally and, therefore, are valid for all sample points.

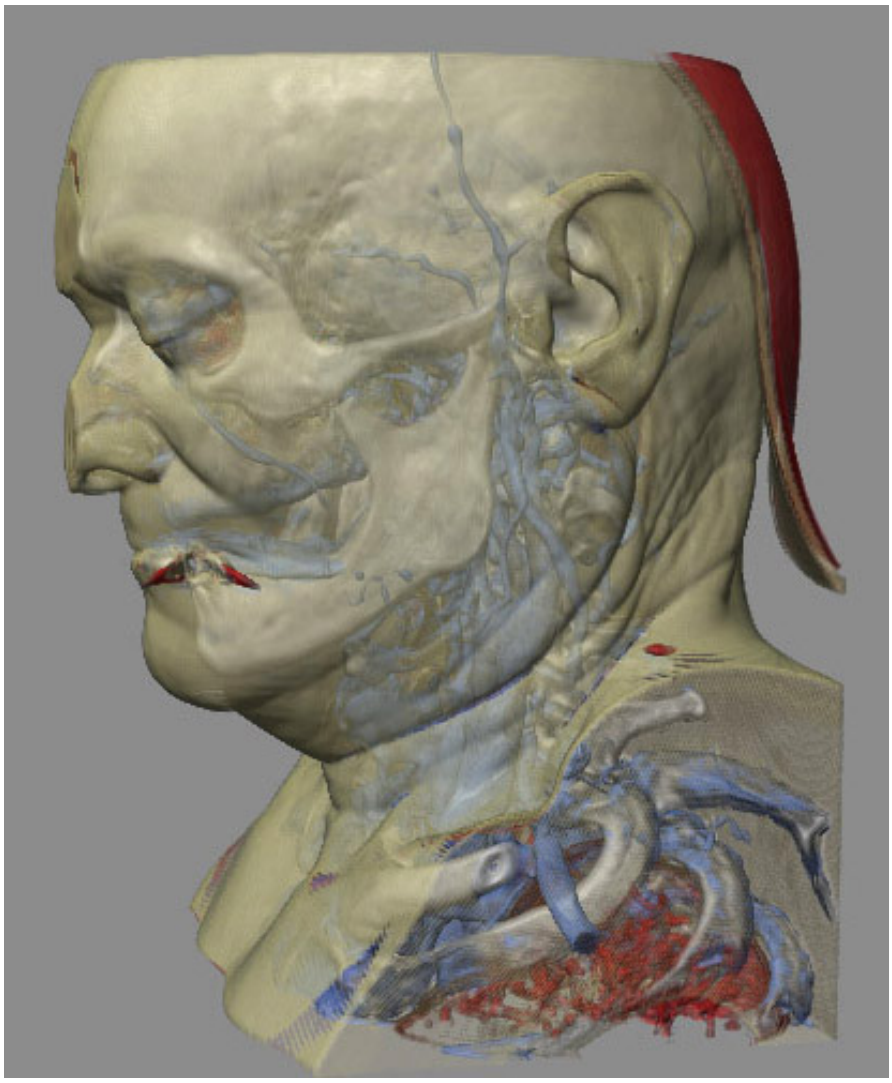


Figure 3.13: A head visualized with standard volume rendering and without using the additional importance value.

The idea behind the importance-driven emphasizing is to manipulate some of these optical properties by the importance value of each sample point. This makes it possible that even globally defined optical properties can be manipulated locally by the importance value. To reach the goal of emphasizing more important parts in contrast to less important parts the following properties are chosen to be manipulated by the importance value:

- Shading
- Opacity
- Silhouette Enhancement

The overall goal of these modifications is to guide the viewer quickly to the most important parts of an image. For this goal two requirements have to be fulfilled: The important parts should not be occluded by less important parts and important parts should have more details and a higher contrast than the rest. The first requirement is controlled by the opacity modification. The second requirement is controlled by the shading modification and by the silhouette enhancement. Below the modifications of all three properties are explained in more detail.

The first property, the shading, should be more realistic for important parts. Less important parts should be shaded with less details because the human cognition is more attracted by parts in an image which look more natural. For this reason no shading is used for unimportant parts and a more complex shading method is used for important parts. Equation 3.9 describes the modification of the shading by the importance value.

$$\vec{c} = \vec{c}_{shaded} * I + \vec{c}_{unshaded} * (1 - I) \tag{3.9}$$

$\vec{c}_{shaded}$ is the resulting color from the more complex shading method and $\vec{c}_{unshaded}$ is the color resulting from no shading. For unshaded parts the color from the transfer function is directly used without any calculations. For the shaded color the original color from the transfer function is just used as parameter together with a light source and some material properties to calculate the final color. The formula for this color calculation is depending on the used shading method. The importance value I is used as interpolation value between the shaded color and the unshaded color.

Figure 3.14 shows the result of the shading modification by the importance value. For calculating the shaded color the Phong shading model was used. The vessel structures with a high importance value look still the same such as in the result with the standard rendering method in Figure 3.13. The less important parts look much flatter after the modification. This results because the Phong shading model modifies the color in relation to the position of a virtual light source. Without this shades from the light source the less important parts lose their depth.

As one can see this shading modification reduces the depth of less important parts but these parts are still occluding the more important parts. Therefore the modification of the opacity can be used to reduce the opacity of these less important parts. By reducing the opacity of all sample points with a low importance value, these parts will vanish more and more. We still want to have an overview where the less important parts are located. A good solution to get this is the reduction of the opacity only for regions which are not important for getting an impression of the shape and position. In most cases the silhouette of these parts will

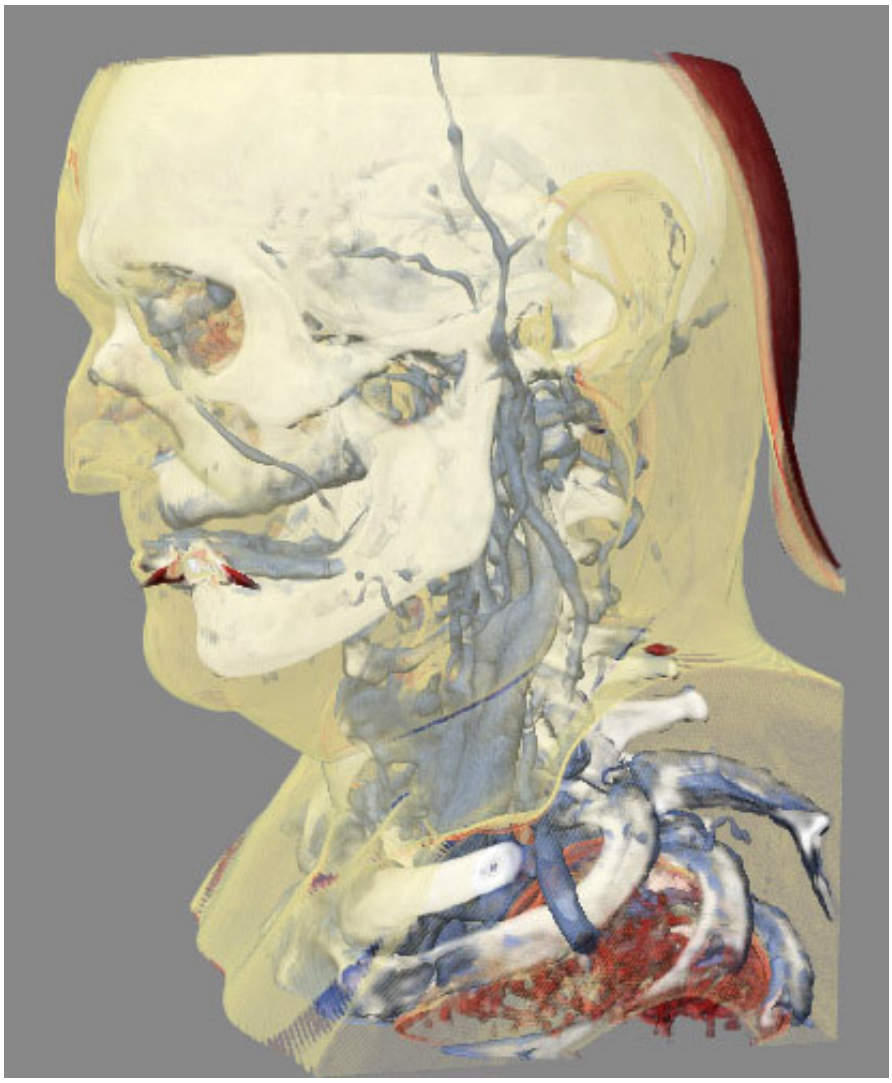


Figure 3.14: For this visualization the importance value has been used to modify the shading.

be good enough to get this impression. Therefore the reduction of the opacity is depending on the dot product of the normal vector $\vec{n}$ and the viewing direction $\vec{v}$. If this dot product is one then the surface orientation at the sample point position is parallel to the viewing direction. If it is zero then the surface orientation is perpendicular to the viewing direction. Sample points with a low value of the dot product belong to the silhouette. Equation 3.10 shows how the importance value together with the dot product is used to modify the opacity.

$$\alpha = \alpha * max\left(I, (1- \mid \vec{n} \cdot \vec{v} \mid)^{\delta}\right) * I^{\frac{1}{\delta}} \tag{3.10}$$

The α value used for the multiplication is the opacity directly from the transfer function. The max function is needed to reduce the opacity just for small importance values I and for sample points with a normal vector almost parallel to the viewing vector. The additional

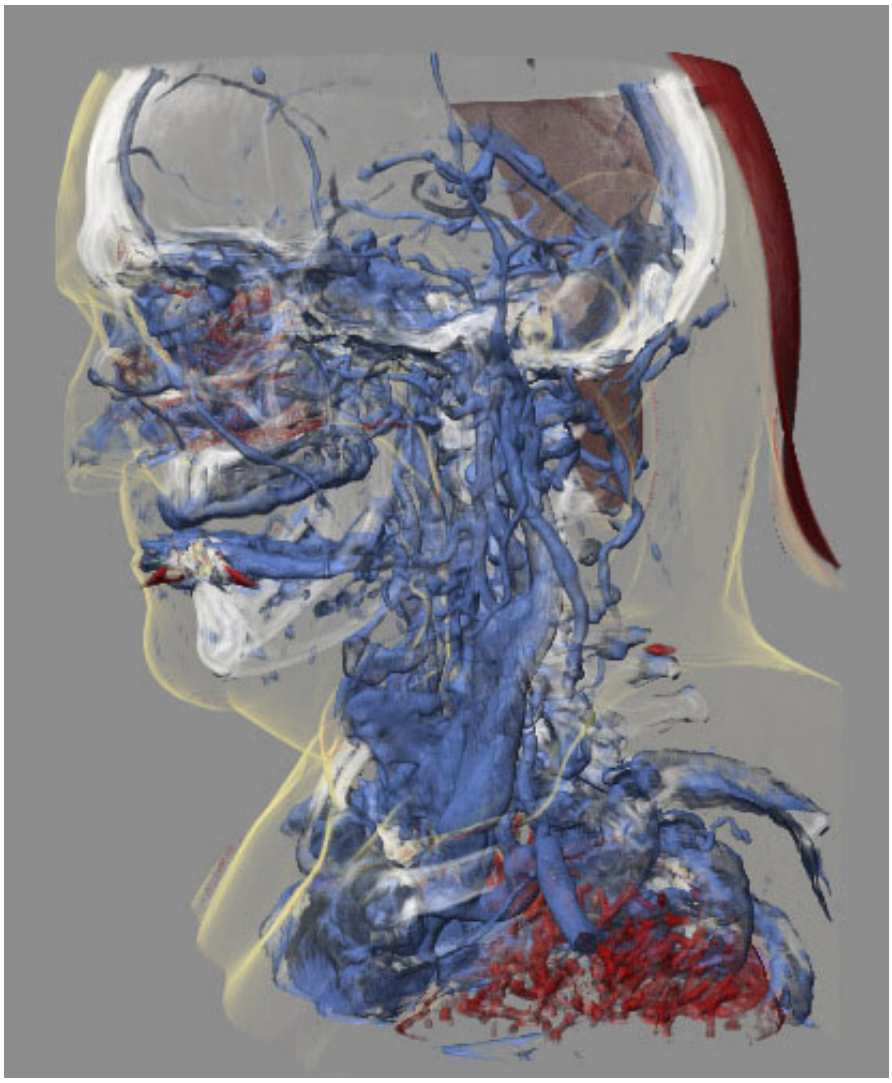


Figure 3.15: With reduction of the opacity for less important parts the more important parts are not occluded anymore. For the δ exponent a value of two was used.

multiplication with the importance value reduces the opacity in general for less important parts. The δ exponent is used to control the opacity reduction. A higher value makes the less important parts more transparent.

Figure 3.15 shows the result of the additional opacity modification together with the shading modification. The less important parts such as the skin and the bone structure are only represented by their silhouette. If we compare the result with Figure 3.14 then the difference is easy to see. The vessels inside the brain are now also visible. The remaining parts of the less important structures are enough to get an impression about their shape and position in relation to the vessels.

The final modification of an optical property should bring an additional enhancement of the contrast between more important parts and less important parts. To get this a simple approach by painters is used. They are drawing thicker outlines around parts which should be highlighted. In volume rendering this effect can be realized by enhancing the silhouette of the more important parts. For this reason the color at the silhouette outline of more important parts is darkened. So the silhouette seems to be thicker. The modification is dependent on the importance value and on the dot product between the viewing vector and the normal vector. The darkening is highest for important parts and sample points with a normal vector which is perpendicular to the viewing direction.

$$\vec{c} = \vec{c} * \max(1 - I, (\mid \vec{n} \cdot \vec{v} \mid)^{\epsilon}) \tag{3.11}$$

The color vector $\vec{c}$ is the already modified color from the shading modification. If the max function returns a low value then the darkening effect is high. A low importance value will always lead to a value close to one as result of the max function because the $(1-I)$ expression will be close to one. So the effect does only affect sample points with a high importance value and a low value for the dot product between the viewing vector $\vec{v}$ and the normal vector $\vec{n}$. The exponent ϵ effects the thickness of the outline. The smaller this value the thicker the outline.

Figure 3.16 shows the final rendering result of all three modifications. The final silhouette enhancement additionally emphasizes the more important parts. If we compare this result with Figure 3.15 then the enhancement of the contrast caused by the effect of the silhouette enhancement is clearly visible. In the final result after applying all three modifications of optical properties to the head data set the difference to standard volume rendering in Figure 3.13 is obvious. With the importance-driven emphasizing all parts of the vessel structure are visible without being occluded by less important parts.

In the shown example with the head data set we always used the Phong shading method for calculating the final color for the shaded parts. The Phong shading model is considered as a photorealistic shading model. In some cases the use of a non-photorealistic shading model can produce an enhancement of the final result. One of these non-photorealistic shading models is the Gooch cool-to-warm shading [17]. By assigning warm colors to points which are oriented toward the light source and cool colors to other points the depth perception of such shaded parts should be enhanced. In Figure 3.17 a comparison between Phong and Gooch shading is shown.

Until now all steps for creating an importance-driven visualization for a volumetric data set were shown. All methods are designed for a practical use. Everything from the transfer

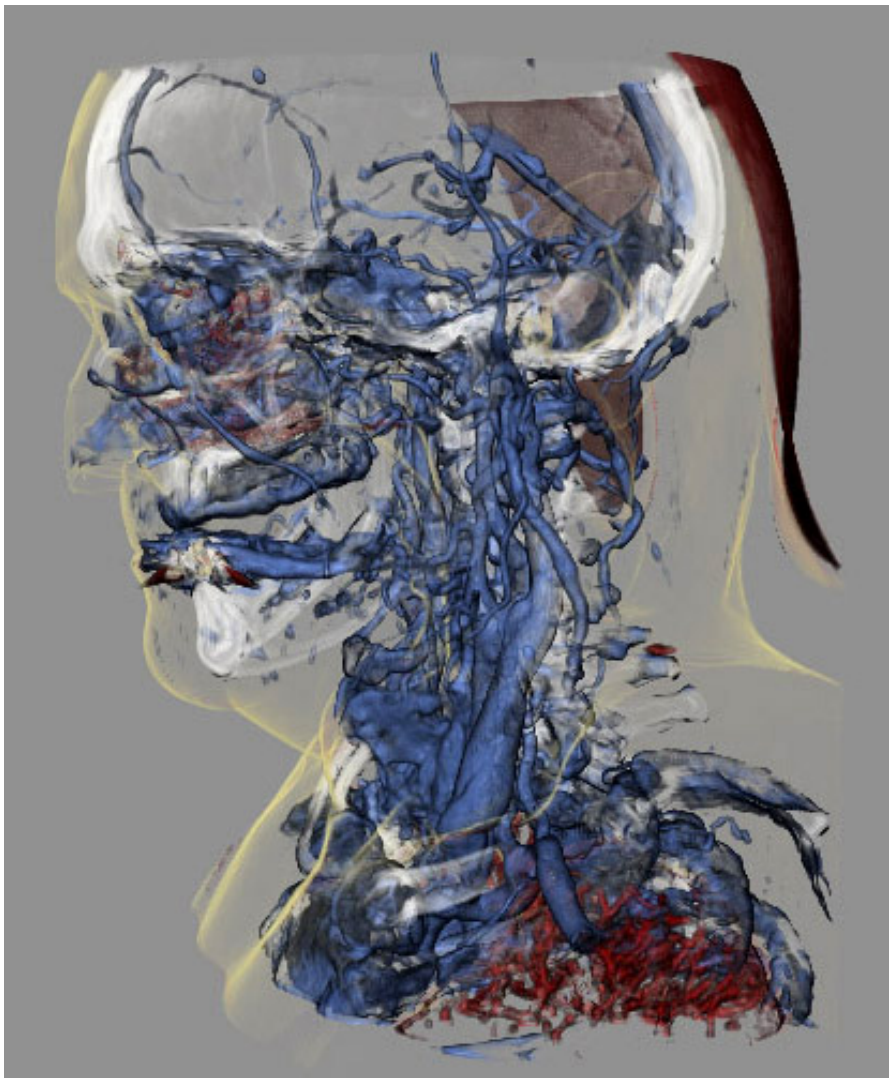
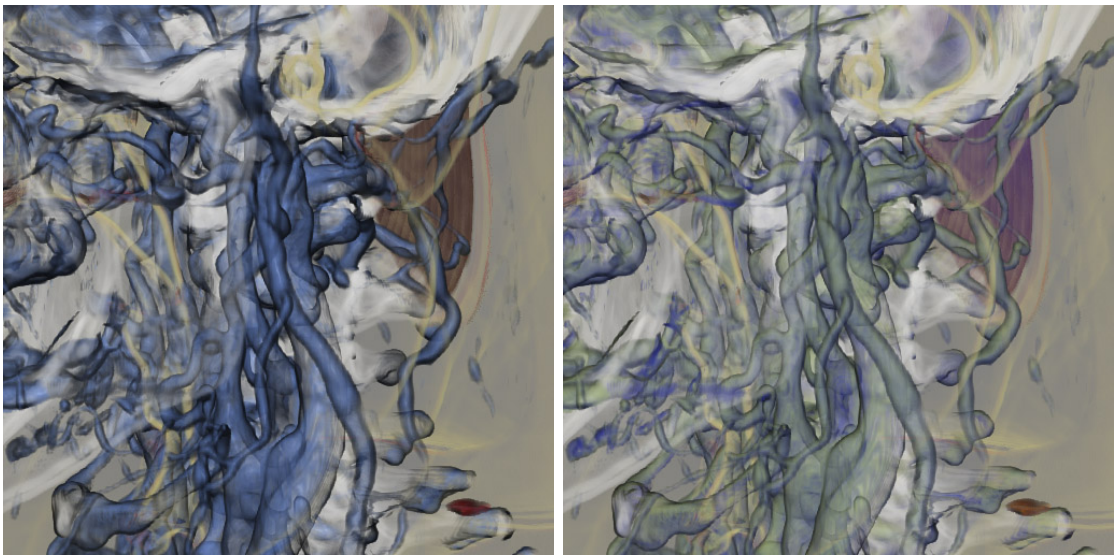


Figure 3.16: The image shows the final result of applying all three modifications of optical properties. For the ϵ exponent of the silhouette enhancement a value of 0.5 was used.

function through the filtering mask and the importance-driven emphasizing has to be defined only once for a given medical task. There should be no user interaction of a physician necessary to get this importance-driven visualization from a volume data set. Until now only the volume data set was considered. In the next section we will concentrate on the second input source, the region of interest. In the final section the region of interest is integrated into the visualization of the volume data set to get a merged visualization where all important parts for the intervention are visible at once without any occlusions by the less important parts.



(a) Phong shading (b) Gooch shading

Figure 3.17: Comparison between importance-driven Phong shading (a) and Gooch shading (b).

3.5 Contextual Cutaway Views

In the processing pipeline in Figure 3.1 it can be seen that there are two main input sources: the volumetric data on the left side and the object of interest on the right side. The previous sections described all processing steps which are applied on the volumetric data. In this section we will now concentrate on the object of interest.

The final goal is to get a merged visualization of the volumetric data together with the object of interest. In this resulting visualization a physician should be able to see all important parts of both input sources at once. The region of interest will always be one of these important parts. In the application of interventional imaging the object of interest will be an image or a small volume which shows a part of the body during the intervention. When, e.g., an ultrasound scan is used as interventional imaging system then the region of interest will typically be a wedge-shaped plane.

As mentioned in the introduction the object of interest is precisely aligned to the volumetric data. During the intervention the position of the object of interest will change because the physician uses the interventional imaging system to explore the body. The region of interest contains the information coming from the imaging system together with the information about its position in relation to the volume data set.

If the region of interest is somewhere in the middle of the volumetric data then it will be

hard to see. Parts of the volume data set will occlude the important object even if they are sparsely visually supressed. For this reason a solution is needed which makes these occluding parts of the volume transparent to see through it. A simple and often used technique to remove occluding parts of the volume is called clipping. Thereby a clipping plane in the volumetric space splits the volume into two parts. All sample points between the clipping plane and the viewer's position are clipped away. The effect is that the viewer can see the part right behind the clipping plane which would be normally occluded by the part of the volume in front of the plane.

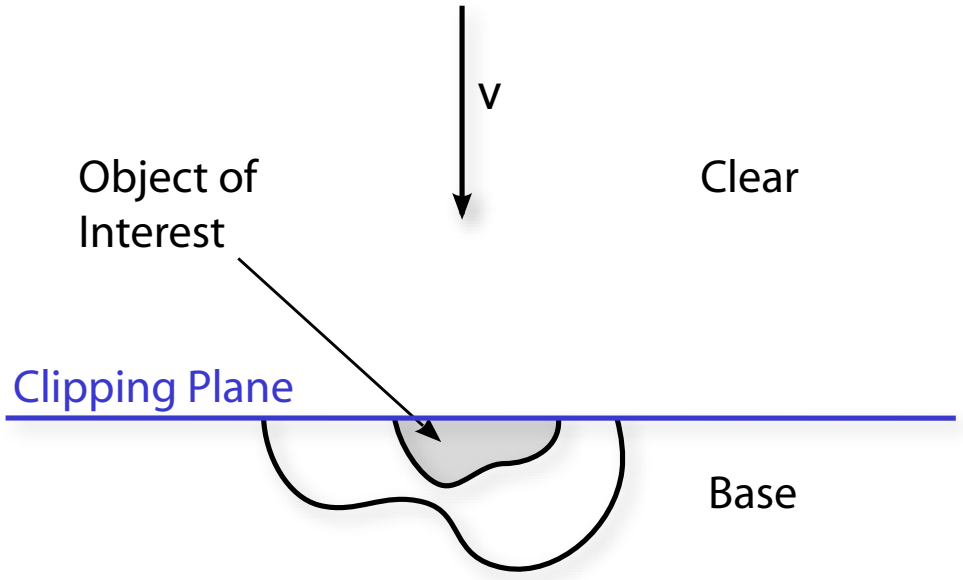


Figure 3.18: The concept of traditional volume clipping.

Figure 3.18 shows how the volume clipping could be used to see the object of interest inside the volume. The arrow labeled with v in the diagram indicates the viewing direction of an observer. The clipping plane divides the volume space in two regions. The region in front of the clipping plane will be called *clear* region and the region behind *base* region. In the *clear* region nothing of the volumetric data is visible. In the *base* region all parts of the volume are rendered as usual.

The volume clipping is very easy to implement but for the application with an object of interest inside the volume it is hard to find a good position for the clipping plane. If the clipping plane cuts the object of interest such as in Figure 3.18 then a part of this object is not visible in the resulting visualization. If the clipping plane is placed right in front of the

object of interest then some parts of the object will be occluded by regions of the volumetric data. Another problem with the volume clipping is that a lot of information is removed in the *clear* region which might be needed to get a better impression where the region of interest is located inside the volume. The use of a more complex function than just a simple plane for the division of the volume into two parts would be able to eliminate these drawbacks.

An often used technique for a more complex volume clipping is called cutaway view. In Figure 3.19 the principle of such a cutaway view is shown. Compared to the simple volume clipping with a single plane the cutaway view does not cut the object of interest. So the entire object can be seen from the observer's direction without any occlusion. The border between the *clear* region and the *base* region is not flat such as for the plane before rather it has the shape of a cone which starts at the silhouette line of the object of interest. The angle Θ controls the steepness of the cone. It is the angle between the viewing direction and the slope of the cone. The region of interest will always be visible from the point of view even with a Θ of 0. The conical shape of the clipping area has the benefit that fewer parts of the volumetric data are clipped away. So the impression of the position of the object of interest inside the volume is better.

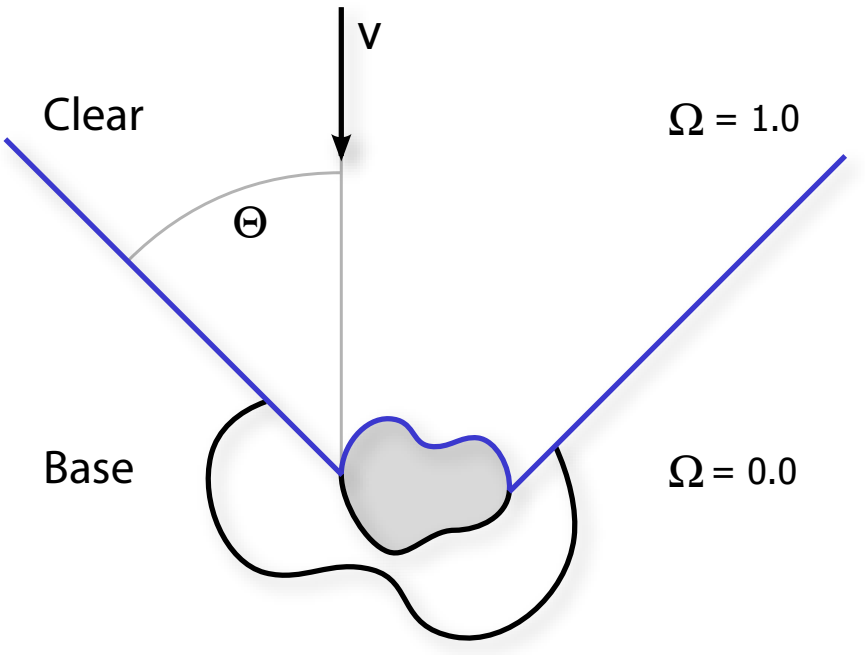


Figure 3.19: Principle of a simple cutaway view.

The partition of the space into the two distinct *clear* and *base* regions can also be formalized by defining an *occlusion function*, denoted as Ω. This *occlusion function* will represent

the degree to which a point in space occludes the object of interest. In a cutaway scenario such as in Figure 3.19, at a given point, $\Omega = 1$ if the point is inside the *clear* region and 0 if it is inside the *base* region. In eye space, a cutaway surface can be represented by a depth function $\xi(\Theta, p) = z$, where z is the depth of the cutaway surface with angle Θ at a given point p projected onto the surface. We can then define Ω for a given point in eye space as follows, where p_z is the z component of the point and $step(a, x) = 0$ if $x < a$ and 1 if $x \geq a$.

$$\Omega = step(\xi(\Theta, p), p_z) \tag{3.12}$$

The binary definition of the *occlusion function* suggests rendering can only have two modes: sparse (for the *clear* region) and dense (for the *base* region). This can be used for visualization where the object of interest is the only important part of the entire volume. In such a case no part of the important region will be occluded by any other part of the volumetric data set.

For the application proposed in this thesis it is possible that not only the region of interest is important during the intervention. For the liver biopsy, e.g., also the vessel structure is important. In Section 3.3 the assigning of an additional importance value to each sample point was described. Which parts of the volume should be visible in the cutaway region should be dependent on this importance value. The simple cutaway view is not complex enough because the *occlusion function* just returns 0 or 1. We want to have more control over the rendering of materials with multiple importance values. Therefore the simple cutaway definition must be generalized by modifying it in a way to also allow occlusion values between 0 and 1.

As first modification of the simple cutaway definition a second cutaway surface is included defined by a wider angle. This new region, which is denoted as *transition* region, can have an occlusion function that varies between the two cutout surfaces. This is defined as shown in Figure 3.20, allowing us to determine the cutout angle of points located in the *transition* region, relative to the two bounding angles. This will ultimately allow a variation of visibility in the image outside the silhouette of the projected object of interest by letting us cut or fade away materials at varying angles.

To control the visibility in the image over the object of interest, another region is added, the *overlay* region. This region is bounded by the cutaway surface of Θ_1 offset by a thickness d towards the camera, as shown in Figure 3.20. In this additional *overlay* region only the most important parts of the volumetric data should be visible to avoid a complete occlusion of the object of interest. If both angles of the cutaway surface are the same and the thickness d is 0 then we have again the simple cutaway view as shown in Figure 3.19.

Considering these four regions, the *occlusion function* for a given point in eye space is defined as follows, where Θ_1 and Θ_2 are the cutaway angles, d is the thickness of the *overlay*

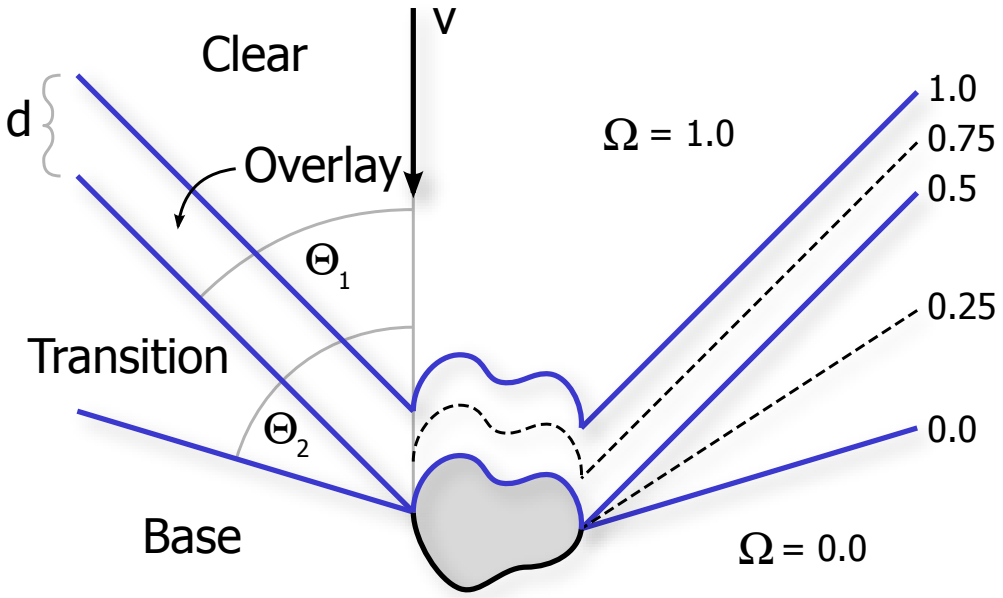


Figure 3.20: Given an object of interest (gray) and a viewing direction, a cutaway with distinct *base*, *transition*, *overlay*, and *clear* regions can be constructed, as shown in this cross section diagram. An occlusion function Ω in 3D space is defined to modify opacity in different areas of the cutaway structure.

region, and $ramp(a, b, x) = 0$ if $x \leq a$, 1 if $x > b$, and is a linear ramp from 0 to 1 for $a < x \leq b$.

$$\Omega = \frac{ramp(\xi(\Theta_2, p), \xi(\Theta_1, p), p_z) + ramp(\xi(\Theta_1, p), \xi(\Theta_1, p) + d, p_z)}{2} \tag{3.13}$$

This definition results in $\Omega = 0$ for points in the *base* area, $\Omega = 0.5$ for points on the *transition-overlay* boundary, and $\Omega = 1$ for points in the *clear* area, with the appropriate linear blending for points in between the boundaries, as shown in Figure 3.20.

In the next section both parts of the processing pipeline are combined to get the final merged visualization. The *occlusion function* of the contextual cutaway view is used to manipulate the importance value of the sample points in the volumetric data. Therefore the final importance value of a sample point is also dependent on its position in space in relation to the cutaway view.

3.6 Modifying Importance by Occlusion

In the previous section a flexible cutaway structure was defined. In this section this cutaway structure will be combined with the importance-driven rendering technique introduced in Sections 3.3 and 3.4. The *occlusion function* will be used to modify the visualization for different regions in the cutaway area. The modification should lead to a final visualization where all important parts for a certain medical task can be seen at once. The less important parts should be visible as context information to get a better orientation.

As described in the previous section the contextual cutaway view is divided into different areas. The modification of the visualization should be dependent on the area in which a sample point lies. The *base* area does not occlude any important part from the observer's viewpoint. Therefore this area should be rendered very dense to get a good impression of the position of the region of interest. In contrast to that the *clear* area is between the observer and the region of interest. Therefore no parts in this area should occlude the important parts behind them. In the *overlay* area only the most important parts of the volume data set should be rendered to have a sparse visualization which does not occlude the entire object of interest. Finally the visualization of the sample points in the *transition* area should be modified in a way to get a smooth transition from the *base* area to the *overlay* area.

The importance value introduced in Section 3.3 is used in Section 3.4 to highlight the more important parts in contrast to less important parts. The importance value is only depending on the position of a sample point in the transfer function space. The *occlusion function* from the previous section additionally assigns an Ω value to each sample point in the volume depending on the position in relation to the object of interest. The idea now is to use this Ω value to modify the importance value. This modification should reduce or increase the importance value of a sample point depending on the area in which it lies. Equation 3.14 shows the formula for this modification.

$$I_m = ramp(\tau_l, \tau_u, I) \tag{3.14}$$

The *ramp* function is the same function as described for Equation 3.13 in the previous section. I is the original importance value from the transfer function and I_m is the modified importance value. τ_l and τ_u are the threshold values for the ramp. τ_l is the lower threshold. If I is lower than this value then it will be modified to 0. If I is higher than the upper threshold τ_u then it will be modified to 1. For an importance value between these two thresholds the resulting importance I_m will be calculated by interpolation. To include the contextual cutaway in this modification the threshold values are dependent on the Ω from the *occlusion function*.

The following two equations describe the influence of the *occlusion function* on the threshold values:

$$\tau_l = \left(\frac{floor(2 * \Omega)}{2}\right)^{\sigma} \tag{3.15}$$

$$\tau_u = \min(1, 2 * \Omega) \tag{3.16}$$

The lower threshold τ_l is always smaller or equal to the upper threshold τ_u. Both thresholds are only depending on the Ω value and an additional σ coefficient which is needed to control the threshold for importance clipping. In Figure 3.21 a graph is given which shows the relation between Ω and the modification of the importance value.

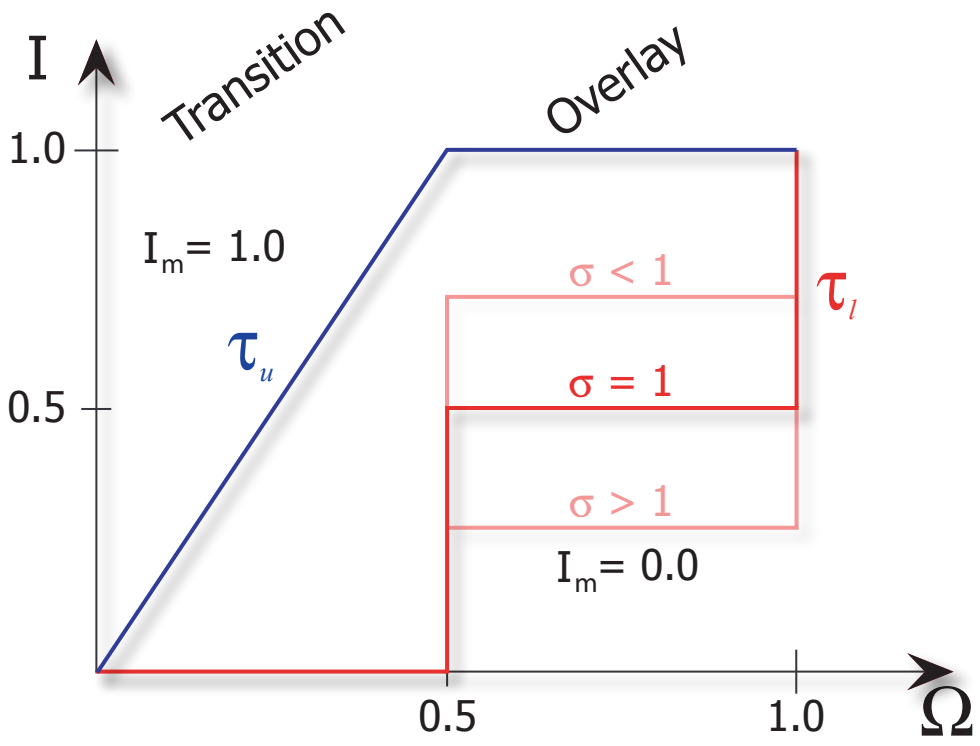
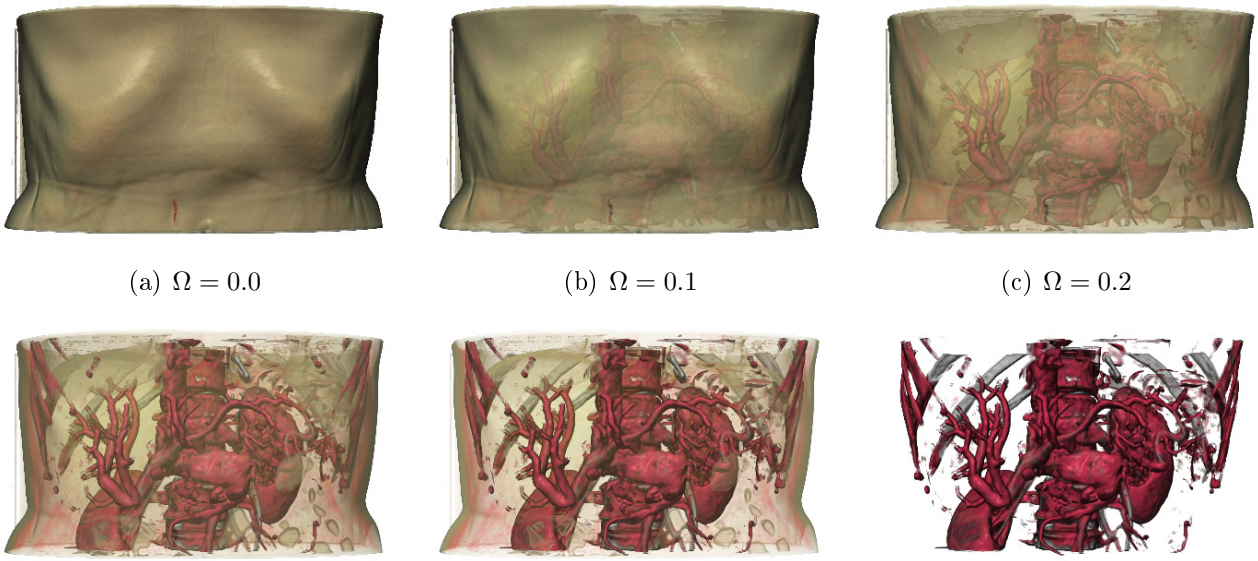


Figure 3.21: The graph shows the relation between Ω and the importance modification.

For an Ω value of 0 in the *base* area both thresholds are 0. This results in an I_m value of 1 for each importance value I greater than 0 and therefore in a dense visualization of all parts. In the *transition* area the lower threshold is always 0 while the upper threshold increases to 1 towards the border to the *overlay* area. This increase of τ_u fades out the less important parts. In the *overlay* area τ_u stays constant at 1 and τ_l jumps from 0 in the *transition* area to 0.5^{σ}. The σ is used to modify the lower threshold value for this area. A high σ makes the threshold lower and a low σ makes it higher. The lower this threshold is the more less important parts



(a) $\Omega = 0.0$ (b) $\Omega = 0.1$ (c) $\Omega = 0.2$

(d) $\Omega = 0.3$ (e) $\Omega = 0.4$ (f) $\Omega = 0.5$

Figure 3.22: Effect of different, fixed Ω values on the importance value. The vessel and the bone structures have the highest importance value in this example. An Ω value greater or equal than 0.5 and smaller than 1 will always result in Image (f). For an Ω value of 1 all parts would vanish.

can be seen in the *overlay* area. Finally in the *clear* area both threshold values are at 1. So all importance values are being modified to 0 and everything is clipped away in this area.

The sequence of images in Figure 3.22 shows the impact of different Ω values on the resulting visualization. In these results the same Ω value was always applied on all sample points to highlight the effect of different Ω values in different areas of the contextual cutaway. In the data set used for the results the vessel and bone structure has the highest importance with a value of 1. The importance value for all other parts is lower than 0.5. A value of 1 is used for the σ coefficient in the lower threshold formula.

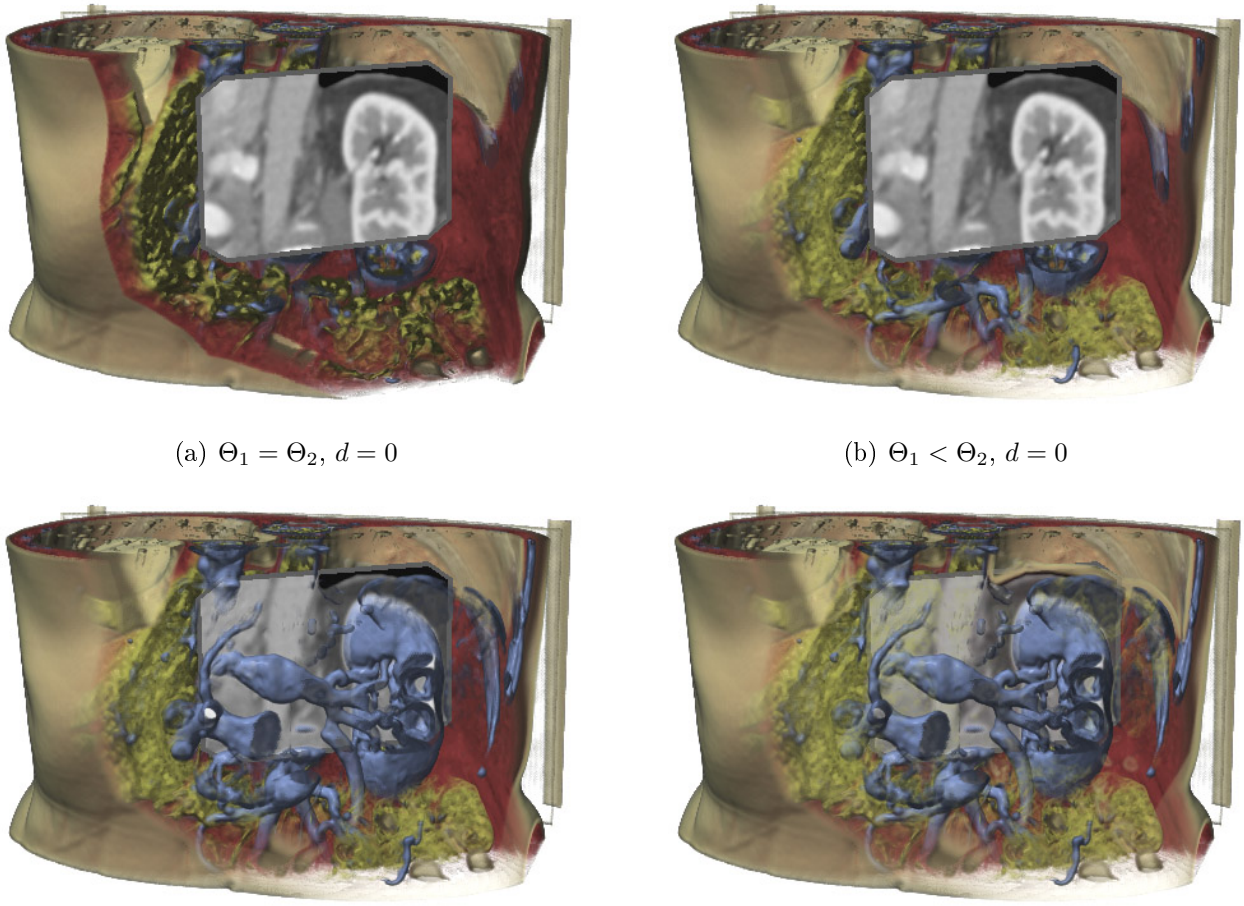
For an Ω value of 0 all importance values are modified to 1 as described before. This results in a very dense visualization of all parts as in Figure 3.22 (a). Even parts with an originally low importance value such as the skin are visualized with all details. In the contextual cutaway view this visualization would correspond to all parts in the *base* area because the Ω value is 0 in this region. The results in Figure 3.22 (b) to (e) would belong to the *transition* area of the cutaway view. The Ω value there is greater than 0 and smaller than 0.5. The modified results show a smooth fade-away of the less important parts with a rising Ω value. Figure 3.22 (f) finally shows a result where only the most important parts are left.

Such a visualization would be used for the *overlay* area where non-important parts should not occlude the object of interest. For each Ω value greater or equal than 0.5 and smaller than 1 the result will always look the same as in Figure 3.22 (f) because there should be no fade-away in the *overlay* area to keep the sharp edges of the important parts. For an Ω value of exactly 1 such as in the *clear* area all parts would be completely transparent.

Now everything is defined which is needed to get the desired visualization from the two input sources. The object of interest is used to create the contextual cutaway view inside the volume data set. The Ω value for each region of the cutaway view is then used to modify the visualization. Depending on the needs of the final visualization the parameters of the contextual cutaway view and for the importance modification can be used to adjust the resulting visualization.

In Figure 3.23 examples with different parameters for the cutaway angles, the *overlay* thickness, and the σ exponent for the lower threshold τ_l are shown. Figure 3.23 (a) shows an example of a traditional cutaway view. The *transition* area and the *overlay* area are not present. In Figure 3.23 (b) the angles Θ_1 and Θ_2 are different and, therefore, the *transition* area is not zero. A smooth fade out of less important parts can be seen. In Figure 3.23 (c) and (d) the thickness of the *overlay* area is set to a value greater than 0. Additionally a higher σ value is used in Figure 3.23 (d) so the lower threshold in the *overlay* area is lower and more less important parts such as the hull of the lung can be seen in this area.

The previous sections described every step of the processing pipeline in Figure 3.1. By applying all these steps to the input sources we will end up in the desired visualization. All settings for parameters and the design of filters and the transfer function has to be done only once for a given medical task, such as the liver biopsy, on a particular medical imaging device. During practical use no interaction by a physician should be necessary to get the final visualization. This is very important because physicians are not interested in setting parameters. Therefore the introduced algorithms are well suited for a practical use. In Chapter 5 results for a special medical task are shown. The next chapter briefly describes the implementation of the algorithms to get a fast visualization of what is very important for a use in real-time medical application such as interventional imaging.



(a) $\Theta_1 = \Theta_2$, $d = 0$

(b) $\Theta_1 < \Theta_2$, $d = 0$

(c) $\Theta_1 < \Theta_2$, $d > 0$, $\sigma = 0.1$

(d) $\Theta_1 < \Theta_2$, $d > 0$, $\sigma = 10$

Figure 3.23: The effects of various parameters in the cutaway structure are shown in the above images. Image (a) shows a traditional cutaway view where the other three images show a smooth transition and an overlay area in front of the important part.

Chapter 4

Implementation

Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest.

Isaac Asimov

In the previous chapter all steps were described which are needed to generate a merged visualization of both input sources where the most important parts are emphasized. This chapter now will explain how these steps are realized in practice. As mentioned in the introduction it is very important that the final application is able to produce rendering results in real-time. The physician wants to have a direct visual feedback from the application when the ultrasound probe is moved. If there is a lag between the movement and the visualization then the navigation is much harder.

The use of graphics hardware is essential for the implementation of a fast visualization because GPUs are more efficient at manipulating and displaying computer graphics than CPUs. In Section 2.1 some methods were discussed which are designed for rendering utilizing the GPU. For the implementation of the new approaches hardware-based raycasting seems to be the best choice because it will be easier to integrate the region of interest and the contextual cutaway view in a raycast algorithm than in other algorithms.

The entire implementation was done at Siemens Corporate Research (SCR) in Princeton, NJ. The implementation is especially designed for a liver intervention application (see Section

1.1). The region of interest is an ultrasound plane and the volumetric data results normally from a CTA scan. The registration of both input sources is done through an application which was implemented by the CAMP (Computer Aided Medical Procedures & Augmented Reality) institute of the Technical University in Munich in cooperation with SCR. For the integration of the ultrasound plane in the visualization this application provides the ultrasound image together with a set of points which are defining the geometry of the ultrasound plane in the space of the volume data set.

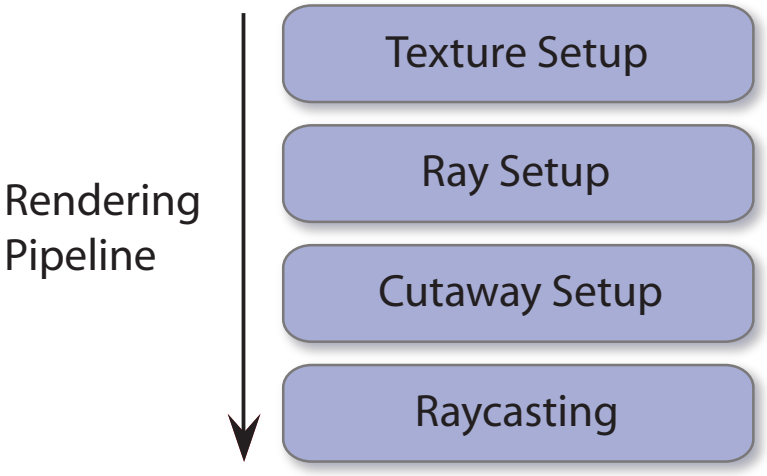


Figure 4.1: Necessary steps which are done to get the final rendering result.

The entire rendering routine was implemented as part of the *RadBuilder* (Rapid Application Development Builder) project. The goal of this project is to provide an environment for the fast development of applications for medical visualizations. For this reason an application has been implemented which provides a graphical user interface for the creation of networks based on OpenInventor ([46], [47]). For the development of new applications elements with new functionalities can be implemented which then can be used in the network. With this network technique the entire rendering process can be split into different processing steps. Figure 4.1 shows the necessary steps for the liver intervention application.

For each frame all the steps of the rendering pipeline have to be applied except the texture setup which has to be done only when one of the textures has changed. In the following sections all processing steps of this pipeline are described briefly.

4.1 Texture Setup

As described in Section 2.1 the data set has to be transferred to the graphics card first to have a fast access to the data during the rendering algorithm. All data sets are treated as textures on the graphics card. The transfer of the data sets to the graphics hardware is done by defining textures which contain the values from the data sets.

In general there are two possibilities how a texture can be defined: as two-dimensional texture or as three-dimensional texture. Besides the dimensionality there are some other properties which have to be defined when the texture is created. Each texture consists of one to four channels. Four channels, e.g., can be used to store a color as RGB value and an opacity value at each point in the texture. Furthermore the data format for the storage of the texture has to be defined. The more bits are used for the storage of each point in the texture the more accurate they are but the more memory it takes to save the texture. Depending on the origin of the data set these definitions have to be set. For the rendering algorithm there are three data sources which are needed:

- Volume data (3D)
- Transfer function (2D)
- Importance function (2D)

The volume data is the CTA data set. It has three dimensions and it has to be stored in a three-dimensional texture. In a CTA data set each voxel has a single value and this value is typically saved in 12 bits. So the texture needs only one channel and the 12 bits for each value can be saved in a data format with a size of 2 bytes. To use all 16 bits of the data format the voxel value range is scaled. After the scaling each voxel can have a value between 0 and 65536. This scaling brings no improvement of the accuracy but it saves an additional processing step in the raycast algorithm.

The second data set which is needed in the raycasting step is the transfer function. For this application a two-dimensional transfer function is used. Therefore it has to be stored in a two-dimensional texture. The transfer function texture is needed as lookup table. The value and gradient magnitude of a voxel are used as coordinates for getting the appropriate entry in the transfer function. The transfer function returns a color and an opacity for each voxel. Therefore the texture has to have four channels for RGB values and an opacity value.

As described in Section 3.3 the importance value should normally directly result together with the color and opacity from the transfer function. In the implementation this is not

possible because the maximum number of channels for a texture is four and these four channels are needed to store the color and opacity. So there is no more space in a single texture to store the importance value. Therefore the importance value has to be saved in a separate texture with only a single channel. In the raycasting algorithm this texture is used exactly in the same way as the transfer function texture.

The texture setup has to be done only before the rendering is started or when one of the data sets has changed. This can happen for the transfer function texture and the importance function texture when they have to be adjusted to change the appearance of the rendering result.

4.2 Ray Setup

In this processing step the texture for the entry and exit points of the ray are generated. As described in Section 2.1 for this reason a color cube is rendered two times with different settings. To generate the texture of the ray exit points the cube is rendered with front face culling. With this setting only the faces are rendered which are pointing away from the observer. The generation of the texture for the entry points works the same way but instead of the front faces the back faces are culled.

With the additional ultrasound plane which is located somewhere in the volume this method has to be slightly modified because a ray should terminate when it hits the ultrasound plane. The geometry points for the ultrasound plane are already given in relation to a normalized volume space from $(0, 0, 0)$ to $(1, 1, 1)$. The position of each point can directly be used as color and it will match with the color cube model. To generate the entire exit points texture the ultrasound plane is rendered first. Then the back faces of the cube are rendered into the same buffer whereby the depth test discards all points which are behind the ultrasound plane. The buffer contains finally the real exit points of the ray.

Both textures for the entry and exit points are saved in a texture by accessing the buffer after the rendering. In the raycasting step they are used for defining the ray entry point and the ray direction through the volume as well as the length of a ray (step size).

4.3 Cutaway Setup

The cutaway setup is the next step in the rendering process. In this step a texture is generated which contains the depth position of each point of the two cutaway cones. This is needed to

calculate the occlusion function as described in Section 3.5. The easiest way for generating this texture is to render a geometry which represents the cone. In the fragment program the depth of each fragment can then be stored in one of the color channels of the texture. To get the depth of both cones two different cones have to be rendered. The depth of the second cone can be stored in the same texture in another color channel.

The starting points of the cones are given by the geometry points of the ultrasound plane. To create the cone surface additionally points are needed which define the end of the cone. These points are called outer points. In general there are two possibilities to generate the geometry of the cones. It can be done either in a software-based or a hardware-based approach. In the software-based approach the outer points are calculated on the CPU and then the complete geometry of the cone is transfered to the graphics card. In the hardware-based approach the position of the outer points is calculated in the vertex program on the GPU. This is faster than the software-based approach therefore this approach has been used for the application to achieve real-time rendering.

With the used graphics hardware for the implementation it is not possible to generate new points in a shader program but the position of the points can be changed in the vertex program. Therefore a quad array is rendered to create the cutaway hull. Instead of calculating and passing the outer points directly to the graphics hardware the bounding points of the ultrasound plane are passed twice to the graphics card for the definition of a new quad in the array. The first time these points are drawn with an option not to be changed in the vertex program and the second time they are drawn without this option. Therefore the position of the points is changed in the second time when they are drawn to the position of the outer points.

Figure 4.2 shows the scheme of the point modification. In Listing 4.1 the corresponding code of the vertex program is shown.

The previous and the next point are passed as variables to the vertex program. The first processing step in the program is to transform the points by the modelview matrix to get their position on the image plane. The x and y positions of the transformed points are used to generate vectors from the previous point to the current point and from the current point to the next point. These vectors are used to calculate the slope direction of the cutaway cone at the current point. In the final step the current point is moved along this vector. How far it is moved is dependent on the predefined angle of the cone.

Listing 4.1: Part of the vertex program for the cone setup.

```
1 // transform points by the modelview matrix
2 vec4 pThisT = gl_ModelViewMatrix * gl_Vertex;
3 vec4 pNextT = gl_ModelViewMatrix * vec4(pNext, 1);
```

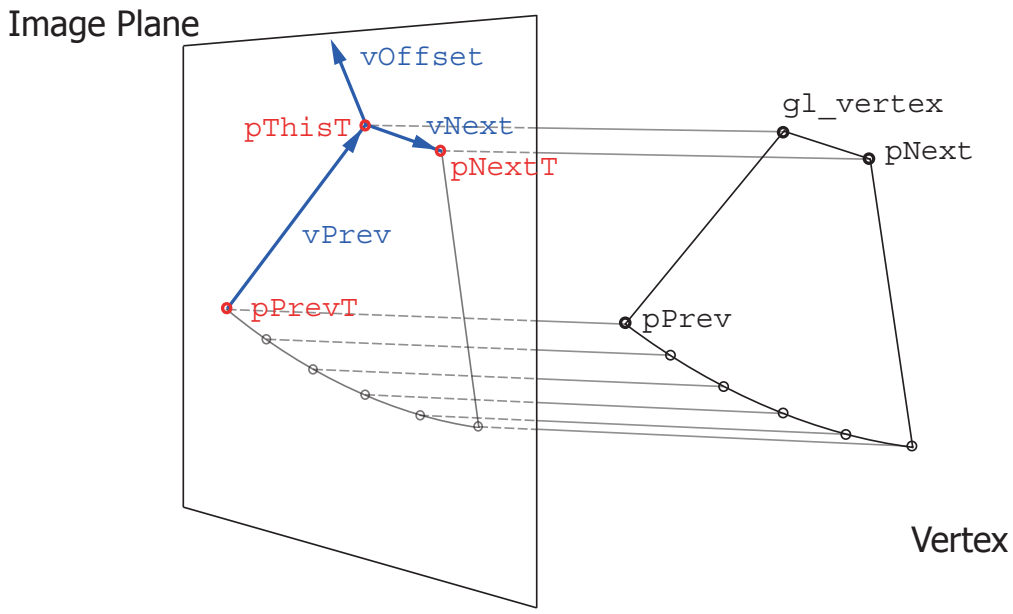


Figure 4.2: Determination of the direction for the cutaway cone.

```
4  vec4 pPrevT = gl_ModelViewMatrix * vec4(pPrev, 1);
5
6  // extract xy direction from the transformed points
7  const vec2 cThis = pThisT.xy;
8  const vec2 cNext = pNextT.xy;
9  const vec2 cPrev = pPrevT.xy;
10
11 // the ccw variable indicates whether the three points are arranged
12 // clockwise or counterclockwise
13 float ccw = cPrev.x * (cThis.y - cNext.y) - cThis.x * (cPrev.y - cNext.y)
14           + cNext.x * (cPrev.y - cThis.y);
15
16 // calculate the previous and next vector
17 const vec2 vNext = normalize(cNext - cThis);
18 const vec2 vPrev = normalize(cPrev - cThis);
19
20 // calculate the offset vector
21 vec2 vOffset;
22 if (abs(ccw) < threshLinear) // assume the points are colinear
23 {
24   vOffset = vec2(vNext.y, -vNext.x);
25 }
26 else
27 {
28   const float cos_gamma = dot(vNext, vPrev);
29   const float offsetLength = 1.0/sqrt((1 - cos_gamma)/2.0);
30   vOffset = -sign(ccw) * offsetLength * normalize(vNext + vPrev);
31 }
32
33 // calculate the new extruded point in dependency of the predefined theta angle
34 vec4 pExtrudeT = vec4(length(pThisT.xyz) * tan_theta * vOffset, 0, 1);
35
36 // set the built-in position and texture coordinate according
37 // to the extruded point
38 gl_Position = gl_ProjectionMatrix * pExtrudeT;
39 gl_FrontColor = gl_Color;
40 gl_TexCoord[0] = pExtrudeT;
```

The drawing of the quad array as described is done two times with different settings for the angle according to both cutaway cones. As mentioned before the depth of each fragment is then stored in one of the color channels of the texture. In the raycasting the depth of both cones can be read from this texture and the occlusion can be calculated according to Formula 3.13.

4.4 Raycasting

The previous steps created all necessary textures which are now used for the raycasting. In the hardware-based raycasting the main processing part is done in the fragment shader using the technique proposed by Krüger and Westermann [30]. To get the final result a ray is cast into the scene from each pixel. To obtain this in a hardware-based approach a quad is drawn over the entire rendering area. This causes every pixel to be treated by the fragment program.

The first processing step in the fragment program is to check if the ray starting from the considered pixel hits the volume. This can be done easily by a lookup in the texture for the ray entry points. If this texture returns no color and opacity for the considered pixel then the ray from this pixel does not intersect the volume. In such a case the raycasting algorithm is skipped in the fragment program. If there is a color value in the entry points texture then also the exit point is acquired by a lookup in the texture for the exit points.

From the entry and exit points of the ray the length and direction of the ray is calculated. Before the raycasting loop is started also the depth values for both cutaway cones at the given pixel are acquired by another texture lookup. With these parameters the loop can be started. In the loop the color and the opacity of points along the ray starting at the entry point are calculated. The color and opacity of each point is blended with the color and opacity of the previous points as described by Levoy [32] using front-to-back compositing. The loop is canceled when the combined opacity is higher than a certain threshold (early ray termination) or when the exit point of the ray is reached.

For each sample point along the ray several processing steps have to be applied to get the color and opacity for this point according to the methods introduced in Chapter 3. Figure 4.3 shows all these steps and which information is acquired in each of these steps. The Listing 4.2 shows the corresponding part of the fragment program which executes these processing steps.

Listing 4.2: Part of the fragment program for the raycast algorithm.

```
1 // volume texture lookup
2 voxelValue = texture3D(volume, rayPos).r;
3
```

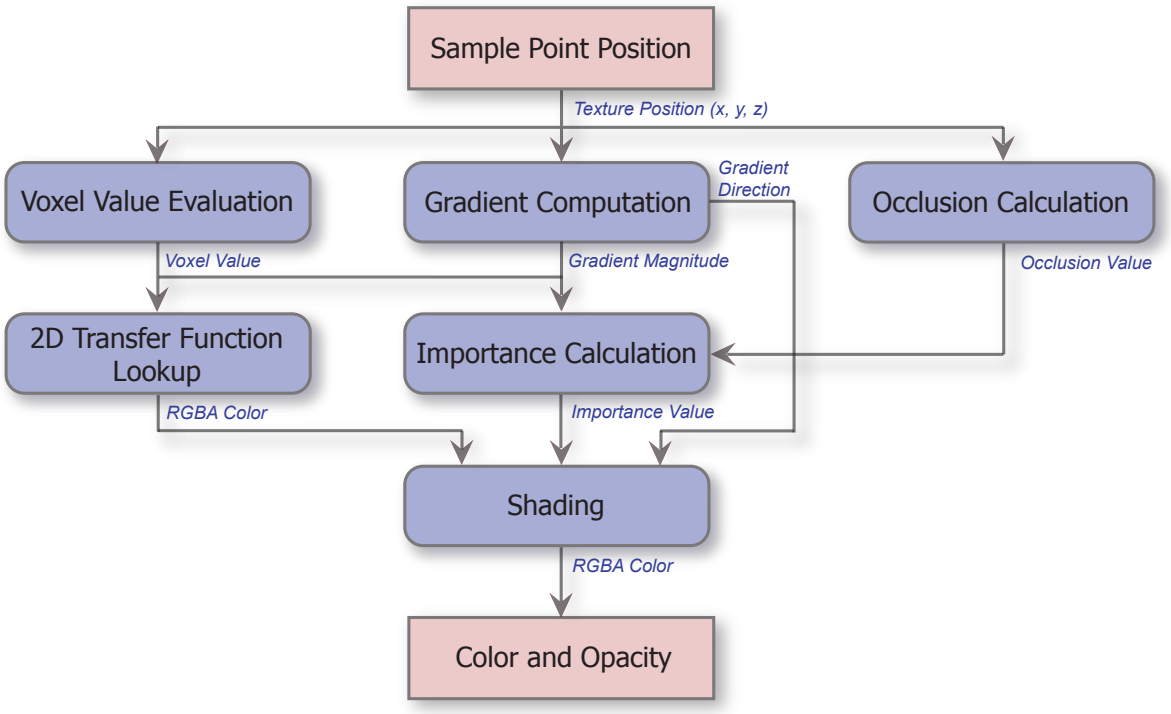


Figure 4.3: Processing steps which are applied on every sample point along the ray to get a final color and opacity for this point.

```
4  // gradient computation (central difference method)
5  vec4 gradient;
6  vec3 sample1;
7  vec3 sample2;
8  const vec4 offset = vec4(gradStep, 0);
9  sample1.x = texture3D(volume, rayPos-offset.xww).r;
10 sample2.x = texture3D(volume, rayPos+offset.xww).r;
11 sample1.y = texture3D(volume, rayPos-offset.wyw).r;
12 sample2.y = texture3D(volume, rayPos+offset.wyw).r;
13 sample1.z = texture3D(volume, rayPos-offset.wwz).r;
14 sample2.z = texture3D(volume, rayPos+offset.wwz).r;
15 gradient.xyz = (sample2 - sample1); // gradient direction
16 gradient.w = length(gradient.xyz);  // gradient magnitude
17 gradient.xyz /= gradient.w;         // normalize gradient
18
19 // 2D transfer function lookup
20 sample = texture2D(transferFunction, vec2(voxelValue, gradient.w));
21
22 // occlusion calculation; the cutoutSample is the texture which holds the depth values
23 // of both curaway cones
24 float occlusion = 0.5*smoothstep(cutoutSample.g, cutoutSample.r, rayPos.z)
25                 + 0.5*smoothstep(cutoutSample.r, cutoutSample.r+overlayT, rayPos.z);
26
27 // importance calculation
28 float importance = texture2D(importanceTexture, vec2(voxelValue, gradient.w)).r;
29 float tau_l = pow(floor(2.0*occlusion)*0.5, sigma);
30 float tau_u = min(1, occlusion*2.0);
31 importanceM = smoothstep(tau_l, tau_u, importance);
32
33 // shading
34 float ndotv = abs(dot(gradient.xyz, viewDir));
35 float diffuse = max(dot(gradient.xyz, lightDir), 0.0);
36 float specular = pow(max(dot(viewDir, reflect(-lightDir, gradient.xyz)), 0.0), specCoefficient);
```

```
37 // shading modification
38 sample.rgb = ((ka + kd*diffuse) * sample.rgb + ks*specular*sample.a) * importanceM  // Phong shaded part
39             + sample.rgb * (1.0 - importanceM);                                     // unshaded part
40 // opacity modification
41 sample *= max(importanceM, pow(1 - ndotv, gamma)) * pow(importance, 1/gamma);
42 // silhouette enhancement
43 sample.rgb *= max(1 - importanceM, pow(ndotv, epsilon));
```

To speed up the entire rendering process some improvement techniques can be used. For example the gradient can be pre-calculated and stored in a texture to avoid the 6 texture lookups which are necessary if the gradient is calculated on-the-fly in the fragment program. Another improvement technique is the empty space skipping where sample points are skipped if they are positioned in a part of the volume which has no impact on the final rendering result. The final goal of the speed improvements is to get a frame rate which is high enough for a real-time visualization.

To get the final result image the ultrasound plane is rendered first on the screen and then the raycasting algorithm is applied. This results in the expected merged visualization where all important parts of both input sources are visible together. In the following chapter some results are shown and discussed which are generated with the described rendering method.

Chapter 5

Results

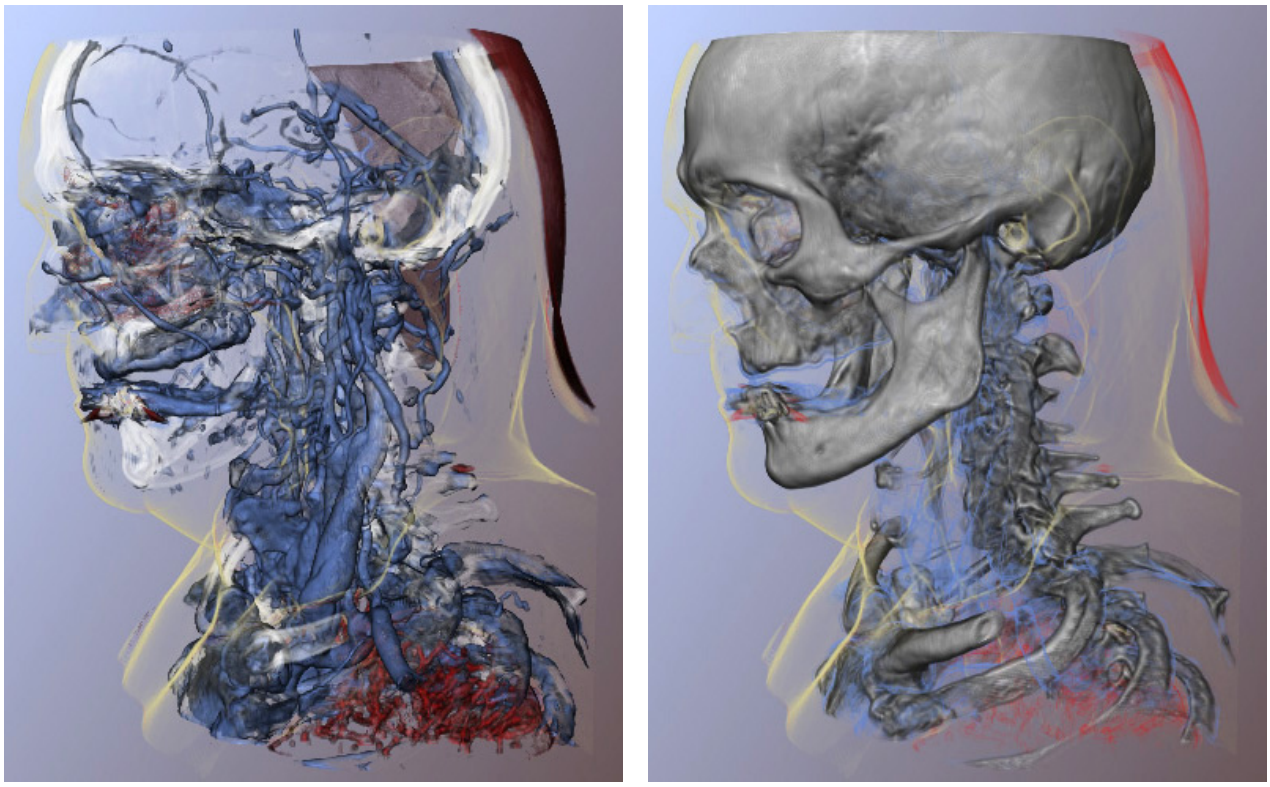
Results! Why, man, I have gotten a lot of results. I know several thousand things that won't work.

Thomas A. Edison

This chapter concentrates on concrete results which are generated with the methods described in Chapter 3. Before the methods can be applied all the input sources must be prepared. In case of liver intervention this will be a volume data set and an image coming from the interventional imaging system. Normally this will be an ultrasound image. Additionally to the input sources some parameters and other data sources have to be defined beforehand. The parameters are used to define the contextual cutaway view and to control how much of the less important parts are visible in the different cutaway areas. An additional data source which is needed for the visualization is the transfer function which provides color, opacity, and importance values for every voxel in the volume data set.

The importance value for each component in the transfer function space is the most important value. It is needed to highlight the most important parts of the volume data set. In Figure 5.1 a head is rendered two times with different importance values. In Figure 5.1(a) the importance value of the vessel structure in the head and in the lung is set to 1. The other tissues are considered as less important and have a low importance value. In contrast to that in Figure 5.1(b) the bone structure is considered as the most important part. Comparing the two figures it can be seen that the importance value has a very high impact on the final

result. The importance value has to be defined carefully for a given medical task to highlight the most important parts.



(a) High importance value for the vessels (b) High importance value for the bone structure

Figure 5.1: Different rendering results for different importance values.

For a special medical task such as the liver intervention all these parameters and also the transfer function can be defined beforehand. This has the benefit that a physician gets a rendering result adapted to the medical task without changing any settings. For CTA scans also the transfer function and the importance function have to be designed only once for a certain CT screening device.

For the results ultrasound exams recorded on patients and volunteers are used. The spatial location and orientation of the ultrasound probe has been tracked during the acquisition. Before the visualization the ultrasound planes are aligned with the corresponding volume data set as described in Section 1.1. To show interventional ultrasound in the context of organ vasculature, an early arterial phase CTA scan from a patient's liver is used as volume data set. The injected contrast agent causes the vessel structure to show up with high intensities in the CTA scan. This makes it easier to distinguish the vascular structure from surrounding tissue.

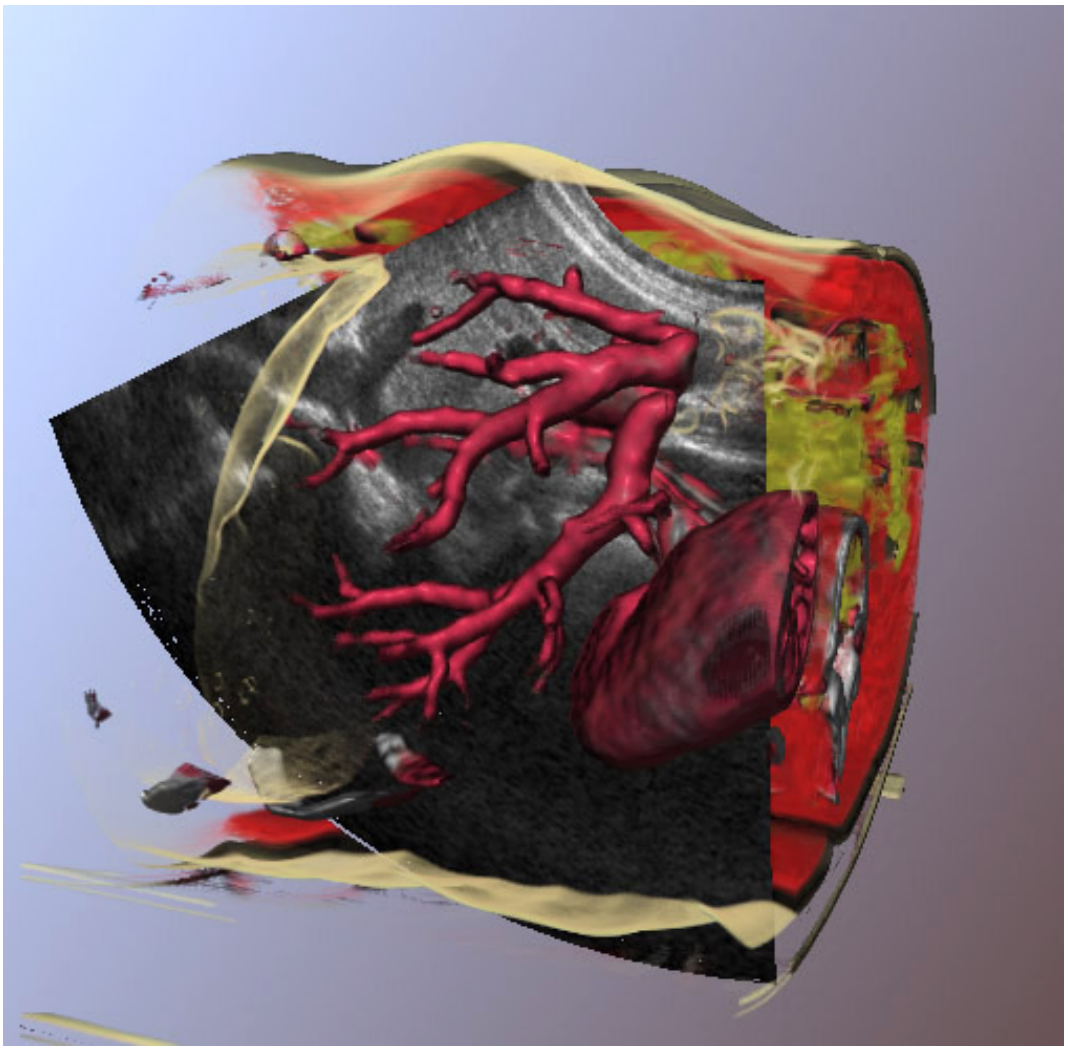


Figure 5.2: Ultrasound image of a liver in the context of a CTA scan. Small overlay thickness, Phong shading.

Figure 5.2 depicts the proposed rendering for a longitudinal ultrasound image of the liver and the corresponding CTA data. In front of the ultrasound plane only the vessels and parts of the skin are rendered. The importance value of the vessels is 1 while the importance value of the skin is 0.3. This is the reason why the skin is rendered more transparently and with less details. All other tissues have an importance value lower than the τ_l threshold, so they are not visible in the overlay area. The thickness d is set to a small value to just show the vessels close to the ultrasound plane.

Figure 5.3 shows a similar rendering with Gooch shading instead of Phong shading. The thickness is set to a higher value to have a larger overlay area. Also parts of the ribs are visible in front of the ultrasound plane because these parts are in the same component as

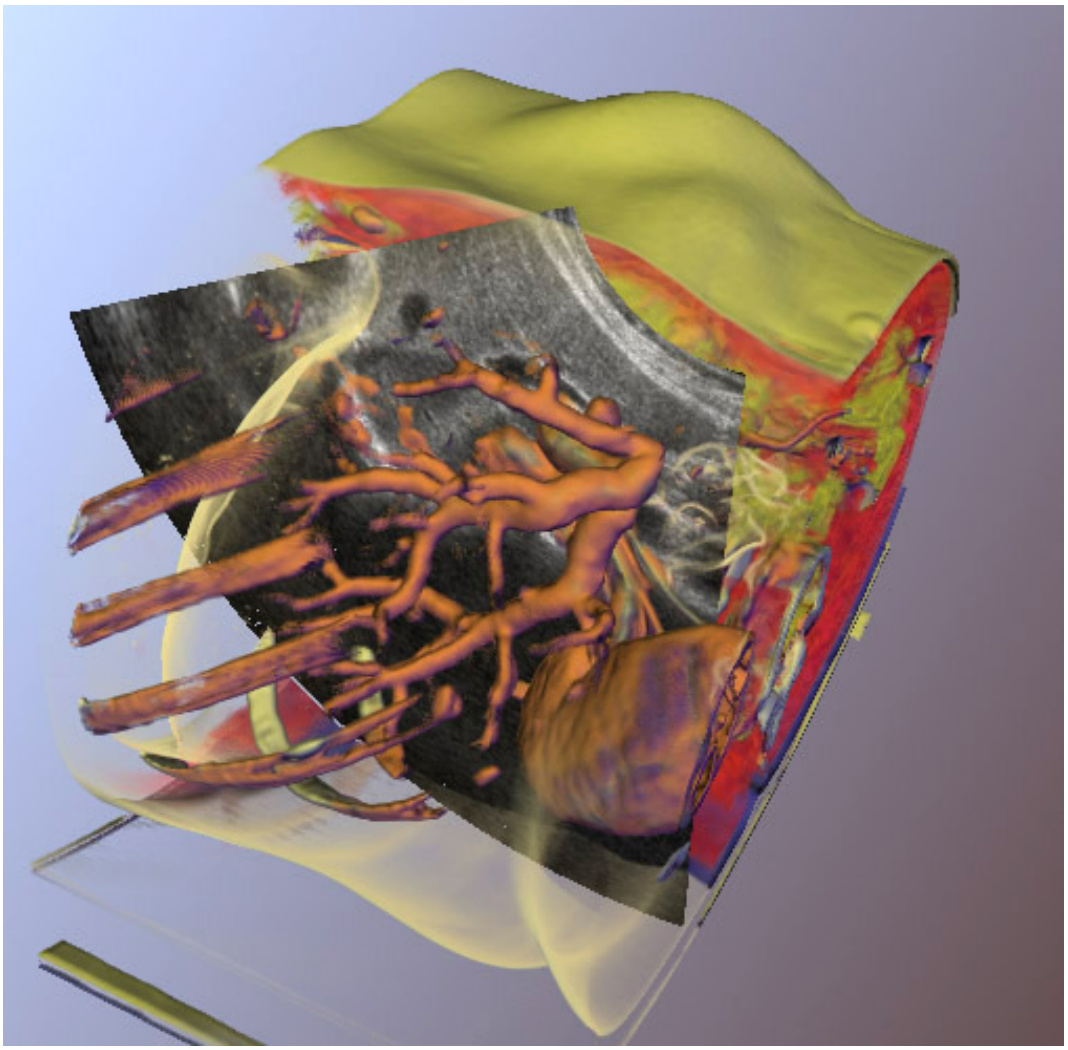
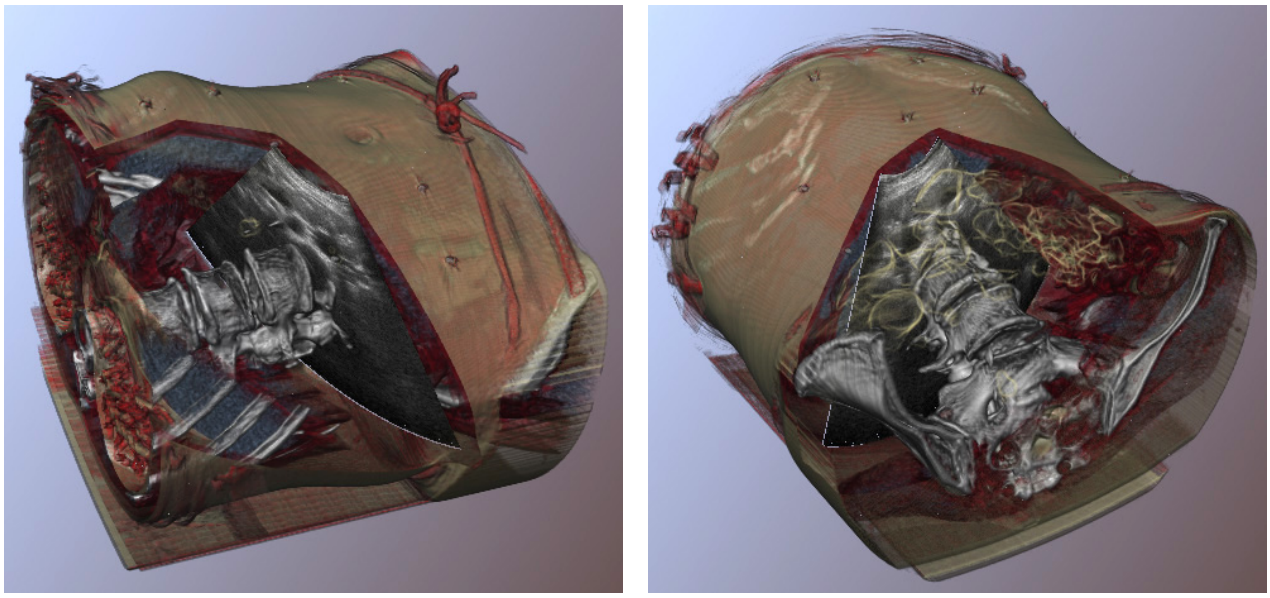


Figure 5.3: Ultrasound image of a liver in the context of a CTA scan. Large overlay thickness, Gooch shading.

the vessel structure in the transfer function space. The Gooch cool-to-warm shading has the effect that parts which are pointing towards the light source are shaded with a warmer color such as yellow than parts which are pointing in the opposite direction. For these parts a cooler color such as blue is used. These warm and cool colors should simulate the effect of the sun in nature where surfaces which are pointing to the sun are warmer than other surfaces. It should be easier with this shading method to see the surface orientation in the rendering result.

For the liver intervention typically a CTA scan is done to highlight the important vessel structure inside the liver. But it is also possible that only a CT scan is done. In such a case the physician has to find the vessels only in the ultrasound image. Nevertheless the proposed



(a) Small overlay area, $\Theta_1 = \Theta_2$

(b) Large overlay area, $\Theta_1 = \Theta_2$

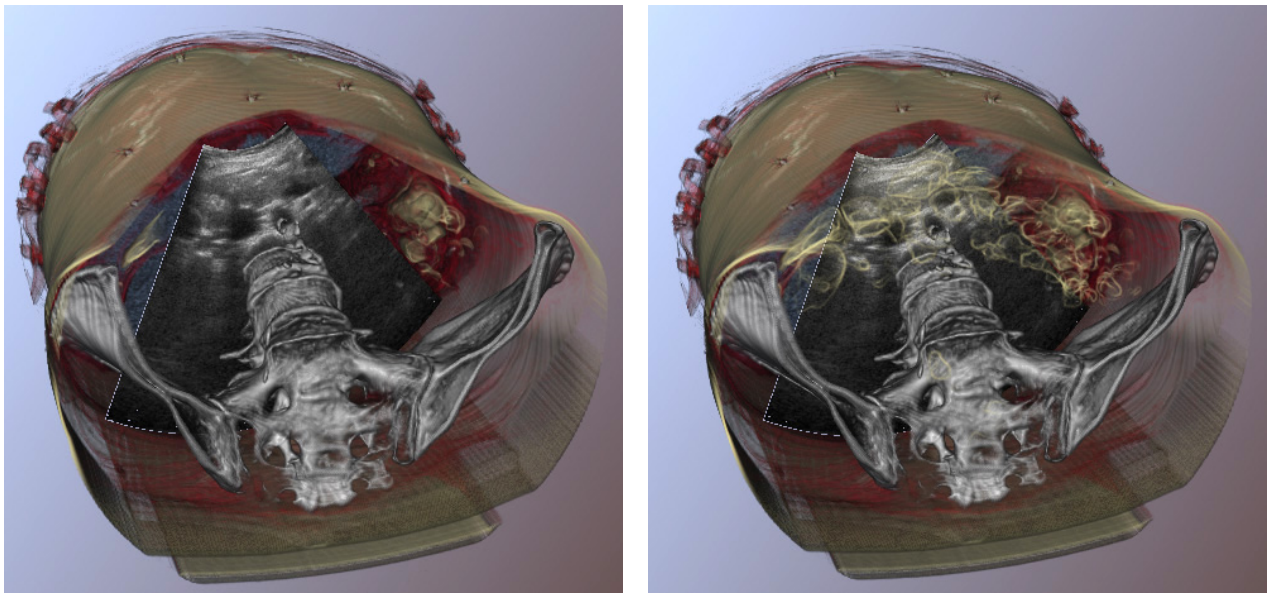
Figure 5.4: Rendering results with a volume data set from a CT scan. Figure (a) uses a smaller overlay area than figure (b) to cut away the ribs which would occlude the ultrasound plane.

methods can be used here as well because a physician gets a good impression of the position of the ultrasound plane in relation to the body. With the magnetic tracking and the automatic registration this technique is a very useful tool to explore the body by using the ultrasound probe as interaction device.

In Figure 5.4 two result images are shown where a CT was used as volumetric data. In the CT the bones are considered as most important parts. In both Figures 5.4(a) and 5.4(b) the angles Θ_1 and Θ_2 are set to the same value. This results in a sharp edge between the base area and the overlay area because the transition area is not present. In Figure 5.4(a) the overlay area has a small thickness. Therefore the ribs are clipped away and do not occlude the ultrasound plane.

Figure 5.5 shows two more results with a CT data set. These images show the effect of the σ exponent in the formula of the lower threshold τ_l. This threshold is used in Formula 3.14 to modify the importance. With a higher σ value more parts are visible in the overlay area. In Figure 5.5(b) the σ value is higher than in Figure 5.5(a). Therefore in Figure 5.5(b) more parts are shown in front of the ultrasound plane.

Besides CTA or CT also MRI can be used as pre-operative imaging system. Figure 5.6 shows two results with an MRI data set. In Figure 5.6(a) there is no overlay area. So nothing



(a) $\Theta_1 < \Theta_2$, $\sigma = 1$ (b) $\Theta_1 < \Theta_2$, $\sigma = 4$

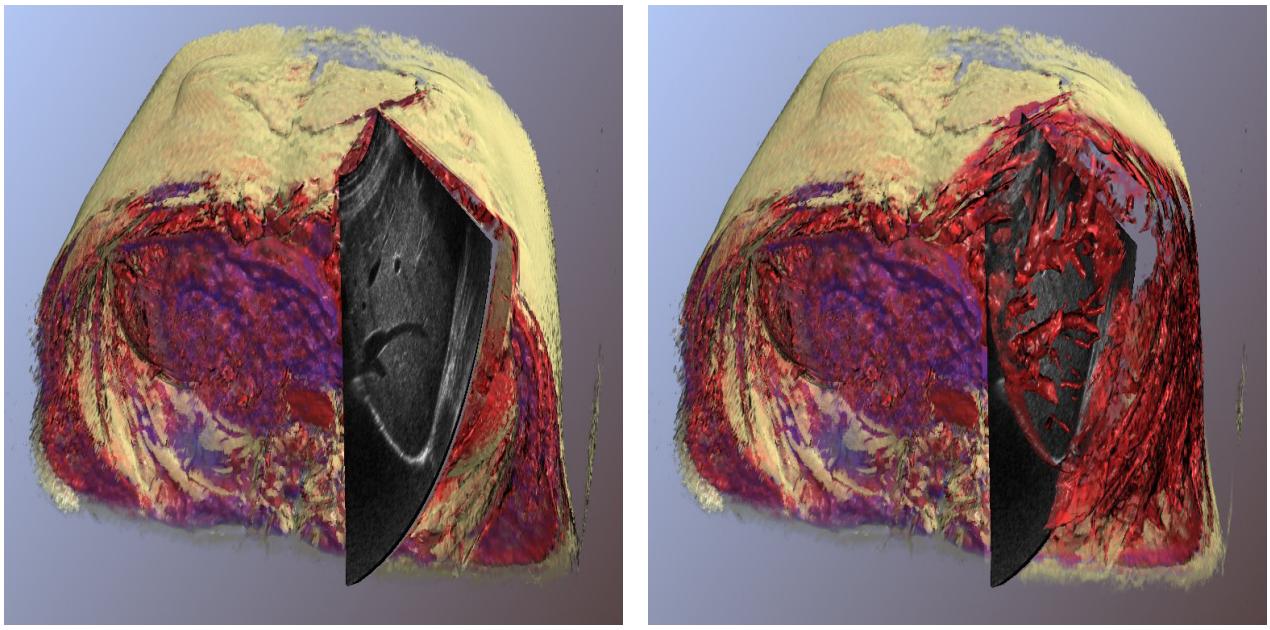
Figure 5.5: The two images show the effect of the σ exponent.

is shown in front of the ultrasound plane. This is useful when the physician is just interested in the ultrasound plane. In Figure 5.6(b) a small overlay area is shown. As one can see it is also possible with the introduced methods to highlight the vessel structure in an MRI data set.

As mentioned at the beginning of this chapter normally ultrasound is used as interventional imaging system during a liver intervention. In some cases the field of activity is too deep inside the body and cannot be recorded by ultrasound. Therefore other interventional imaging systems such as interventional MRI have to be used. However, for the proposed techniques in this thesis all of these imaging systems can be used in the same way as ultrasound.

In all data sets, the ultrasound images are made clearly visible and their spatial context within the 3D volume is well defined due to the view dependent cutaway structure. In addition to the global 3D relation, critical anatomical structures such as liver vasculature that must not be punctured in an interventional scenario can be visualized as well. It is also straightforward to integrate planning information, such as target volumes and needle path, using the visualization tools proposed in this thesis.

Finally, the presented techniques have been designed for and tested with a hardware-based GPU raycaster as described in the previous chapter. As such, the system can operate at interactive frame rates, which is necessary for the use in a clinical scenario. With an NVIDIA GeForce 6800 GT (see http://www.nvidia.com) frame rates up to 10 frames per



(a) No overlay area, $\Theta_1 = \Theta_2$

(b) Small overlay area, $\Theta_1 = \Theta_2$

Figure 5.6: The volumetric data results here from an MRI scan.

second were achieved for a volumetric data set with a size of 256×256×256. With a graphics card of a newer generation such as the NVIDIA GeForce 8800 GTX the frame rate will be even higher. There will be no lag between the interaction with the intervention imaging system and the rendering of the merged visualization which is very important for the safety of the intervention.

Chapter 6

Summary

I may not have gone where I intended to go, but I think I have ended up where I needed to be.

Douglas Adams

In this final chapter a short summary is given as well as an outlook in the future how the proposed techniques can be used in practice and what can be improved. The goal as defined in the introduction was to find a merged visualization of two input sources where the most important parts are visible. The results in the previous chapter show that this goal can be achieved with the proposed techniques.

To get such a visualization it is necessary at the beginning to know which parts are important for the intervention. This is highly related to the kind of medical application. In this thesis the main focus was on liver intervention. Thereby the object of interest (e.g. a tumor) and the vessel structure are the most important parts in the volume. Nevertheless it is also possible to use these techniques for other medical applications. The only component which has to be changed to highlight other parts is the importance value.

When the importance is assigned to all voxels in the volume then the contextual cutaway view, starting at the region of interest, can be generated. With all the parameters for the size of the different areas in this cutaway view it can be easily adapted to the needs of the physician. For example the thickness of the overlay area can be reduced if there are too many parts of the volume occluding the region of interest.

By modifying the visualization of each voxel in the volume according to its position in relation to the cutaway view it is possible to generate the final merged visualization. The different areas in the contextual cutaway view influence the visualization in different ways. The formulas for this modification have also some parameters which can be used to adjust the final appearance.

With this visualization it is possible to show the pre-operative data directly during the intervention together with the image coming from the interventional imaging system. In contrast to the conventional technique, where only the image from the interventional imaging system was used during the intervention, this new technique provides more information and, therefore, the intervention should be safer.

Another benefit of the merged visualization is a faster surgery planning. In the traditional way the pre-operative data is used to plan the surgery. The path for a needle, e.g., was planned with this data. During the intervention the physician tries to follow this path by using the information provided by the interventional imaging system. With the new approach the planning of the surgery can be done directly during the intervention because the physician sees all important parts of the pre-operative data.

Furthermore the integration of the region of interest in the pre-operative volume allows to explore the pre-operative data by changing the position of the region of interest. When ultrasound is used as in the liver intervention then the ultrasound probe can be used as input device for the navigation through the pre-operative data. Due to the fact that the position of the ultrasound probe on the patient's body matches with the position in the pre-operative volume the physician has a direct link between the visualization and the patient.

Besides the generation of a good visualization which shows the most important parts of both input sources, the usability of the new techniques in practice was an important design criterion. All processing steps which are needed to get the final result can be applied on the input sources without any user interaction. This is important because physicians do not want to spend time on this. For this reason all parameters and other settings have to be set only once for a certain medical task with the use of input sources from certain imaging systems. After this first setup the visualization for all further data sets is generated automatically without any user interaction.

Nevertheless there are also some parts which can be improved in the proposed approach. The following section will take a closer look on these improvements.

6.1 Further Work

The proposed approach delivers an expressive visualization which shows all necessary parts for the intervention. Nevertheless the physician also needs other visual feedbacks beside the merged visualization. In the liver intervention, e.g., the ultrasound plane is additionally visualized in the traditional way in a separate window because physicians are used to this view.

As described in Chapter 3 the viewpoint of the observer is needed to generate the contextual cutaway view. This viewpoint should be at the same position as the physician's position in relation to the patient. The viewpoint position can easily be changed in the application but the problem is that the physician has no free hand during the intervention to handle the application. Therefore it would be good if the physician is tracked by magnetic tracking the same as the ultrasound probe to get his position. Furthermore this information can be directly used to change the viewpoint in the application according to the position of the physician. This additional tracking is only necessary if the physician moves during the intervention. For the liver intervention the position of the physician is more or less fixed. In such a case the viewpoint can be fixed in the application and no tracking of the physician is needed.

For the guidance of the intervention tool such as a needle in the liver intervention application some other visualizations are possible. For example the viewpoint can be set to the top of the intervention tool. With this view it would be easier to navigate the intervention tool to the destination inside the body.

As shown in the results chapter the new approach for assigning importance values works very well for CTA and MRI data. Especially for CTA it is easy to emphasize the vessels with this new method because they are highlighted by the contrast agent. In other cases such as for CT or other tissues than the vessels it is not always possible to assign an importance value in the transfer function space. For such cases other techniques such as volume segmentation would be necessary. The only crucial aspect for this importance assignment is that it should work without any user interaction.

Another part which has to be improved is the registration. A perfect registration should deliver the position of the region of interest inside the pre-operative data by considering all movements of the body such as breathing. The registration of the ultrasound plane and the CTA in the liver intervention is done only once before the intervention. Therefore these movements are not considered during the intervention. To integrate this movements a solution would be the measurement of the belly movement. This information can then be used to deform the pre-operative data to get a more accurate merged visualization.

So far the region of interest was always considered as one of the most important parts in the visualization. For this reason the region of interest was always rendered completely opaque. It is also possible to render this region semi-transparently to see the pre-operative data through it. In the most extreme case the region of interest can be completely transparent and is just used for the navigation through the volume. Besides modifying the transparency of the region of interest there are plenty of other methods which can be used to merge the region of interest with the pre-operative data.

Another possibility for an improvement is the use of augmented reality. With this technique it would be possible to overlay the merged visualization directly on the patient. The physician can see this visualization through a head-mounted display. This would be the most intuitive visualization of all but it also has some drawbacks such as the need of wearing a head-mounted display during the intervention.

As mentioned in this section, there are several possibilities to improve the proposed approach in this thesis. Nevertheless the new approach works well in the liver intervention application and it would be possible to adapt it for other applications with only small changes. There is open space for further research to extend this technique and find novel approaches for multimodal volume rendering.

Bibliography

[1] D. Agrafiotis, M. G. Jones, S. Nikolov, M. Halliwell, D. Bull, and N. Canagarajah. Virtual liver biopsy: image processing and 3D visualization. In *Proceedings 2001 International Conference on Image Processing*, volume 2, pages 331–334, 2001.

[2] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

[3] S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *VIS '05: Proceedings of the 16th IEEE Visualization 2005*, pages 671–678, 2005. http://www.volumeshop.org/.

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

[5] M. Burns, M. Haidacher, W. Wein, I. Viola, and M. E. Gröller. Feature emphasis and contextual cutaways for multimodal medical visualization. In *EuroVis '07: Proceedings of EUROGRAPHICS/IEEE VGTC Symposium on Visualization 2007*, pages 275–282, 2007.

[6] C. Correa, D. Silver, and M. Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1069–1076, 2006.

[7] T. J. Cullip and U. Neumann. Accelerating volume reconstruction with 3D texture hardware. Technical report, University of North Carolina at Chapel Hill, 1994.

[8] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation. In *Proceedings of VisSym '03*, pages 239–248, 2003.

[9] A. B. Ekoule, F. C. Peyrin, and C. L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, 10(2):182–199, 1991.

[10] T. T. Elvins. A survey of algorithms for volume visualization. *ACM SIGGRAPH Computer Graphics*, 26(3):194–201, 1992.

[11] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. A K Peters, Ltd., 2006.

[12] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, 2001.

[13] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[14] H. Fuchs, M. Levoy, and S. M. Pizer. Interactive visualization of 3D medical data. *Computer*, 22(8):46–51, 1989.

[15] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, 1986.

[16] G. Glombitza, W. Lamade, A. M. Demiris, M. R. Gopfert, A. Mayer, M. L. Bahner, H. P. Meinzer, G. Richter, T. Lehnert, and C. Herfarth. Virtual planning of liver resections: image processing, visualization and volumetric evaluation. *International Journal of Medical Informatics*, (53):225–237, 1999.

[17] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452, 1998.

[18] H. Hauser and M. Mlejnek. Interactive volume visualization of complex flow semantics. In *Proceedings of VMV '03*, pages 191–198, 2003.

[19] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[20] G. T. Herman and H. K. Liu. Three-dimensional display of human organs from computed tomograms. *Computer Graphics and Image Processing*, 9:1–21, 1979.

[21] G. N. Hounsfield. Computerized transverse axial scanning (tomography). 1. description of system. *Br J Radiol*, 46:1016–1022, 1973.

[22] A. E. Kaufman. Accelerated volume graphics. In *GMP '02: Proceedings of the Geometric Modeling and Processing - Theory and Applications*, pages 3–10, 2002.

[23] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19:2–11, 1975.

[24] C. Kirbas and F. Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys*, 36(2):81–121, 2004.

[25] J. M. Kniss, G. Kindlmann, and C. D. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *VIS '01: Proceedings of the 12th IEEE Visualization 2001*, pages 255–262, 2001.

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

[27] A. König, H. Doleisch, and M. E. Gröller. Multiple views and magic mirrors - fMRI visualization of the human brain. In *Proceedings of SCCG '99*, pages 130–139, 1999.

[28] A. Krüger, C. Tietjen, J. Hintze, B. Preim, I. Hertel, and G. Strauß. Interactive visualization for neck dissection planning. In *EuroVis '05: Proceedings of EUROGRAPHIC-S/IEEE VGTC Symposium on Visualization 2005*, pages 295–302, 2005.

[29] J. Krüger, J. Schneider, and R. Westermann. Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):941–948, 2006.

[30] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003*, pages 38–43, 2003.

[31] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458, 1994.

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

[33] Wei Li, K. Mueller, and A. Kaufman. Empty space skipping and occlusion clipping for texture-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003*, pages 42–50, 2003.

[34] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, 1987.

[35] R. Mullick, R. N. Bryan, and J. Butman. Confocal volume rendering: fast, segmentation-free visualization of internal structures. In *Medical Imaging 2000: Image Display and Visualization*, pages 70–76, 2000.

[36] R. A. Novelline. *Squire's Fundamentals of Radiology.* Harvard University Press, sixth edition, 2004.

[37] L. Ren, H. Pfister, and M. Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum (EUROGRAPHICS 2002)*, volume 21, pages 461–470, 2002.

[38] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume rendering on standard PC graphics hardware using multi-textures and multi-stage rasterization. In *HWWS '00: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 109–118, 2000.

[39] C. Rezk-Salama and A. Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum (EUROGRAPHICS 2006)*, 25(3):597–606, 2006.

[40] P. Rheingans and D. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.

[41] T. Ropinski, F. Steinicke, and K. Hinrichs. Visual exploration of seismic volume datasets. In *Proceedings of the WSCG '06*, pages 73–80, 2006.

[42] M. Straka, M. Červeňanský, A. La Cruz, A. Köchl, M. Šrámek, M. E. Gröller, and D. Fleischmann. The VesselGlyph: Focus & context visualization in CT-angiography. In *VIS '04: Proceedings of the 15th IEEE Visualization 2004*, pages 385–392, 2004.

[43] I. Viola and M. E. Gröller. Smart visibility in visualization. In *Proceedings of Computational Aesthetics in Graphics, Visualization and Imaging*, pages 209–216, 2005.

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

[45] W. Wein, B. Röper, and N. Navab. Automatic registration and fusion of ultrasound with CT for radiotherapy. In *Proceedings of MICCAI '05*, pages 303–311, 2005.

[46] J. Wernecke and Open Inventor Architecture Group. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Addison-Wesley Professional, 1994.

[47] J. Wernecke and Open Inventor Architecture Group. *The Inventor Toolmaker: Extending Open Inventor, Release 2.* Addison-Wesley Professional, 1994.

[48] L. Westover. Footprint evaluation for volume rendering. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 367–376, 1990.

[49] Yin Wu, V. Bhatia, H. Lauer, and L. Seiler. Shear-image order ray casting volume rendering. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 152–162, 2003.

[50] J. Zhou, A. Döring, and K. D. Tönnies. Distance transfer function based rendering. Technical report, University of Magdeburg, Germany, 2004.

[51] J. Zhou, M. Hinz, and K. D. Tönnies. Focal region-guided feature-based volume rendering. *IEEE First International Symposium on 3D Data Processing Visualization and Transmission*, pages 87–90, 2002.