DIPLOMARBEIT

Advanced Visualization Techniques for Vessel Investigation

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

unter der Anleitung von Ao.Univ.Prof. Dipl.-Ing. Dr.techn Eduard Gröller

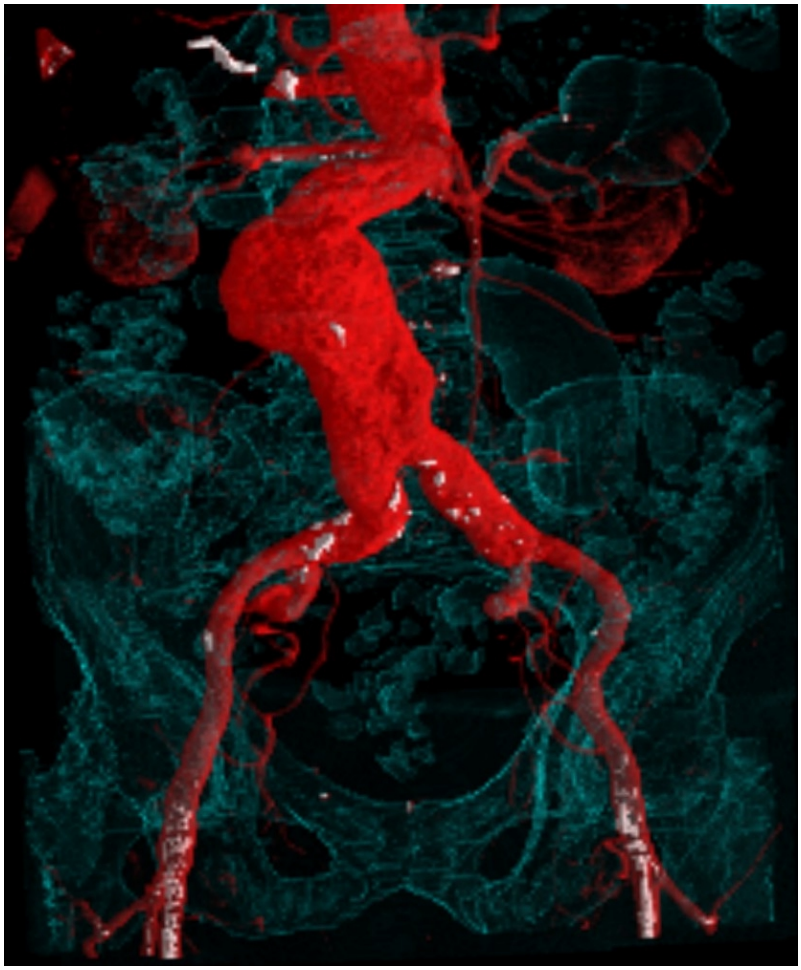
durch

Armin Markus Kanitsar
Matrikelnummer.: 9625009
Mühlbreitenstraße 24, 8280 Fürstenfeld

Wien, 13. Februar 2001 Armin Kanitsar

Advanced Visualization Techniques for Vessel Investigation

Armin Markus Kanitsar



Februar 13, 2001

Overview

Abstract

Different approaches of visualization techniques and segmentation methods for computed tomograpy angiography datasets are investigated. The particular characteristics of this data are addressed. A global path-optimisation method for a reliability-enhanced vessel-tracking method was introduced. Furthermore an interactive segmentation technique focusing on the clinical use is proposed.

Kurzfassung

Verschiedene Ansätze von Visualisierungstechniken und Segmentierungs-methoden für Datensätze aus der Computertomograhie-Angiographie werden untersucht. Auf die besonderen Eigenschaften dieser Daten wird näher eingegangen. Eine Methode zur globalen Optimierung eines Pfades für zuverlässige Gefäßverfolgung wird vorgestellt. Weiters wird eine interaktive Segmentierungstechnik mit Fokus auf den klinischen Einsatz vorgeschlagen.

Contents

Chapter 1

Introduction

Medical image analysis is a rapidly growing field in information technology. Especially in the field of computed tomography new areas of application arise due to the progress in computed tomography modalities. As higher resolutions in the spatial domain can be achieved, further research in new investigation techniques has to be done. The methods used for these techniques depend very much on the application, which they are focused on. One of such applications is computed tomography angiography (CTA) of peripheral vascular diseases.

Lower extremity arterial disease is a significant health problem in the industrial world. The prevalence of symptomatic disease (intermittent claudication) in patients between 55 and 74 years of age is 4.6% [24]. For planning of surgical or transluminal revascularization procedures of patients with sever claudication or limb threatening ischemia, the entire inflow and runoff vessels of the lower extremities need to be visualized. Today, intraarterial digital subtraction angiography (iaDSA) is the pretherapeutic imaging technique of choice. iaDSA, however, is an invasive and costly procedure, which requires arterial catheterisation. A non-invasive technique for imaging the entire inflow and runoff vessels is therefore highly desirable.

Computed tomography angiography (CTA) of the arteries using standard (single-slice) helical CT has been performed previously and has shown high accuracy in the detection of vascular occlusions and in the grading of high-grade stenoses. Latest technical developments in CT – notably multi-slice helical CT –

allow an approximately three-fold increase of volume coverage, while maintaining longitudinal resolution, and without incurring more helical artifacts. Multi-slice helical CT thus has the potential to accurately show the entirety of the lower extremity vessels with a single intravenous contrast-medium injection at unprecedented, near isotropic spatial resolution.

In order to visualize the entire arterial system of the lower-limb vessels with CTA, a stack of approximately 900 to 1500 transversal images have to be reconstructed per patient. It is obvious, that such datasets cannot be reviewed by a radiologist on the basis of single images to diagnose the presence, degree, and location of obstructive vascular diseases. Image post-processing is thus an indispensable prerequisite for using multi-slice CTA as a non-invasive imaging tool.

Currently available editing and rendering techniques require more than 4 hours of manual editing and interaction of a well-trained user (radiologist or technologist) to obtain images that provide any diagnostical value. Compared to the acquisition time of 25 to 60 seconds it is obvious that post-processing time has to be reduced.

Segmentation of medical data is a very difficult task. A lot of research has been done up to now by many groups of scientists around the world. However, no fully automatic segmentation technique for medical image analysis is known yet which produces correct results in all possible cases. In the following two different investigation methods are presented which provide semi-automatic investigation for arteries of the lower limbs. The first one proposes a new technique for the generation of curved planar reformations. The second approach was developed as a combination of well-established techniques.

In chapter 2 the acquisition and the main properties of the used datasets are discussed. Furthermore some cases of the spatial relationship between vessel and bony structures are described in order to give a better insight into the problematic properties of CTA datasets.

Chapter 3 gives a rough overview of the most common visualization techniques in medical image processing. The focal point is set on techniques that are used later on.

The segmentation techniques described in chapter 4 are the current state of the art in medical image processing. Furthermore these techniques are put into relation with CTA datasets of lower extremities.
In chapter 5 the two proposed algorithms are described in detail. The theoretical background as well as the similarities and differences to existing algorithms are described.
Chapter 6 gives a short insight into implementation details and complexity estimations. In chapter 7 the results of the used algorithms are presented. A short discussion and conclusions are stated in chapter 8. In chapter 9 a short version of the thesis is added, pointing out the most important topics. Chapter 10 is an acknowledgment to all persons who supported the author of this thesis.

See where you are,
Define, where you want to be,
Make your plan, and
Go!

Robert Johnson

Chapter 2

Characteristics of the patients' datasets

This chapter describes the properties of the datasets in more detail. A short section deals with the acquisition methods used in clinical environments. The most relevant arterial diseases targeted by CTA are described. Furthermore the properties and the spatial relationships between bones and vessels within the datasets are discussed.

2.1 Data acquisition

Computed tomography angiography (CTA) is a true volumetric imaging modality for non-invasive imaging of the human arterial system. The principle of CTA is to rapidly acquire a series (stack) of cross-sectional (transversal) CT images through the human body while at the same time opacifying the arterial system with a radiographic contrast-medium. The contrast-medium is administered intravenously, synchronized with the acquisition time, which is in the range of 25 to 60 seconds.

The latest technical development in computed tomography (CT) is multi-slice CT. In contrast to standard single-slice CT, this technique simultaneously acquires 4

slices per revolution of the x-ray tube around the patient (see figure 2.1). Together with faster acquisition times, this results in a higher spatial resolution and increased volume coverage in the longitudinal or *z*-direction and thus less partial volume effects. Only with multi-slice CT technology it has become possible to perform a CT angiography of the entire arterial tree of the lower extremities.

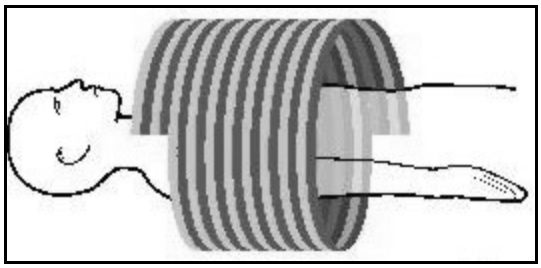


Figure 2.1: The multi-slice CT method

2.2 CTA datasets

CTA datasets consist of a stack of transversal (*xy*-plane) cross-sectional images. Each transversal image contains 512 squared sampling points (*pixels*), representing the radiographic density. These transverse images are also referred to as slices. Depending on the size of the anatomic region of interest, a CTA dataset may consist of up to 1500 cross-sectional images, stacked in the longitudinal or z-direction. Datasets can be characterized by an inherent spatial domain consisting of single scalar values. Each of the sampling points in the three-dimensional datasets – also referred to as *voxels* (for volume elements) has a depth of 16 bits. A single three-dimensional dataset may become as large as 750 MB.

The attenuation (or density) values in the datasets are given in Hounsfield units (HU). Per definition, water has an attenuation value of 0 HU; the attenuation value of air is –1000.

2.3 Properties of arteries

The arterial tree of interest, which supplies blood to the legs, includes the abdominal aorta, the pelvic arteries and the arteries of both legs. The entire vascular tree including the abdominal aorta is shown in figure 2.2. Usually the attenuation values of opacified blood within the vessels varies from 70 to 480 HU. This is because of inhomogenous vascular opacification and because of changes of the luminal diameter. The diameter of blood vessels ranges from 45 voxels around the aorta down to 1 voxel at the bottom end of the vessel tree.
The most relevant arterial abnormalities are:

- *Stenosis*: A stenosis is a narrowing of the arterial flow lumen. Arterial stenoses are caused by atherosclerotic *plaque* (as can be seen in figure 2.2*a*). Atherosclerotic plaques are soft tissue density lesions within the vessel wall, but plaques may also calcify. Atherosclerotic plaques often occur near the areas of high turbulences, such as bifurcations and bends. A *bifurcation* is for example the area where the aorta branches into the iliac (pelvic) arteries.
- *Occlusion*: A complete obstruction of a vessel is referred to as an occlusion. The blood flow is redirected through secondary vessels, which circumvent the occluded vascular segment, and which are called collateral vessels. An example of an arterial occlusion is shown in figure 2.2*c*.
- *Calcification*: The vessel wall of diseased arteries, as well as atherosclerotic plaque may calcify. With CT, calcified tissue is of high attenuation. In figure 2.2 several areas of calcification can be seen. One is marked with circle *b*.

Figure 2.2: The vessel tree of the lower extremities. *a*) a short, segmental stenosis in the femoral artery. *b*) calcification near the aortic bifurcation. *c*) a long occlusion in the femoro-popliteal artery.

2.4 Properties of bones

In contrast to magnetic resonance angiography (MRA) bones are visible in CTA due to their high density. Usually bones have a non-monotonic density as shown in figure 2.3. At the level of the knee the density is lower than elsewhere, e.g. in

the shinbone. It is clearly visible in figure 2.3 that the bone is not solid as one might expect. Inside the bone the density is much lower because of the bone marrow. The density values therefore range from 70 to 1876 HU.

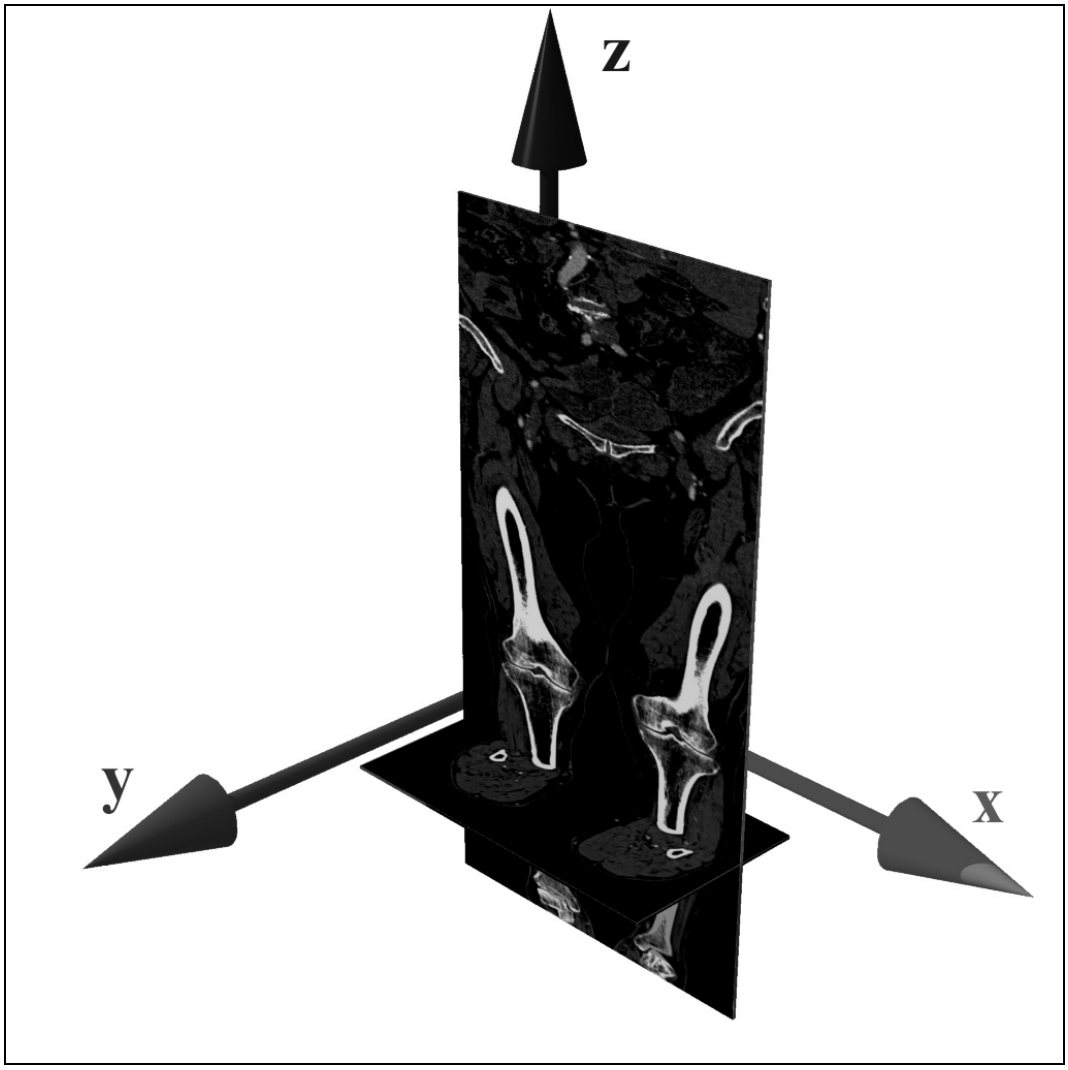


Figure 2.3: Two cutting planes of a CTA-Dataset.

2.5 Range of density values

As mentioned above the density of the four different tissues of interest overlap significantly. The four different categories are the bones, the blood opacified with

contrast agent, calcifications and the remaining soft tissue. A comparison can be seen in table 2.1.

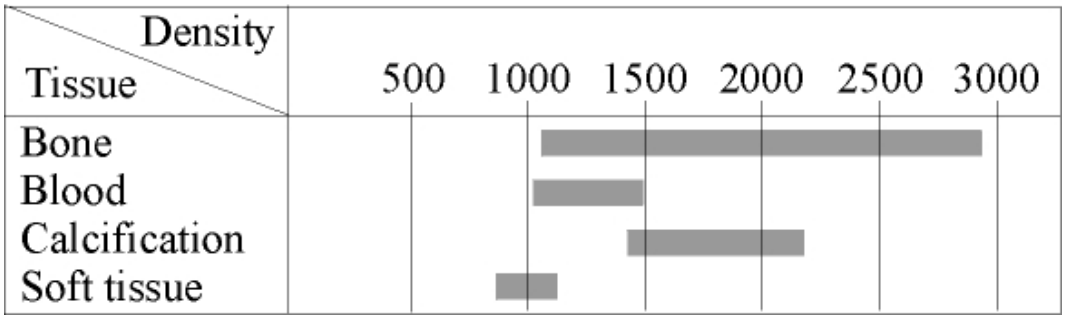


Table 2.1: Comparison of the density intervals of the four different tissues in absolute data values. (i.e., a density-data value acquired by the CTA procedure of 1500 corresponds to 500 Hounsfield units (HU))

2.6 Spatial arrangement of different objects

Apart from the range of the density values the spatial arrangement of different objects is a critical point. Especially for segmentation algorithms, which will be described below, this is a key factor to their success. During the research and evaluation process some particularily noteworthy spatial arrangements were discovered.

2.6.1 Spatial vicinity of bone and vessel

Especially in the lower (calf) portion of the vessel tree, arteries frequently come very close to bones. This situation can be seen in figure 2.4. In the center of the left image the femur (thigh bone) with an artery to it's right border can be seen. The same picture contains an artery with calcification in the right lower part of the figure. On the right side the corresponding height field of density values is shown. It is difficult to distinguish the artery near the bone from the surrounding soft tissue.

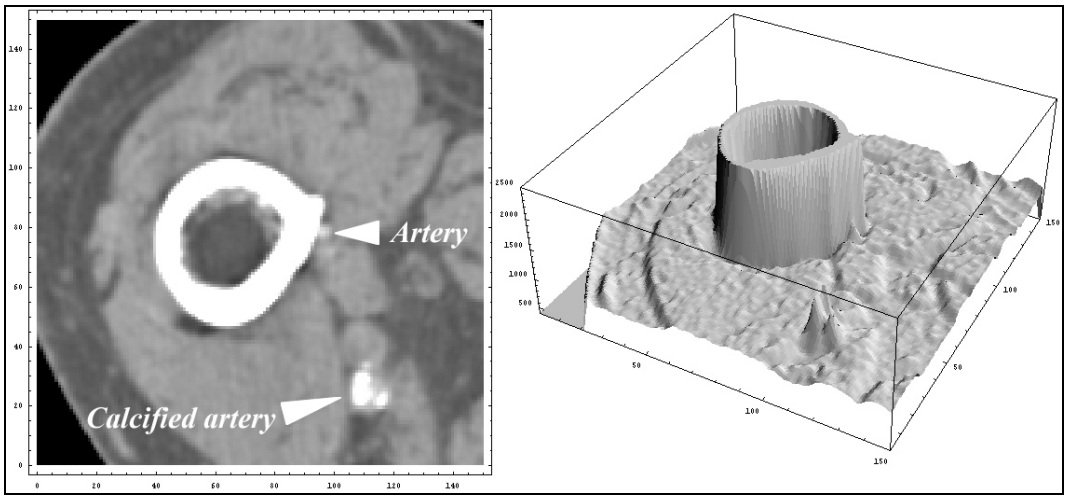


Figure 2.4: Left: Grey scale image of a 150 x 150 cutout of a slice. Right: Height field of the same region based on the density value. The range is from 0 to 2500 HU.

2.6.2 Spatial vicinity of bone and calcification

It may occur that calcified vessel walls and bony structures come together very closely. In this case aliasing and partial volume effects might merge these two objects. This situation can be seen in figure 2.5.

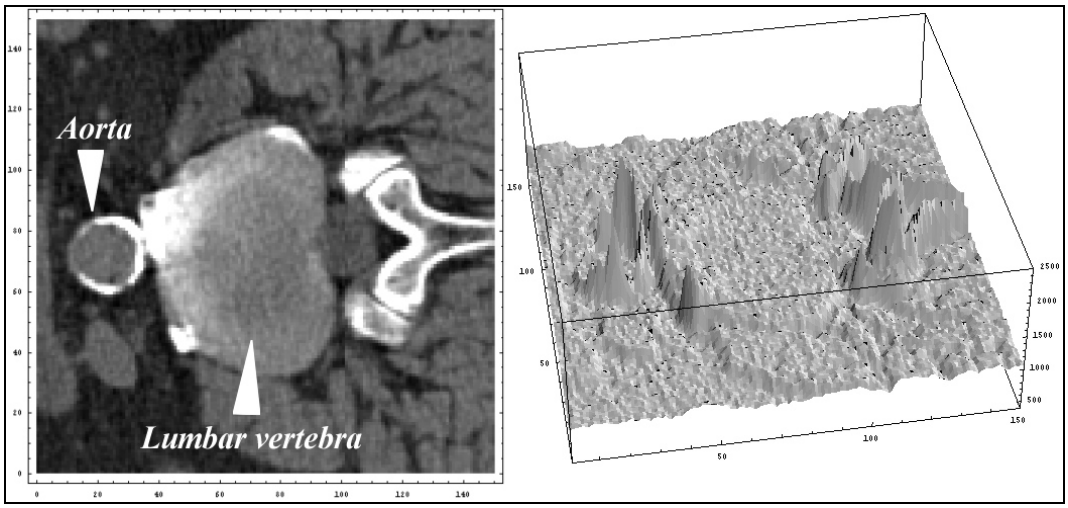


Figure 2.5: Cut out of the aorta with circular calcification of the vessel wall in close vicinity to a lumbar vertebra. Note that no contrast agent was applied in this case.

2.6.3 Fuzzy border

As mentioned in section 2.4 bones are not entirely bordered by high-density values. In some cases arteries may come close to regions where the bones are less dense. Hence, no exact separation between bones and arteries based on their attenuation values can be done in this situation (as can be seen in figure 2.6).

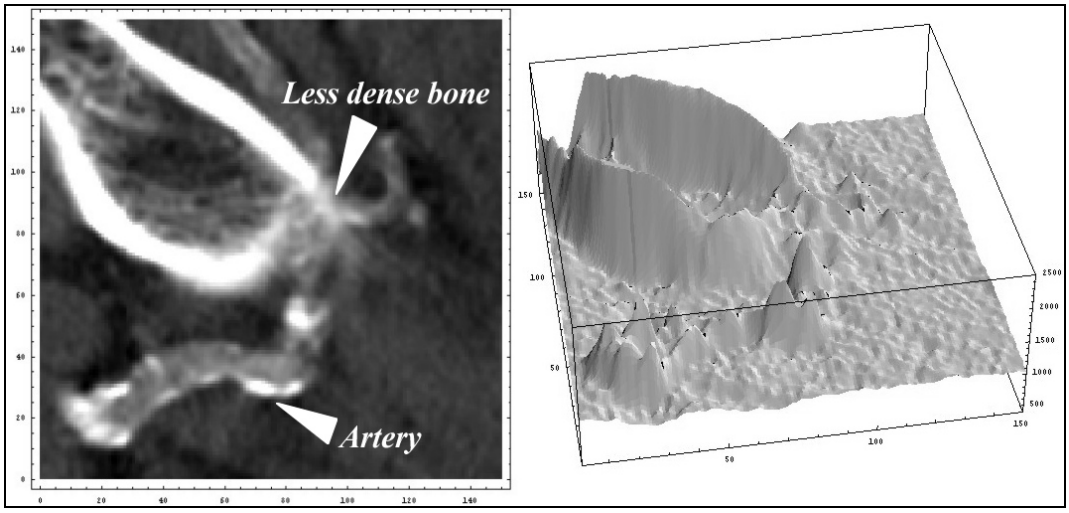


Figure 2.6: A part of a CTA-dataset in the area of the pelvic bone. The tight spatial relationship of a bony structure with low density and an contrast enhanced artery is clearly visible.

2.7 Conclusions

Multi-slice CTA provides high-resolution three-dimensional datasets of the lower extremities and their arterial tree. The first challenge in an attempt to automatically extract the arterial system in the datasets is that the density values of different tissues overlap. The second difficulty is the tight spatial relationship of different objects in some anatomic regions. Another inherent problem with lower-extremity CTA-datasets is their large size. For these reasons algorithms applied to these datasets have to be checked carefully concerning their accuracy and efficiency in order to retrieve satisfying results in an acceptable time.

Chapter 3

Visualization techniques

Visualization is a rather wide-spread field of research. There are a lot of different areas, for instance flow visualization, information visualization, etc. This chapter will focus on volume-visualization techniques, which are applicable to CTA-datasets.

3.1 Slicing

Slicing is a very simple visualization method. However it provides a way to precisely navigate through the whole dataset. This feature is provided by almost every visualization systems dealing with CT data.

The basic concept is to resample the volume data on three planes perpendicular to the x-, y- and z-axes (transversal aka axial, coronal, and saggital). It is up to the user to change the position of the planes within the dataset (see figure 2.3). In this way it is possible to investigate the whole dataset by viewing the different cutting-planes. The data values are usually displayed using grayscale images. In common commercial visualization systems the technique of slicing is known as Multiplanar Reconstruction (MPR).

It is very often desirable to highlight different properties within the dataset by focusing on a subset of data values. This method is called *windowing* and is

defined as a function that can be seen in figure 3.1. The *window center* and *window width* define an interval of data values. Every data value below the lower boundary of the interval is displayed as black. Data values above the upper boundary are displayed as white. Within the windowing interval the color is interpolated according to the data value.

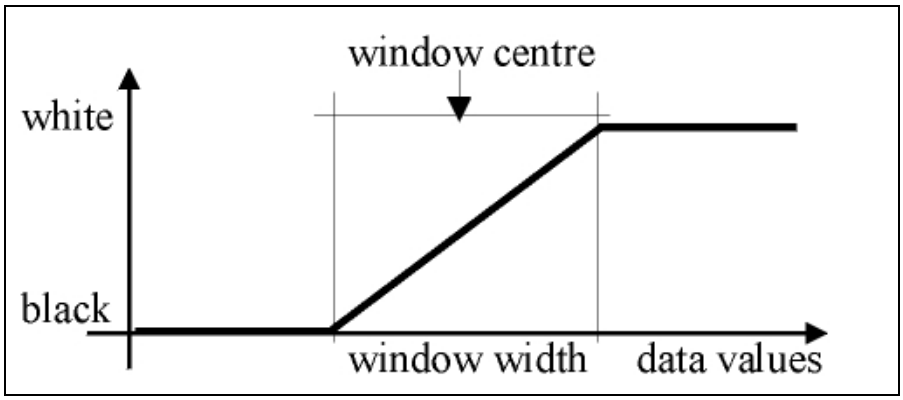


Figure 3.1: The windowing function

The drawback of slicing is the extremely time consuming process of inspection. The reason is that the objects of interest (blood vessels, in our case) are not entirely visible within one slice. Obviously it is impracticable for a surgeon to browse through more than 1000 slices per patient.

3.2 Direct volume rendering

Direct volume rendering is a visualization technique that is capable of displaying a volume dataset in its entirety. Internal structures can be emphasized by using transfer functions and transparency values, which map data values to optical properties. Several different algorithms exist for direct volume rendering:

- *Ray casting*: For every pixel in the pixel space a ray is cast through the voxel space (i.e., the cuboid made up by the volume data set). The direction and starting point of the ray depends on the viewing direction and the location of the observer. The resulting pixel value is computed from all voxels hit by the ray [2] by means of integrating the optical

properties of the traversed volume data set. Therefore, ray casting is a typical image-order rendering technique. An example image can be seen in figure 3.2. In general this algorithm is computationally expensive and not fast enough to display volume data interactively. For this reason the application of ray casting in an interactive medical environment is impractical.

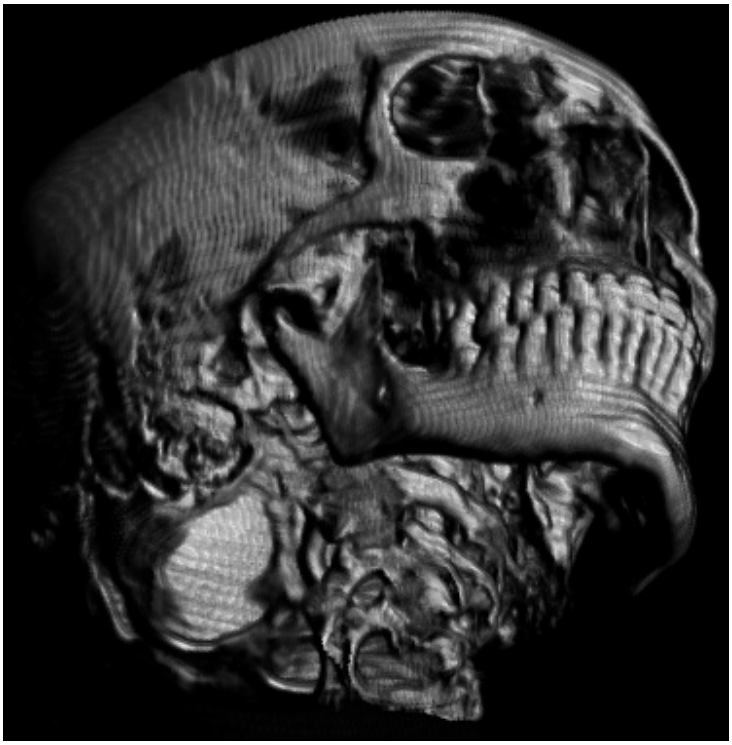


Figure 3.2: Direct volume rendering of a human head dataset. For this image a ray casting technique is used. The resolution of this sample-dataset is 184 x 256 x 170 voxels and the image resolution is 512 x 512 pixel.

- *Shear-warp factorization:* The basic idea of this technique is to achieve a trivial case traversing the viewing rays through the dataset by shearing the whole dataset in a first step. Shearing is done until all voxels along the viewing direction are aligned evenly in the computer's memory space. As a result of this first step a distorted image is produced. A second step has to be applied in order to achieve an undistorted picture. The basic process

can be seen in figure 3.3. The rendering speed can further be increased by run-length encoding the original data and skipping already opaque pixels in the pixel space. In this way interactive visualization of datasets with a size of up to 256^3 is possible [3]. The drawbacks of this object-order method are the poor zooming capabilities because of the 1:1 ratio between picture size and voxel set size.

Because of its simplicity this algorithm is used in a well-known hardware-based volume-rendering accelerator [4]. The VolumePro 500 board is an add on for PC-class computers based on the PCI interface. This graphics-accelerator device renders a volume of up to 256^3 voxels in real time. The lightning is based on the Phong lightning model. The quality of the final image (i.e., after user interaction) is enhanced using supersampling along the rays.

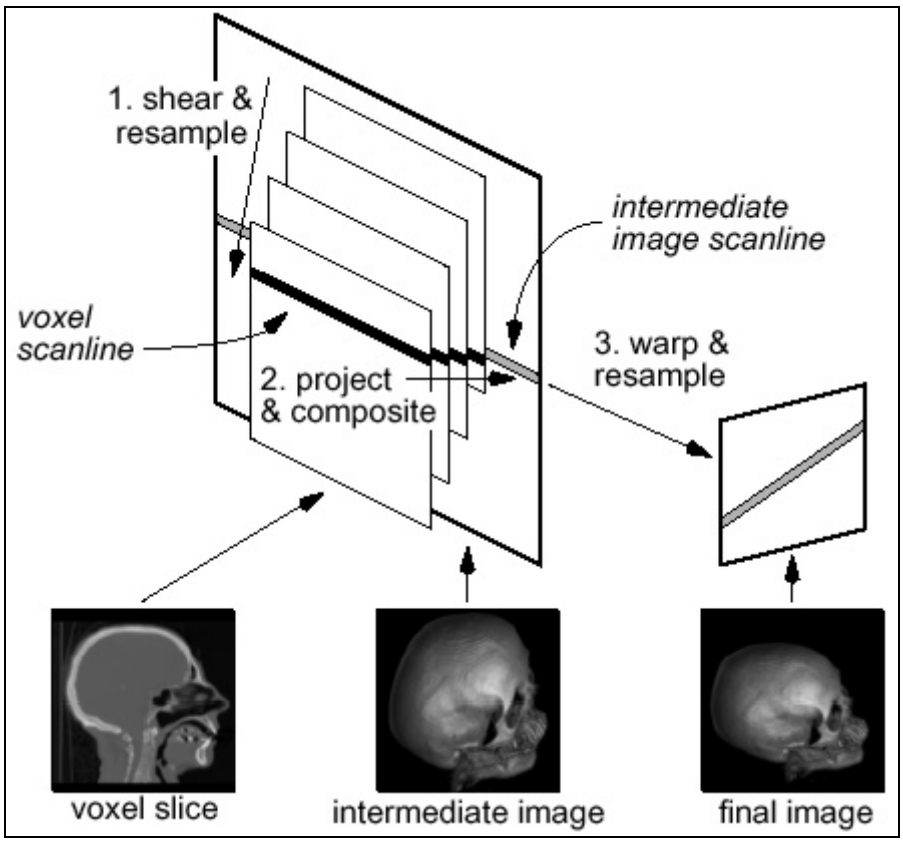


Figure 3.3: Shear-warp factorization [3].

- *Splatting*: This object-order rendering technique generates a footprint of every voxel in the pixel space [26]. One can image this process as throwing a snowball on a wall. The resulting footprints are composed within the pixel space. This technique produces high quality images in a possibly adaptive way. A tradeoff between fast rendering and high quality has to be done.

3.2.1 Maximum intensity projection

Maximum intensity projection (MIP) is a visualization technique, which extracts high intensity structures from a dataset. For each pixel only the most intense voxel value along the corresponding viewing ray is displayed. Therefore it does not matter if an area of voxels with lower intensity lies in front of a region with higher values or behind that region. Only the region with higher data values will be visible. As can be seen in figure 3.4 bones will always hide vessels if these structures cross each other. Note that the left MIP in figure 3.4 is taken from the same dataset as the MIP on the right side in figure 3.4. On the right image, all bone structures where segmented and removed. Therefore it provides a much better overview of the vessel tree as the MIP on the left side. For this reason it is desirable to provide a possibility to segment bone structures.

Fast algorithms exist for calculating maximum intensity projections. They are mainly based on the shear-warp factorization. In some cases an intelligent analysis of the data can eliminate voxels that will never have an impact on the resulting image. Acceptable frame rates can be achieved without hardware acceleration for datasets of a size of up to 256x256x124 even on low-end workstations [1].

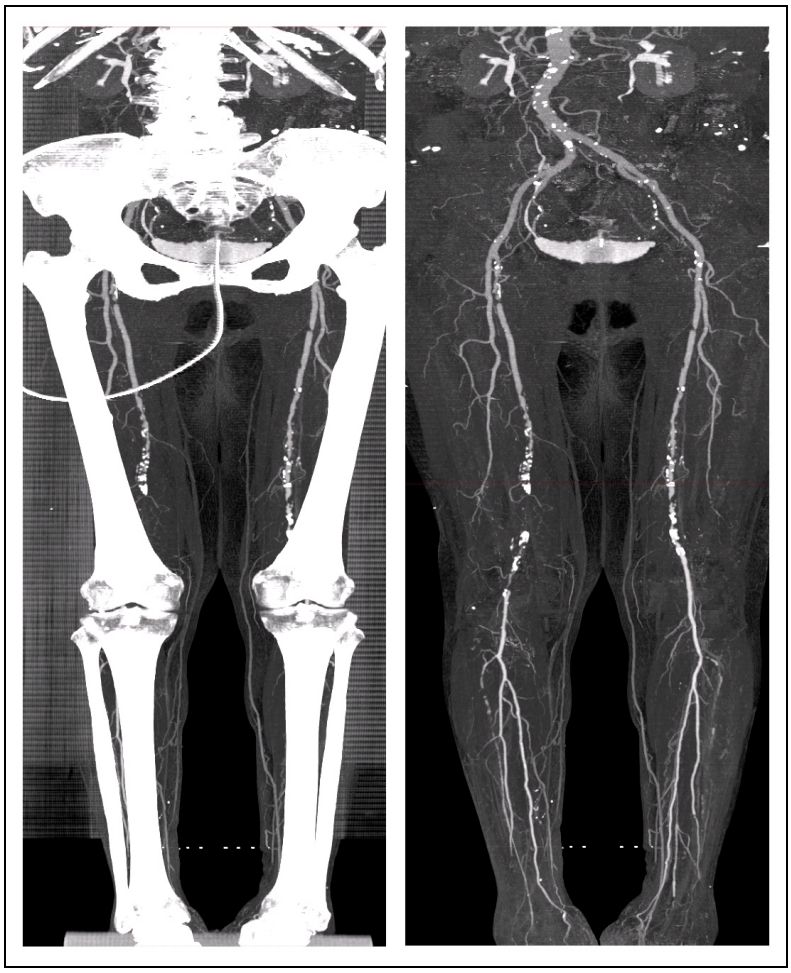


Figure 3.4: MIP of a CTA-dataset including bones (left) and bones removed (right).

3.2.2 Compositing in direct volume rendering

The compositing method takes all density values along a viewing ray into account. The traversal can be *front-to-back compositing*, which allows early termination when a certain opacity threshold is reached. Another possibility is *back-to-front compositing*, which does not need an accumulation buffer for the accumulated opacity.

To a voxel at position x_i an opacity $\alpha(x_i)$ is assigned according to its intensity value and its local density-gradient vector. Furthermore a color $c(x_i)$ is assigned to each voxel. The resulting colour value C for a pixel is defined recursively along the ray. The following equation is evaluated for back-to-front compositing [2]:

$$C_{j+1} = C_j \cdot (1 - \alpha(x_i)) + c(x_i) \cdot \alpha(x_i) \tag{3.1}$$

The equation for front-to-back compositing is similar:

$$C_{j+1} = C_j + (1 - \alpha_{akk}) \cdot c(x_i) \cdot \alpha(x_i) \tag{3.2}$$

where

$$\alpha_{akk} = \alpha_{akk} + (1 - \alpha_{akk}) \cdot \alpha(x_i) \tag{3.3}$$

Both methods produce the same results and only differ in terms of performance.

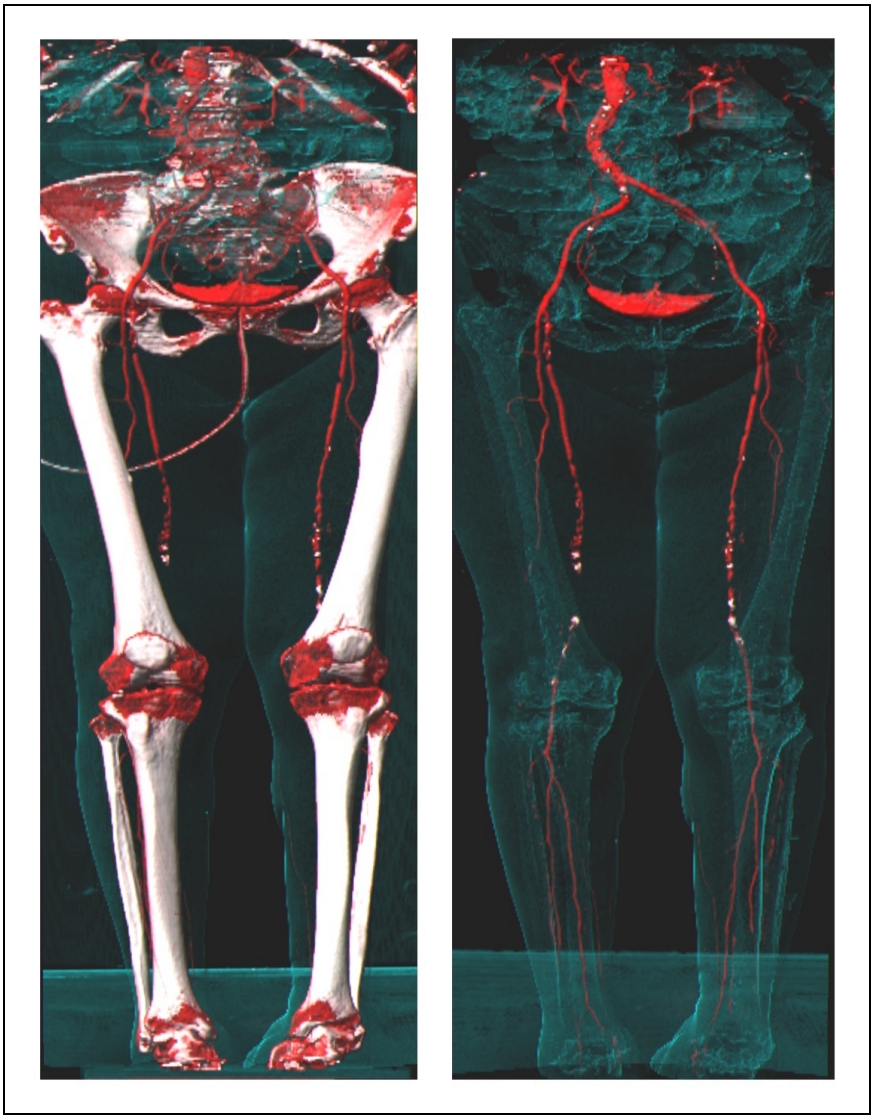


Figure 3.5: Left the CTA-dataset is shown. On the right side bones were removed. Both pictures are rendered using the same transfer function. The shape of the bones was made visible in the right image as a landmark for the surgeon.

Assigning a color and opacity to a certain vertex is defined by a so-called *transfer function*. A major goal of this technique is to extract areas of equal intensity from the dataset. Some of these regions may have a certain transparency in order to provide a better overview. This technique provides a kind of low-level tissue classification [2].
A transfer function can be defined highlighting the vessel structures by assigning a high transparency to density values usually related to bony structures. However, this method is not able to make bones from CTA-datasets entirely transparent. Due to bone marrow and partial volume effects the intensity histogram of bones overlaps with the intensity histogram of arteries as mentioned in chapter 2. Furthermore calcified parts of the artery would also disappear which is not desirable. This fact can be seen in figure 3.5.

3.3 Surface-fitting methods: Marching cubes

In contrast to direct volume rendering techniques described above surface-fitting methods use an intermediate representation of the volume data. This is often done by using polygons (i.e., triangles). The Marching cubes technique builds an iso-surface based on a user-defined threshold. Conceptually the dataset can be divided into small *cubes* consisting of 8 adjacent voxels. Each cube can be intersected by an iso-surface in 256 different ways. Therefore, 256 different *cases* of triangulation exist. The original method reduces these 256 cases by rotation and inversion to 15 topological different basic cases [5] [6]. A drawback of this solution is the existence of so called ambiguity cases. Ambiguity cases arise if it is not decidable how to triangulate two adjacent cubes. Due to these cases holes within the triangulated mesh can appear. A good solution is to extend the amount of basic cases to 23 [7]. With these extra basic cases inversion is not necessary anymore. This solves the problem entirely.
The output of the marching cubes algorithm is a triangulated mesh, which can be rendered using mainstream graphics acceleration. Therefore high rendering performance can be achieved even on low-level hardware systems. Selecting a

proper threshold is the main problem using this visualization technique. The value of the threshold has a crucial impact on the diameter and length of a vessel for instance (see figure 3.6). Another difficulty is to prevent arteries and bones from being merged. For this reason a segmentation algorithm should be applied before.

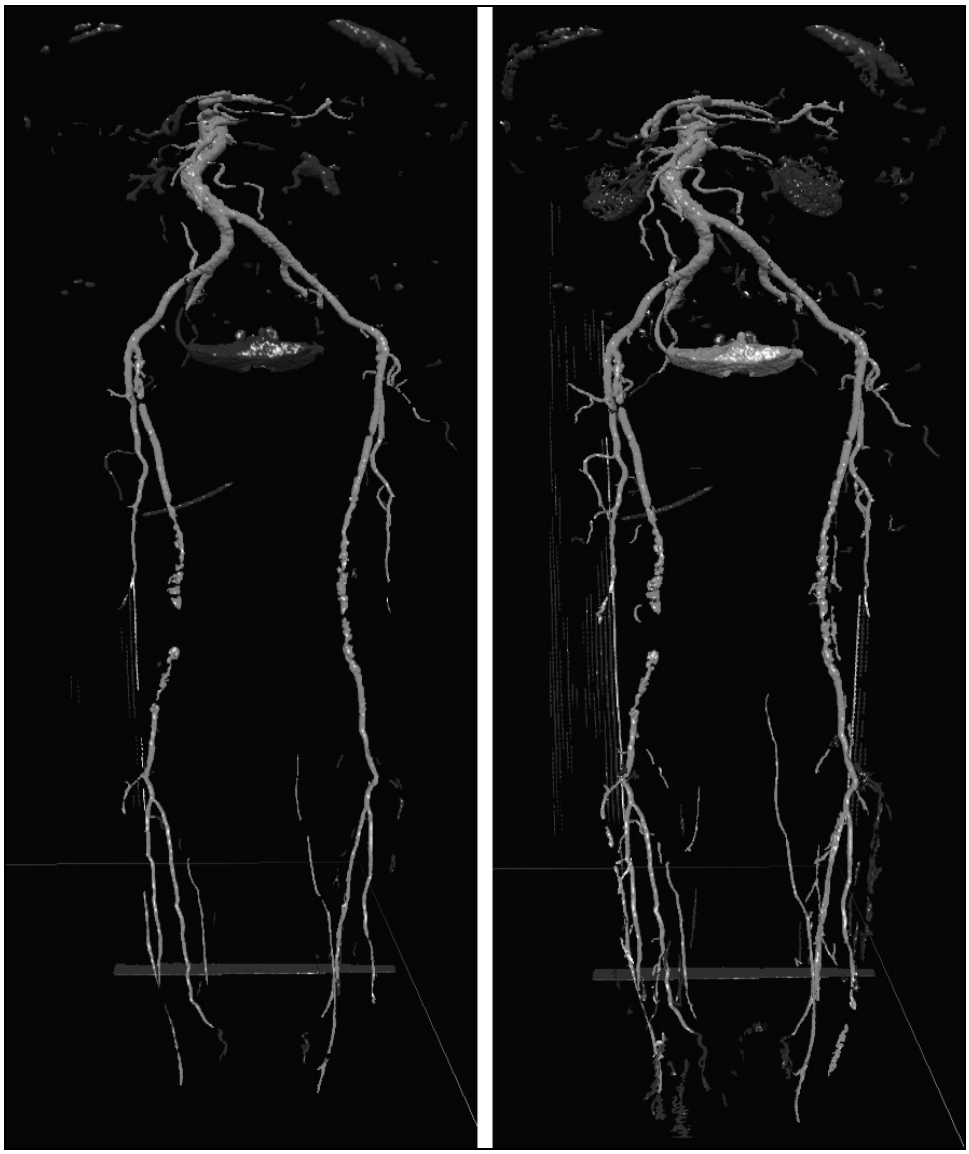


Figure 3.6: Marching cube iso-surface extraction using 180 HU (left) and 140 HU (right) as threshold. Bones were segmented and removed from the dataset previously.

3.4 Curved planar reformation

A *curved plane* in 3D space is described by a curved line and a vector. For each point of the curved line a straight line is defined which is collinear to this vector. The result is a curved plane in 3D space. Figure 3.7 shows a curved line on the left side. On the right side a curved plane is constructed using a vector that is parallel to the x-axis for each point of the line.

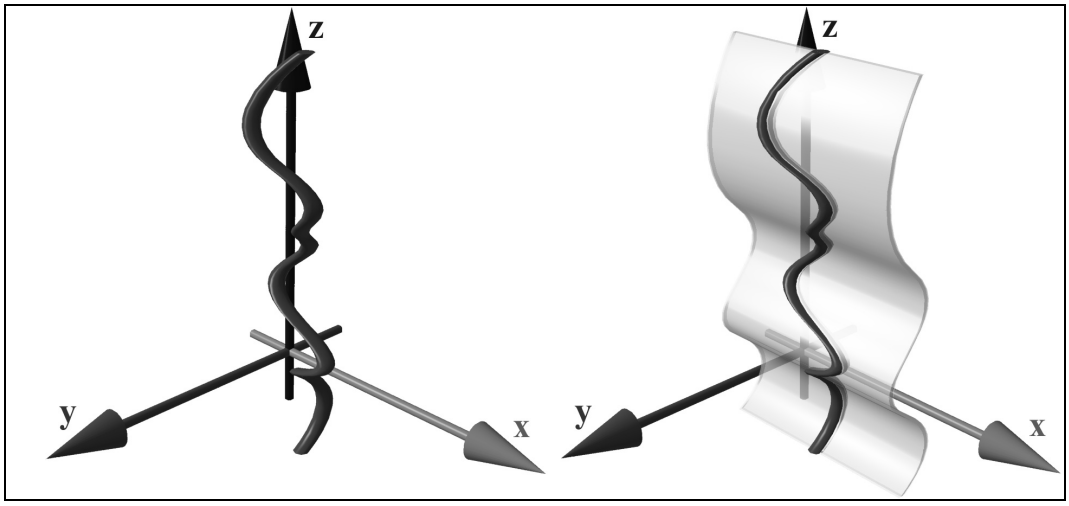


Figure 3.7: Left: A curved spatial line. Right: A curved plane defined by the curved line and a vector parallel to the x-axis.

The process of extracting a set of voxels lying on the curved plane and displaying this set as a straightened plane is called *curved planar reformation*. This process distorts the resulting image in terms of distances and anatomic relationships.
However, this visualization technique resolves the problem of overlapping (i.e., occluding) objects. A curved planar reformation of an artery can be seen in figure 3.8. Another advantage of this visualization method is that artery diseases can be seen very fast.
The main disadvantage is the time consuming and error prone generation process if the curved line has to be defined manually. Another point is that for every

vessel a different curved planar reformation has to be done. Which means it is not possible to display all arteries simultaneously in one image.

Figure 3.8: Curved planar reformation of an artery inside a CTA-dataset

3.5 Conclusions

This short overview of applicable visualization techniques for CTA showed that satisfying methods exist in case that some pre-processing has been done. Maximum intensity projection and direct volume rendering using compositing as well as the marching cubes algorithm require a segmentation step prior to the

visualization process (bone removal). On the other hand the curved planar reformation needs an automatic pre-processing method to find a proper curved line inside the arteries.

Chapter 4

Segmentation techniques

The task of segmentation is to separate an image or volume of optical or physical properties into different meaningful objects. In the area of CTA the objects of interest are vessels and bones. As segmentation has been a field of research for many years, the following collection is far from being complete however it includes the most common methods.

4.1 Segmentation of bone structures

Segmentation focused on bones attempts to identify bones and detect their correct shape. These processes are known as *recognition* and *delineation*. Recognition is the method of identifying the approximate location of a particular object of interest and of distinguishing it from other objects. Delineation is the process of specifying the precise spatial extent of an object [10].

4.1.1 Density thresholding and object labeling

Density thresholding is a rather simple segmentation method. However, it is very fast and a possible segmentation technique applicable to very large datasets in case interactive segmentation is required. [11] [12]. In a first step all voxels, which satisfy the threshold requirements (i.e. being in the range of two specified

threshold values) are marked as potentially belonging to objects. These voxels are merged into an object if they are adjacent. A second step assigns a type to each object. This process is called *object labeling*. If intensity regions of different object types overlap, it is very likely that the segmentation process fails because different object types are merged. Therefore a so-called opening operation may be performed. The *opening operation* consists of an erosion operation and a subsequent dilation operation. *Erosion* shrinks a certain object shape by a predefined amount of voxels by peeling off layers of the object like the skins of an onion. This operation separates different regions of an object allowing the correct labeling of wrongly merged objects of different type. In order to recover the original size of the object a dilation operation is applied. The *dilation* operation enlarges the object by a defined amount of voxels by adding layers of voxels to the surface of the object.

4.1.2 The live-wire method

Live-wire is a user-steered boundary detection segmentation technique [11] [13] [14]. User-steered means in this context, that the user has to control the segmentation process and is able to intervene immediately if the segmentation process starts to fail. It is assumed that objects have relatively decidable boundaries. This means that the gradient magnitude is high on the border of the object. The segmentation process tries to identify these boundaries. Live-wire is also sometimes referred to as intelligent scissors [15].

4.1.2.1 User interaction

Using the mouse as input device the user defines a starting point. By moving the mouse cursor a line is automatically generated from the starting point to the mouse cursor location. This line is attracted to high gradient boundaries. By defining a new point (i.e. clicking a mouse button) the current line is “frozen”. The new point is considered to be a new starting point and the user may continue the process (see figure 4.1). The contour of the object is defined in a piecewise manner. Finally the contour can be closed and the interior area is segmented.

Figure 4.1: User-steered live-wire segmentation. The image was brightened in order to make the live-wire lines clearly visible. In black color the fixed line can be seen. The still interactive movable part is painted in a dark gray color.

4.1.2.2 Boundary detection

With live-wire boundary detection is formulated as a graph searching problem. Each pixel defines a node. The edges belonging to a node are defined by the transitions from the pixel to its eight adjacent pixels. Each edge is weighted by a local cost function. This function simulates the physical influence of external forces. *External forces* depend on properties of the image such as image gradient magnitude and direction. In contrast to external forces, *internal forces* only depend on the shape of the line, like for instance curvature. W. Barrett and E. Mortensen [15] propose formula (4.1) as local cost function: The local cost function $l(p,q)$ from pixel p to its neighboring pixel q consists of three weighted components. These components are the gradient magnitude $f_G(q)$, the Laplacian zero crossing $f_Z(q)$, and the gradient direction $f_D(p,q)$.

$$l(p,q) = \omega_G \cdot f_G(q) + \omega_Z \cdot f_Z(q) + \omega_D \cdot f_D(p,q) \tag{4.1}$$

Experiments have shown that appropriate weights are: ω_G=0.43, ω_Z=0.43 and ω_D=0.14. As high gradient magnitude *G* should correspond to low cost the function $f_G(q)$ is defined as follows:

$$f_G(q) = 1 - \frac{G(q)}{\max(G)} \tag{4.2}$$

The second component is based on the output I_L of the convolution of the image with the Laplacian edge operator. This operator is used for edge localization and the convolution of an image with this operator approximates the 2nd derivative of the image. If $I_L(q) = 0$ or $I_L(q)$ has a neighbor with different sign then $f_Z(q) = 0$. Else $f_Z(q) = 1$.

The third, less weighted function $f_D(p,q)$ penalizes radical changes in boundary direction and therefore acts as a smoothening operator. A high cost to an edge is assigned if the image gradient vector *G(p)* and *G(q)* are nearly parallel and the vector $\vec{v}_{pq} = \overrightarrow{PQ}$ is nearly perpendicular to *G(p)* or *G(q)*. If all three vectors *G(p)*, *G(q)* and $\vec{v}_{pq}$ are nearly parallel, a low cost value is added.

The cost function mentioned above assigns low costs to image areas like borders or well-defined edges. A line, which is most likely to be the border of an object, can be defined as the set of adjacent pixel pairs $(p,q) \in P$ from a starting point *s* to an ending point *e* satisfying the following expressions:

$$P = \{(p_0,q_0),(p_1,q_1),...,(p_i,q_i),(p_{i+1},q_{i+1}),...,(p_n,q_n)\} \tag{4.3}$$

where

$$q_i = p_{i+1} \forall i < n$$
$$p_o = s$$
$$q_n = e$$

and the cost of path *P* ($\sum_{(p,q)\in P} l(p,q)$) is minimal for all paths from *s* to *e*.

4.1.2.3 Optimal path computation

The computation of such a line is referred to as *single-source shortest paths problem*. It can be shown that finding a single optimal path from a given starting point has equal complexity as finding all optimal paths from a given start point. The most appropriate algorithm to solve this problem is referred to as Dijkstra's algorithm. The algorithm will be described in chapter 5.

4.2 Segmentation of blood vessels

Segmentation focused on vessels identifies vessels as objects. In contrast to bone segmentation this is more complicated because of possible disconnections due to occlusions and stenoses. Other difficulties arise with the variations in contrast medium and partial volume effects. Furthermore calcifications should not be removed from the resulting dataset.

4.2.1 Vessel tracking by delineation

Vessel tracking is a segmentation technique of low computational effort. Two user defined starting points are set as the input of the procedure. Based on a heuristic process the algorithm tries to find a path from these starting points to the end of the vessel. The resulting curved line can be taken as basis for a curved planar reformation. Another possibility is to segment the tracked vessel.

The algorithm is described in detail as follows [16]. First two points roughly corresponding to the beginning of the vessel axis are set by the user. Subsequent points of the vessel axis are calculated incrementally. The user-defined points are c_0 and c_1. Always the two last points are taken into account in order to approximate a plane, perpendicular to the vessel axis. In the first step the direction vector $\vec{a}$ is given by $\vec{a} = \overrightarrow{c_0 c_1}$. The step vector $\vec{b} = d_{\min} \cdot \vec{a}_n$ is the normalized direction vector $\vec{a}_n$ scaled with the last minimum vessel diameter $d_{\min}$ (see figure 4.2). The new point c is defined by $c = c_{i+1} + \vec{b}$. A plane can now be defined using

c and $\vec{b}$. Within this plane the new center point c_{new} is calculated as described below. The algorithm continues using c_{i+1} and c_{new} as input for the next step.

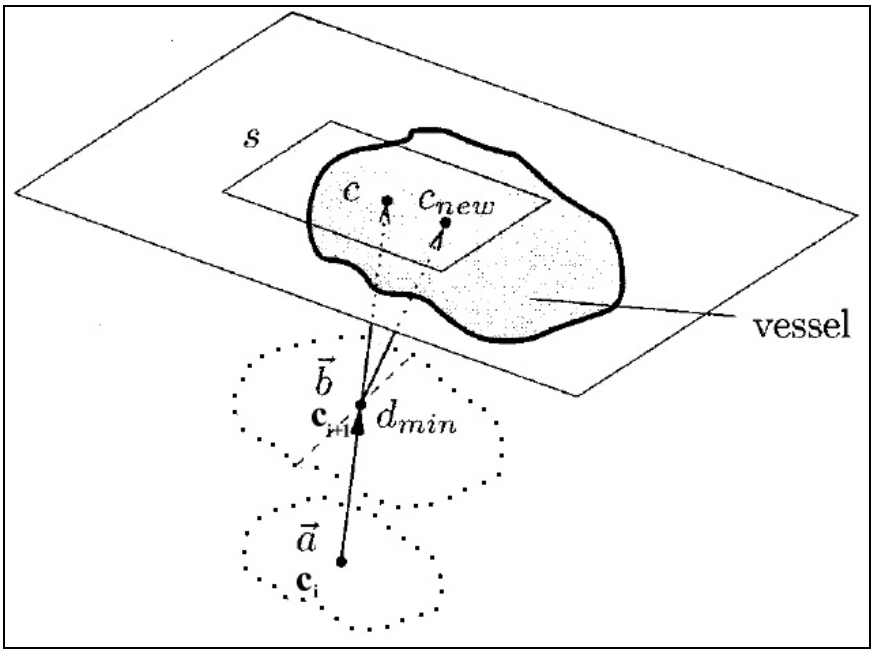


Figure 4.2: One vessel tracking step. The input points are c_i and c_{i+1} and the new center point c_{new} is computed [16]

The computation of the new center point c_{new} within the plane is based on a so-called likelihood function. For each point p in the search rectangle s (see figure 4.2) a set of lines l_i (each going through p) is computed. Each line is limited by the border of the vessel (see figure 4.3*c*). Point p divides each line l_i into $r_{1_{l_i}}$ and $r_{2_{l_i}}$.

$$C_L = \frac{1}{n}\sum_{i=1}^{n}\frac{\min(\left|\overrightarrow{r_{1_{l_i}}}\right|,\left|\overrightarrow{r_{2_{l_i}}}\right|)}{\max(\left|\overrightarrow{r_{1_{l_i}}}\right|,\left|\overrightarrow{r_{2_{l_i}}}\right|)} \tag{4.4}$$

The likelihood function C_L (formula 4.4) is computed for every point p in s. A height-field of the probabilities of being the center of the vessel results (figure 4.3*b*). The point with the maximum likelihood function is assumed to be the new center point c_{new}.

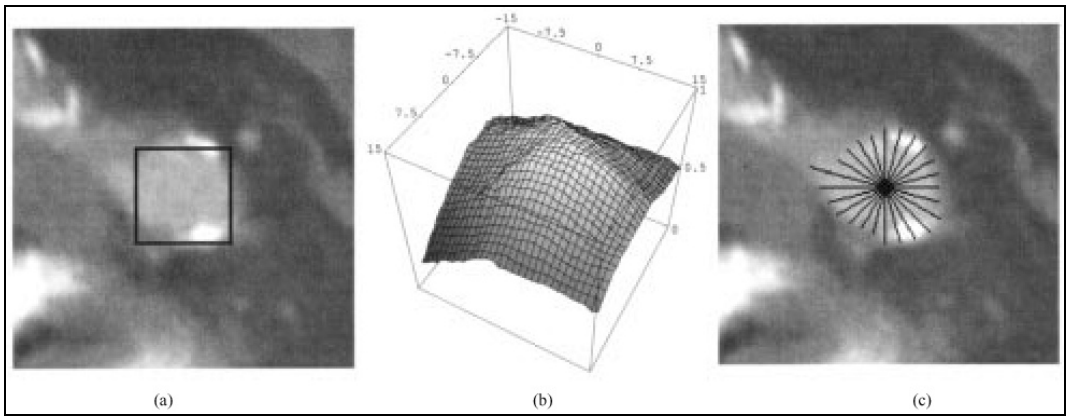


Figure 4.3:*a*) A cut plane with the search rectangle *s*. *b*) The corresponding probability heightfield C_L. *c*) Lines used for likelihood computation. This figure has been taken from [16].

4.2.2 The wave algorithm

The wave algorithm [17] is a special technique for the segmentation of vessels in contrast enhanced CT data. After a pre-processing step, which consists of filtering and thresholding, the wave algorithm is applied to the pre-segmented data.

Further, based on a region growing strategy and starting from a seed point (the first wave) all direct neighbor voxels are checked, if they belong to the vessel tree or not. If yes, these voxels are added to a new wave. To ensure that every voxel is only included once into a wave and to ensure that the wave does not run backwards, a history structure is created for every voxel.

To detect branchings in the vessel, every wave is checked if it consists of non-connected parts. If so, the wave is split into separate waves, which are processed independently of each other. To avoid a wrong detection of bifurcations, a correction of the direction of the wave is introduced. To get the direction of the wave spreading the center point of every wave is calculated.

This approach yields a graph, which stores the bifurcations and end points of vessels.

4.2.3 Segmentation based on Multiscale 3D filtering

Multiscale 3D filtering is a very powerful but computationally expensive technique. A dataset of 256x256x102 voxels uses about 10 minutes on a Sparc Enterprise Server with eight CPUs and 1GB main memory for multiscale filtering [20].

The main idea is to take into account the local structure of an object. Three main types of object structures exist: lines, sheets, and blobs. These three structures correspond to vessels, bone cortices, and nodules. From the point of view of CTA investigation line-like structures are of course the most interesting kind of object structure [18].

The basis of this approach is the so-called Hessian matrix:

$$\nabla^2 f = \begin{bmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{bmatrix} \tag{4.5}$$

Where $f_{xx} = \frac{\partial^2}{\partial x^2} f(x)$ is the second derivative of *f(x)*. The expression *f(x)* is the intensity function (i.e. the density value) at a point $\boldsymbol{x}$ = *(x, y, z)*. The second derivatives of a discrete dataset can be determined by the convolution of the data with the second derivatives of a Gaussian filter [19]. The D-dimensional Gaussian is defined as:

$$G(x,s) = \frac{1}{\sqrt[D]{2 \cdot \pi \cdot \sigma^2}} \cdot e^{-\frac{\|x\|^2}{2 \cdot \sigma^2}} \tag{4.6}$$

As can be seen in figure 4.4 the kernels of the partial derivatives of the Gaussian are symmetric. Therefore the Hessian matrix is also symmetric. Symmetric matrices have orthogonal *eigenvectors* e_i. The geometric interpretation of the Hessian matrix is that a sphere centered at the origin is mapped by the Hessian matrix to an ellipsoid whose axes are along the directions given by the eigenvectors. The length of the axes are the magnitudes of the corresponding *eigenvalues* v_i. This ellipsoid locally describes the second order structure of the image.

The eigenvector and eigenvalues are defined as follows:

A vector $x \in \Re^n, x \neq 0$ is called eigenvector of a $(n \times n)$-matrix A correlated to the eigenvalue $v \in \Re$ if $Ax = vx$.

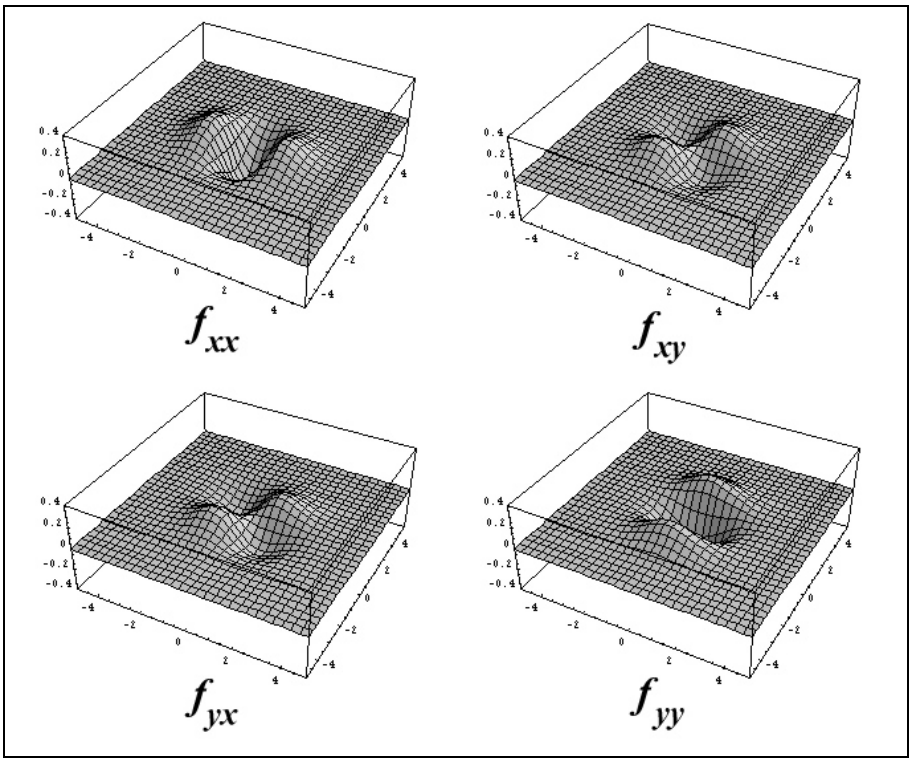


Figure 4.4: The four derivatives of the Gaussian filter (2D case). In this example $\sigma = 1$.

Focusing on the eigenvalues a correlation between the relationship of the eigenvalues v_i and the local structure can be established as can be seen in table 4.1.

Local Structure	Eigenvalue Condition	Example
sheet	$v_3 << v_2 \cong v_1 \cong 0$	cortex
line	$v_3 \cong v_2 << v_1 \cong 0$	vessel
blob	$v_3 \cong v_2 \cong v_1 << 0$	nodule

Table 4.1: Basic condition for each local structure and representative anatomical structures [18].

The eigenvalues are where v_1 is the highest eigenvalue and v_3 is the lowest eigenvalue. For CTA the line structure is the most interesting structure and will be discussed in more detail. It is very intuitive that along the vessel axis the second derivation should be close to zero. Thus a vessel is a line-like structure where the curvature in all other directions than the one parallel to the vessel axis should be high. Therefore v_2 and v_3 should have values well bellow zero. Arteries are assumed to have a rather circular cross section. This is true because of the high blood pressure inside arteries. For this reason v_2 and v_3 should have nearly equal values.

According to these relationships separation functions can be defined enhancing the different local structures. Combining them with other properties like the intensity value or the gradient magnitude, multidimensional thresholding can be applied in order to classify voxels. The method described above is also sometimes referred to as vesselness filtering.

Local structures can exist at various scales. For example the cross section of the aorta is larger than the cross section of collateral vessels in the lower portion of the vessel tree. The response of the Gaussian filter will be maximum at a scale that approximately matches the size of the vessel to detect. By adjusting the standard deviation of Gaussian convolution, local structures with a specific range of widths can be enhanced. A good solution is to apply filtering and eigenvalue computation several times at different scales. The maximum value of each computation pass is finally taken into account for classification.

4.3 Other, general segmentation techniques

This section contains methods, which do not concentrate on a specific tissue type. These methods are very general and do not make use of the problem specific properties.

4.3.1 Region growing

Region growing is a well-known technique for image segmentation. Its efficiency mainly depends on the employed aggregation criterion [21]. Region growing is initiated by the specification of seed locations. The algorithm attempts to merge neighboring voxels into this growing regions until no more voxels can be added. The goal is to obtain final regions that correspond to whole objects. The process of merging voxels is usually based on density values.

In order to produce larger regions a *region-merging* step can be applied after the first step [8]. The region merging process is often governed by a homogeneity criterion.

Speeding up region growing can be done using parallel computing as in [9].

4.3.2 Water-shed segmentation

Water-shed is also a well-established segmentation technique [22]. The algorithm works as follows: A gray scale image can be considered as a topographic relief: the gray level of a pixel resolves to the elevation of a point, the basins and valleys of the relief correspond to the darker areas, whereas the mountains and the crest lines correspond to the lighter areas (see chapter 2). The watershed line may be intuitively introduced as the set of points where a drop of water, falling there, may flow down towards several catchment basins of the relief.

Basically watershed algorithms can be divided in two groups [23]. The first group contains the algorithms simulating the flooding process. The second group is made up of procedures aiming at the direct detection of the watershed points.

In order to identify homogeneous regions by means of the watershed transform, the CT-dataset should be first subjected to a smoothening filter. Since the original data is very likely to contain noise and therefore a huge amount of small basins, the technique results in data over-segmentation. A lot of tiny regions would be identified. This problem can be solved using hierarchical segmentation or techniques similar to region merging [23].

4.4 Conclusions

Pure density thresholding does not require any user interaction. As intensity ranges of objects overlap, due to calcifications, bone marrow and partial volume effects, automatically segmented datasets are not very satisfying. Thresholding is however a very fast technique and therefore applicable for nearly interactive user-interaction for even large datasets.

Live-wire is a very powerful segmentation technique. Nevertheless it is a 2D segmentation technique. Even though 3D extensions of live-wire exist the working space is still 2D [10]. To develop a user friendly or even generalized 3D segmentation method based on live-wire is an interesting and promising field of research. Especially the property of contour catching is very important for segmentation of medical datasets. This feature for instance solves the problem of low-density marrow inside bones described in chapter 2.

Vessel tracking is a fast segmentation technique. However substantial drawbacks have to be taken into account. In the case of high-grade stenoses and occlusions the vessel tracker algorithm is very likely to loose the correct center path. The algorithm needs user intervention in these cases. Problems also arise in the case of bifurcations because the possibilities how to proceed are ambiguous. So far no results for small vessels have been presented in the scientific literature. It has to be expected that various problems arise when applying the algorithm on these vessels.

It is very likely that the wave algorithm also yields problems when trying to overcome occlusions. Handling stenoses and occlusions can be considered as one of the main challenges of segmentation algorithms focusing on vessel structures.

The multiscale filtering approach is a powerful technique with a sophisticated mathematical background. The main problem is the high computational effort. Especially the fact that the Gaussian kernel has to be at about the same size as the objects that should be enhanced has a high impact on the computational costs.

Region growing suffers from the tight spatial relationships of arteries and bones. In this case partial volume effects are leading to incorrect results.

As water-shed segmentation needs clearly defined valleys and basins in order to produce correct results it is very unlikely to apply this method successfully on regions as described in figure 2.4. A possibility to solve this problem is to use the algorithm with the gradients of the dataset. A drawback might be that gradients are sensitive to noise.

Finally it can be stated that many different and powerful segmentation techniques exist. Yet every method has its advantages and drawbacks. Until now the one ‘perfect’ segmentation technique has not been discovered. Especially in medical image analysis datasets are very complex. Therefore up to now the human operator is often obliged to intervene during a segmentation process in order to prevent incorrect results.

Chapter 5

Vessel investigation

Basically two different approaches were followed in order to provide a feasible tool for investigating CTA datasets. The first is to generate a curved planar reformation (CPR) as described in chapter 3. This method is already used in medical environments. Therefore it is a visualization technique that is very likely to be accepted for daily clinical use by the medical personnel in hospitals. One of the biggest disadvantages of this technique is the very time-consuming and error prone manual generation process. For this reason a semiautomatic generation method is desirable. The following section is going to describe such an approach.
The second approach is quite different. As the vessel tree in lower extremity areas consists of a huge amount of blood vessels of all sizes it is very difficult to identify every single vessel. This is true for algorithms as well as for experienced human operators. For instance in the case of very small vessels with a diameter in the range of 1–2 voxels partial volume effects have a high impact on the vessel size computed by the algorithm. Therefore algorithms identifying vessels as distinct objects or algorithms trying to follow vessel structures may produce incorrect results. The human operator on the other hand may find it impractical to identify every single vessel. These small vessel structures are nevertheless important to a radiologist. For instance the lumen of the small collateral arteries may allow a deduction of the spatial extend of a stenosis of the main artery. If

these secondary arteries are relatively large in size blood supply of subsequent vessels through these secondary arteries can be assumed.
The basic idea is that if it is difficult to identify the structures of interest, it might be easier to hide structures of less or no importance. Following this approach the whole vessel tree can be made visible by removing the bones from the dataset.

5.1 Curved planar reformation generation

Manually defining a curved planar reformation means that an initial cross section is rotated by the human operator until the object of interest is visible in its entirety. Very often this user interaction is done by adjusting a line in top view, in side view, and in frontal view. This line plus a vector corresponds to the curved plane in three-dimensional space. The manual generation of curved planar reformations is thus a very time-consuming task. For each vessel of interest the same procedure has to be repeated.
Another important fact is the correctness of the curved planar reformation. Tubular structures like vessels need to be cut through their central axes in order to show the true diameter as the curved planar reformation. Any deviation from this axis results in wrong results (see figure 5.1).

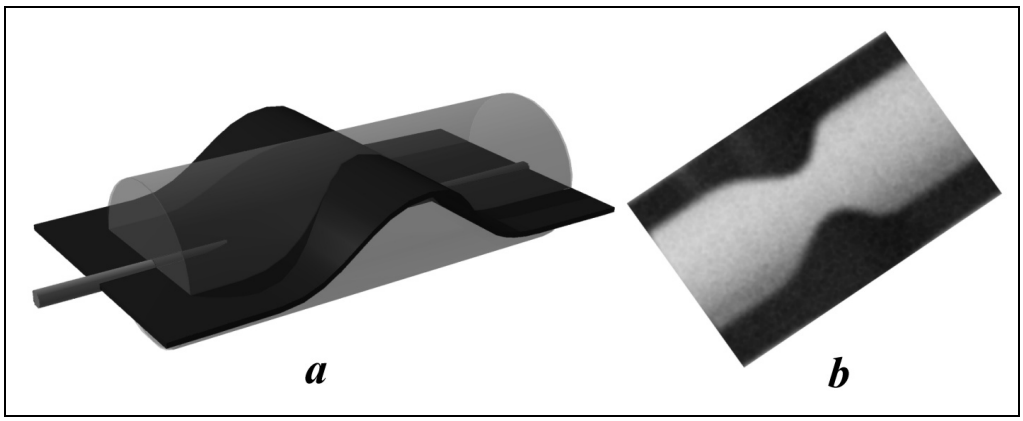


Figure 5.1: *a*) A curved plane through the vessel with a deviation from the central axis. *b*) The error caused by the deviation on a curved planar reformation, (false stenosis).

It is important to avoid such errors because of this similarity to stenoses. The radiologist might misinterpret these errors as stenoses and the danger of a incorrect diagnosis exists.

In the following sections a method is described to generate curved planar reformations in a semi automatic way. The terms optimal path and shortest path are used as synonyms in the following sections.

5.1.1 User interaction

Due to the planned integration of the algorithm into an existing software system the user interface provides only a minimum functionality. First the user identifies a starting point and at least one endpoint within each desired vessel. Afterwards the curved line computation can be started from the attached popup menu. In figure 5.2 the right iliac artery was chosen for investigation.

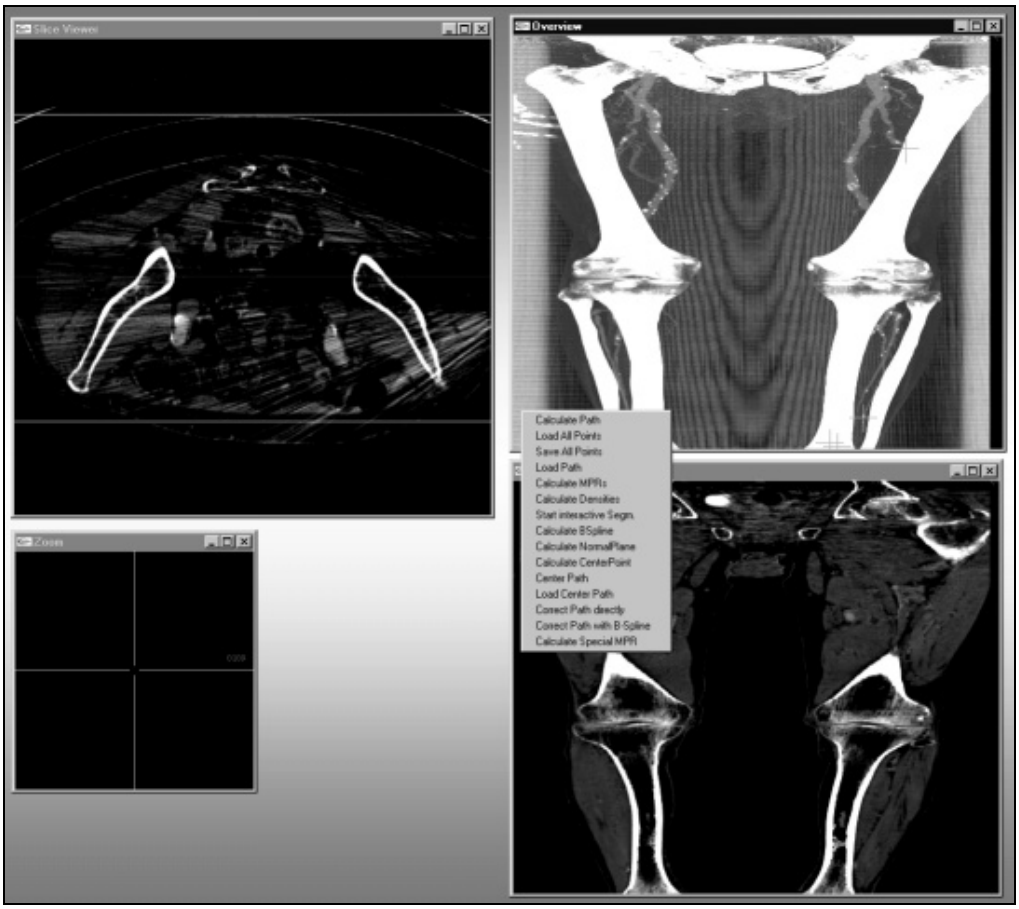


Figure 5.2.: The user interface for curved planar reformation computation

5.1.2 Curved line computation

The curved line computation is based on a modified live-wire algorithm. A proper cost function is defined as to keep the calculated path inside the vessel. In contrast to the 2D live-wire method this algorithm searches in 3D space for the optimal path.

5.1.3 Cost function

The cost function is defined in order to make it cheap for the algorithm to go through contrast-enhanced vessels. The local cost function consists of an internal force and some external forces. The internal force f_S helps to keep the path short. This is necessary to avoid undesired high curvature regions of the curved line. Since this function represents the cost per step, f_S is constant:

$$f_S(x) = c_{step} \tag{5.1}$$

Empirical experiences have shown that 200 is an appropriate value for c_{step}.

The external forces consist of the density interval function f_I, the gradient function f_G and the Laplacian function f_L.

The *density interval function* f_I ensures that the path stays inside the vessel. In the following equations the density function of a voxel x is considered to be *f(x)*. The function f_I is defined as follows (see figure 5.3):

$$f_I(x) = \begin{cases} \infty & f(x) < c_{lowerBorder} \\ (c_{lower} - f(x)) \cdot \omega_{lower} & c_{lowerBorder} \le f(x) < c_{lower} \\ 0 & c_{lower} \le f(x) \le c_{upper} \\ (f(x) - c_{upper}) \cdot \omega_{upper} & c_{upper} < f(x) \le c_{upperBorder} \\ \infty & c_{upperBorder} < f(x) \end{cases} \tag{5.2}$$

Basically this function consists of four thresholds and two weighting factors. The thresholds $c_{lowerBorder}$ and $c_{upperBorder}$ define the valid intensity region. Vessels with density values within these boundaries are detected. These boundaries assure that the calculated path does not leave the vessel. Within the interval defined by the other two thresholds c_{lower} and c_{upper} the intensity area is considered to be optimal. Therefore no punishment is given for voxels having these density values. The path

on the border of a vessel is intended to be more costly than in the center of the vessel. For this reason all density values between the thresholds $c_{lowerBorder}$ and c_{lower} as well as between the thresholds c_{upper} and $c_{upperBorder}$ are penalized. The impact of this penalty can be adjusted by the weights ω_{upper} and ω_{lower}. Function $f_I(x)$ is shown in figure 5.3.

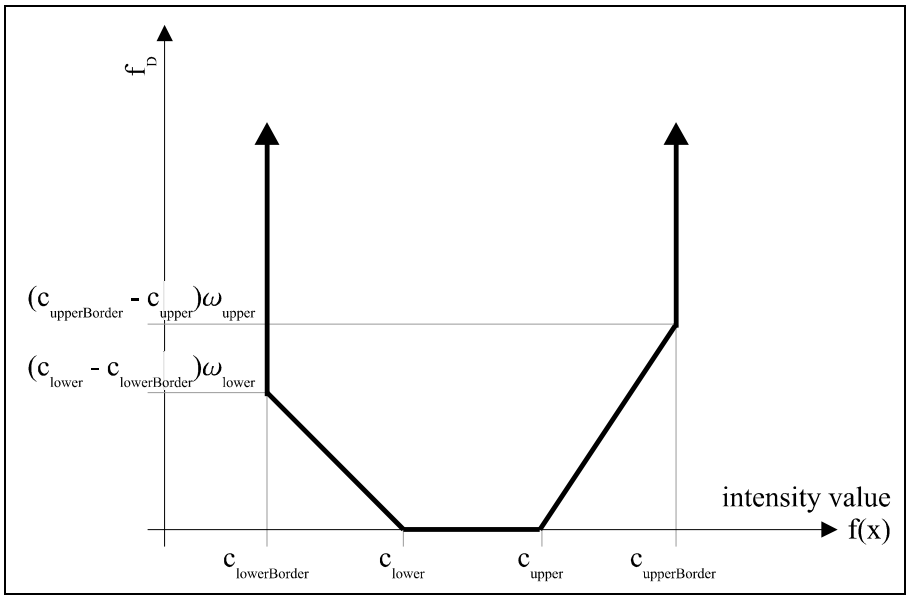


Figure 5.3: Density interval function

The *gradient function* $f_G(x,y)$ gives the difference of the intensity value of the previous voxel x and the intensity value of the subsequent voxel y. This function results from the assumption that in the direction of the central axis of the vessel the gradient magnitude is lower than in the direction of the vessel boundary.

$$f_G(x,y) = \left| f(x) - f(y) \right| \tag{5.3}$$

Finally the *Laplacian function* $f_L(x)$ prevents the algorithm from tracking along and into bones. As mentioned in chapter 2, bones do not have homogeneous intensity values. Therefore the boundary of the bone structure consists of density values, which would be preferred by the algorithm. This circumstance should be avoided because wrong paths are likely to be calculated and a larger volume has to be processed. In some cases it is also possible for the algorithm to track into the

bone structure. This for instance can be possible near the knee. Inside the bone structure marrow allows the algorithm to continue. The result is an optimal path that however does not correspond to the vessel structure anymore. In figure 5.4 this situation can be seen in image *a)* and *b)*.

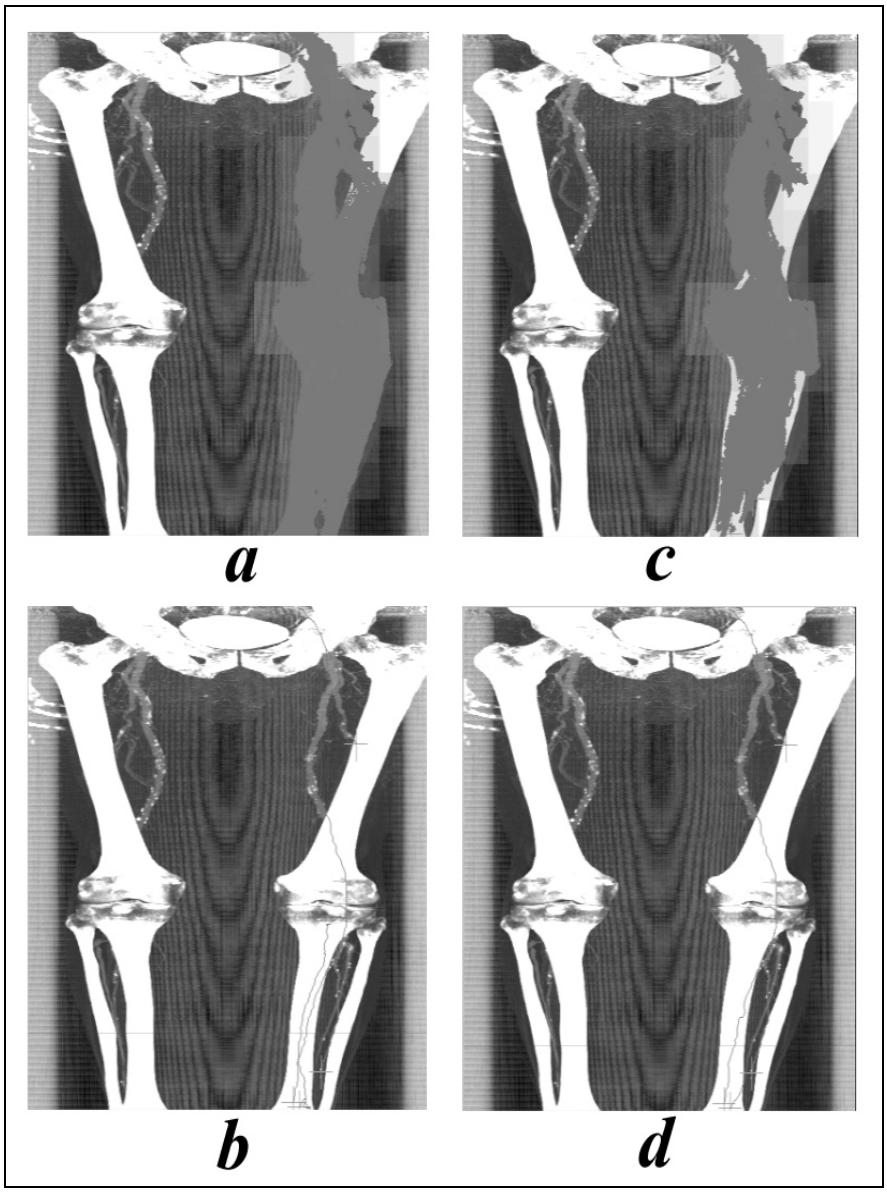


Figure 5.4: a) The region computed without Laplacian function. b) The erroneous paths calculated without Laplacian function. c) Reduced area affected by the algorithm due to the Laplacian function. d) Correct path-calculation making use of the Laplacian function.

The function f_L is defined as follows:

$$f_L(x) = \begin{cases} \infty & (L \otimes f)(x) > c_{laplace} \\ 0 & else \end{cases} \tag{5.4}$$

where the Laplacian kernel L is defined in 2D as:

$$L = \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 0 & -4 & 0 & 2 \\ 3 & -4 & -16 & -4 & 3 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix} \tag{5.5}$$

and $c_{laplace}$ is a predefined threshold.

The local cost function $f_C(x,y)$ for a single step from a voxel x to the adjacent voxel y can be defined as:

$$f_C(x,y) = f_S(x) + f_I(x) + f_G(x,y) + f_L(x) \tag{5.6}$$

An improvement concerning performance can be realized taking into account the characteristics of CTA–datasets: A start point is always defined in one of the topmost slices. The endpoints are usually defined at the bottom area of the dataset as shown in figure 5.2. In order to push the algorithm into the correct direction a scaling of the local cost function according to the current slice can be done.

5.1.4 The path-finder algorithm

The method of curved line computation is in the following referred to as *path-finder algorithm*. As mentioned above the algorithm is based on the live-wire method. The differences between the path-finder algorithm and live-wire are:

- *Different working space*: Live-wire is working in 2D space whereas path-finder computes the optimal path in 3D.
- *Different local-cost function*: Live-wire tries to find a path following edges and object boundaries. Hence the cost-function penalizes regions with low gradient magnitude. In contrast path-finder is attracted by regions, which are very likely to be contrast-enhanced vessels. In other words the

cost function prefers regions with a special intensity value and low gradient magnitude.

- *Interactivity*: Live-wire is an interactive segmentation tool. This is one of the main key concepts of live wire. Due to the 3D working space and the large datasets mentioned in chapter 2, path-finder provides no interactive user interface.
- *Structure of the result*: The live-wire algorithm takes a starting point and the interactive movable endpoint. If the algorithm starts to fail an intermediate point is set. This operation takes the previously movable endpoint as new starting point and the actual mouse cursor position is the new endpoint. Hence one and only one endpoint is active at one time. The path-finder algorithm takes one starting point and *n* endpoints. Where *n* is only limited to an arbitrary constant for implementation reasons.

5.1.5 Computed-region reuse

One key feature of live-wire is interactivity as mentioned above. In order to maintain this feature it is not possible to recompute the whole path when changing the endpoint. Therefore the already computed area is reused [13]. Even though path-finder is not an interactive algorithm this feature also makes sense to be included in path-finder. Whenever additional endpoints are defined the algorithm reuses the already computed area and therefore significantly reduces computation time. In the following the algorithm is described in further detail.
Three different sets of voxels can be defined (see figure 5.5):

- *L*: The set of already calculated voxels using Dijkstra's algorithm.
- *C*: The set of voxels that have not been taken into account yet.
- *Q*: The set of voxels located on the boundary between *L* and *C*.

Furthermore some information for each voxel is represented by:

- *cc(v)*: The cumulative cost of a path starting at point *s* and reaching *v*.

- *dir(v)*: The direction for the optimal path at the position of voxel *v*. In other words it is the pointer to the predecessor of the optimal path reaching *v*.

First of all a starting point *s* is defined, by the user for instance. This operation requires some initialization:

$$C = dataset$$
$$Q = \{\}$$
$$L = \{\}$$
$$cc(v) = \infty \quad dir(v) = null \quad \forall v \in C$$
$$cc(s) = 0 \quad Q \leftarrow s$$

The definition of the first endpoint e_1 starts Dijkstra's shortest path computation algorithm:

```
while e1 ∉ L do:
   remove a voxel v from Q with minimal cc(v).
   place v in L.
   for each adjacent voxel v' of v and v'∉ L do:
       compute temp = cc(v) + f_C(v,v') .
       if temp < cc(v')
           set cc(v') to temp.
           set dir(v') to the direction from v' to v.
           if v'∉ Q
               insert v' in Q.
           endif
       endif
   endwhile
endwhile
```

Following the calculated *direction information dir(v)* for every voxel *v* from the endpoint e_1 to the starting point *s* produces the shortest path.

If a new endpoint e_2 is defined two possibilities arise:

1. $e_2 \in L$: In this case simply the direction information *dir(e_2)* has to be followed. The reason for this is that all optimal paths for voxels in L are already computed according to Dijkstra's algorithm.
2. $e_2 \notin L$: The new endpoint was not computed yet. Therefore the algorithm is resumed with the new endpoint e_2. It is obvious that the sets L and Q can be reused because the boundary Q expands independent of the endpoints.

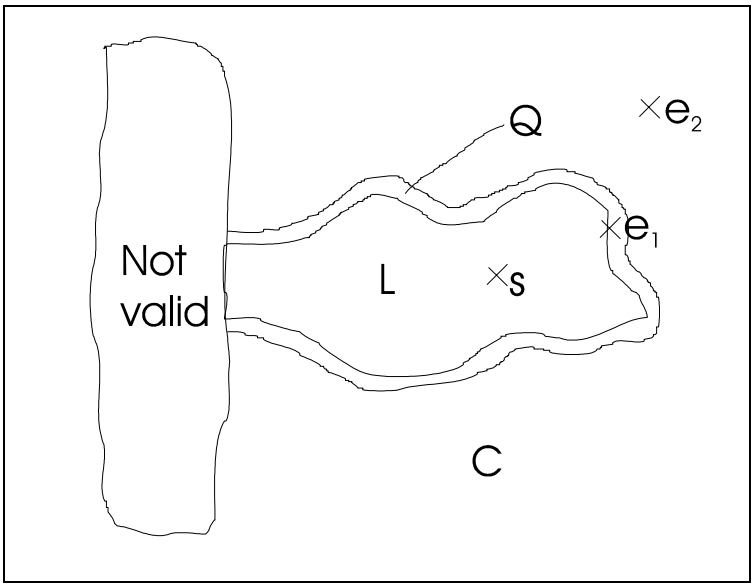


Figure 5.5: Outline computed-region reuse. *Not valid* marks a region where the algorithm should not continue searching for an optimal path (i.e., the cost function f_C is infinite).

5.1.6 Centering the path inside the vessel

As mentioned above any deviation from the vessel's main axis causes errors in the curved planar reformation. The pure path-finder algorithm does not assure the path to be in the center of the vessel. This can be seen in figure 5.6. It is obvious that the path-finder algorithm tries to find a curve within the vessel, which is as straight as possible. For this reason a feature is added to correct the path in terms of finding the center of the vessel. This algorithm is referred to as *center-finder*

algorithm. The main goal of this algorithm is to improve the path calculated by the path-finder algorithm. A comparison of before and after the center correction is applied can be seen in figure 5.6.

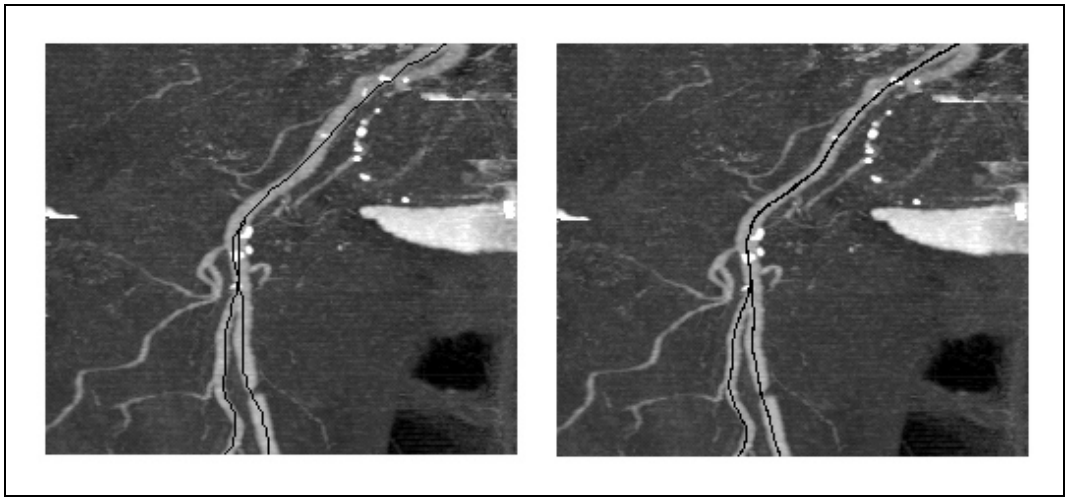


Figure 5.6: Left: Path-finder algorithm without center correction. Right: Path-finder algorithm with center correction.

5.1.7 The center-finder algorithm

The center-finder algorithm is similar to the vessel tracker approach in [16]. The difference to the approach in [16] is that the path from the starting point to the endpoint is already known (as approximated by the path-finder algorithm). For this reason no user interaction is necessary in cases of bifurcations or high-grade stenoses. The basic idea is to find the true vessel-center point in a plane perpendicular to the previously defined path. Within this plane the center point should be the midpoint of a circle. The algorithm consists of four main steps:

1. *Gradient computation*: This step approximates the tangent at each voxel of the original path. A B-spline curve is used in order to smoothen the gradient.

2. *Plane construction*: A 2D cross section is extracted from the 3D dataset according to the current point of the original path. It's generating vector is calculated from the B-spline curve (tangent vector).
3. *Center approximation*: The true center within the 2D plane is approximated. Afterwards the retransformation into 3D space is done.
4. *Path reconstruction*: The new path consists of holes and loops because the points were moved in 3D space during step 3. These artifacts are removed.

The following sections describe each step in more detail.

5.1.7.1 Gradient computation

A plane in 3D can be defined by a point in 3D space and a 3D vector. This vector is perpendicular to the plane. Therefore a vector tangent to the central vessel axis together with a point on the according position on the vessel axis defines a perpendicular cutting plane of the vessel. Unfortunately none of the above mentioned prerequisites are given precisely. For this reasons some approximations have to be taken into account.

The local gradient was found to be much too noisy for perpendicular plane computation. Noise within the vessel causes the path to have very high curvature. Therefore a *B-spline curve* $B_{n,k}(t)$ was applied for gradient computation.

$$B_{n,k}(t) = \sum_{i=0}^{n} P_i \cdot N_{i,k}(t) \tag{5.7}$$

where the *blending function* $N_{i,k}(t)$ (sometimes also referred to as *B-spline basis function)* is defined as

$$\begin{aligned} N_{i,1}(t) &= \begin{cases} 1 & t_j \le t \le t_{j+1} \\ 0 & else \end{cases} \\ N_{i,m}(t) &= \frac{(t - t_j) \cdot N_{j,m-1}(t)}{t_{j+m-1} - t_j} + \frac{(t_{j+m} - t) \cdot N_{j+1,m+1}(t)}{t_{j+m} - t_{j+1}} \\ for \quad & m = 2,3,.....k \end{aligned} \tag{5.8}$$

and the knot vector or knot sequence t_j is given by

$$t_j = \begin{cases} 0 & j < k \\ j - k + 1 & k \le j \le n \\ n - k + 2 & j > n \end{cases} \tag{5.9}$$

for t the suitable interval is defined as

$$t \in [0, n - k + 2] \tag{5.10}$$

where n is the number of control vertices.

Vessels tend to have quite low curvature. For this reason k = 40 seems to be suitable to approximate the structure of the vessel. The control vertexes of the B-spline curve are the points calculated by the path-finder algorithm. The vector necessary for the plane construction is taken from the B-spline curve. A 3D B-spline curve is shown in figure 5.7. The distance from one cross to the next is approximately one voxel. The high curvature of the original path can be seen clearly. However the low curvature of the vessel is followed by the B-spline curve correctly.

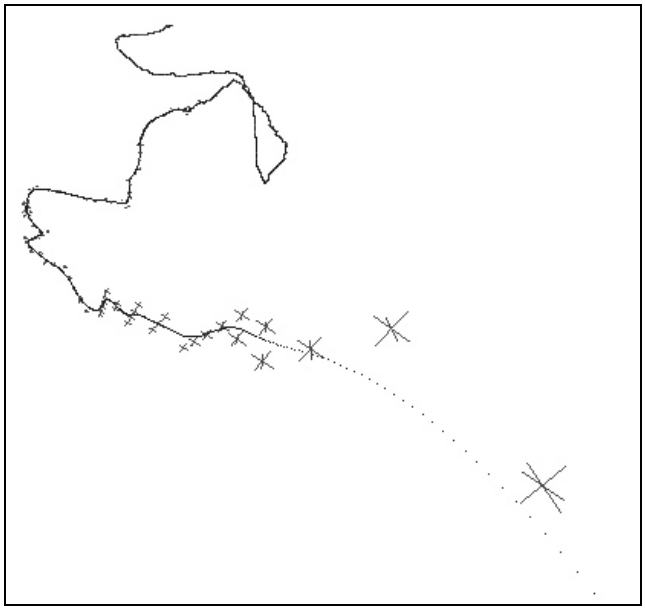


Figure 5.7: Dotted B-spline curve constructed from the control vertices marked by crosses.

5.1.7.2 Plane construction

In this step a plane through every point of the calculated path is generated. The plane construction step is using the tangent v_B from the B-spline curve at the current processed point P. From this vector v_B:

$$v_B = \begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix} \tag{5.11}$$

Two generating vectors can be defined. The first generating vector v_1 is normal to the vector v_B and is in the yz-plane.

$$v_1 = \begin{pmatrix} 0 \\ 1 \\ -\frac{y_B}{z_B} \end{pmatrix} \tag{5.12}$$

and finally v_2 is computed as the cross product of v_B and v_1.

$$v_2 = \begin{pmatrix} z_B + \frac{y_B^2}{z_B} \\ -\frac{y_B x_B}{z_B} \\ -x_B \end{pmatrix} \tag{5.13}$$

The plane can now be defined by

$$s = M + a \cdot v_1 + b \cdot v_2 \quad a,b \in [-d,d] \tag{5.14}$$

Where d defines the size of the rectangle s inside the plane. The densities within plane s are resampled using tri-linear interpolation. Within this plane the center point is computed. Similar to [16] the size d of the rectangle s inside the plane depends on the most recent vessel diameter.

5.1.7.3 Center approximation

In contrast to the method described in section 4.2.1 this center approximation approach is more inaccurate. However, it is far more efficient as only one point has to be computed. As defined in the previous section the current point of the

path being processed is exactly the midpoint of the plane. In the following this point is referred to as *M*. From this point *M m+2l* rays are cast in a radial fashion. The end points of the rays are defined by an upper and a lower threshold. The ray is extended until the density value is outside the density value region defined by the two thresholds. These thresholds should include the intensity range of vessels and calcifications. For the sake of fault tolerance the *l* longest and the *l* shortest rays are discarded. This improves the stability of the algorithm as single tracings into bones or early stops due to noise are not taken into account. At the end of each ray a point P_i can be defined. The distance d_i between the two adjacent points P_i and P_{i+1} is calculated. Figure 5.8 shows the above described situation.

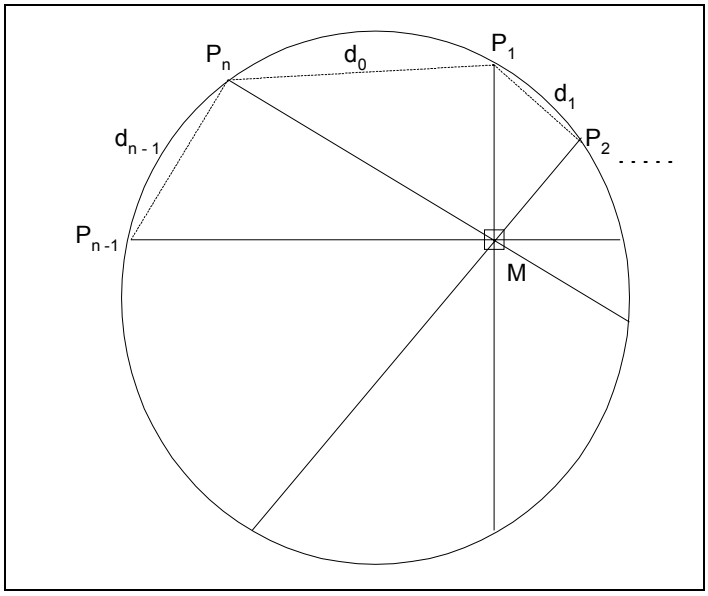


Figure 5.8: Finding the true center point.

Now the true center point *C* can be approximated by

$$C \approx \frac{\sum_{i=1}^{n} P_i \cdot (d_{i-1} + d_{(i) \bmod n})}{2 \cdot U} \tag{5.15}$$

where

$$U = \sum_{i=1}^{n} d_i \tag{5.16}$$

Even though this is an approximation of the true center point the computation produces satisfying results. The resulting center point C is still defined in 2D space. The transformation into 3D is accomplished by

$$C_{3D} = M + x_C \cdot v_1 + y_C \cdot v_2 \tag{5.17}$$

where x_C and y_C are the 2D coordinates of C and C_{3D} is the calculated center point in 3D space. This resulting center point C_{3D} replaces the original point M from the optimal path computation.

5.1.7.4 Path reconstruction

Due to the vessel center computation points of the optimal path are moved in 3D space and holes and loops may appear. Therefore another smoothing operator is applied. Again a B-spline curve is used. Hence k is set to 20, which is lower then for the gradient estimation mentioned previously. A higher value would have a too significant impact on the centered path. This operation closes holes and cuts loops. The set of control vertices is now given by the set of points produced by the center-finding algorithm. From this B-spline curve the corrected and centered path is computed. This path can now be used as a curved line to produce the curved plane reformation.

5.1.8 Conclusions

The curved line computation was found to be very reliable. The algorithm is very likely to produce correct results. On the other hand if the algorithm fails, the failure is obvious and therefore not dangerous. Dangerous is meant in terms of the possibility of a false diagnosis done by the radiologist. In some cases centering of the path is not necessary. In general it can be stated that the centering and correction of the path generated by the path-finder algorithm produces better results in the following curved planar reformation.

5.2 Gradient-enhanced segmentation of bones

The method we use for bone removal is a rather basic segmentation technique. A combination of thresholding and region growing is applied. One of the main reasons for using a simple but cost efficient approach is the large size of the datasets. Even though the datasets are large the objects of interest (the vessels) are often small and have a tight spatial relation to other objects. For this reason down sampling the data volume results in a substantial loss of information. Another reason is that no high quality bone segmentation is required. As long as small errors do not hide important features the method is applicable. An example of these minor errors are for instance the ribs. Due to their spatial relationship to the arteries a not correctly identified rib typically has no impact on the investigation by the radiologist.
This approach provides quite acceptable results. Together with the capability of user intervention the method provides a useful tool for the segmentation of bones.

5.2.1 The algorithm's outline

The *bone-seg algorithm* is working on so called *slabs*. A slab consists of several spatially adjacent volume image slices. Typical 30 to 50 slices are combined in one slab. The advantages of this arrangement are:

- *Better response time*: Because the amount of data is significantly smaller than the whole dataset the computation time after a user interaction is reduced.
- *Error containment*: The error inside one slab cannot propagate over the whole dataset.
- *Region dependent parameters*: As the correlations and properties of the objects differ depending on the regions within the dataset, the parameters can be set according to these regions.

The bone-seg algorithm is applied independently for each slab. Basically the algorithm consists of 3 steps. First a rough distinction between the different

objects is done. Secondly the objects are labeled. In the final step the correct shape is computed. A predefined set of parameters is used for each slab. The user can change this set of parameters during the segmentation process. For each slab the set of parameters consists of:

- t_{class}: This threshold is used to distinguish different objects.
- t_{expand}: This threshold defines the enhancement of already identified objects. This threshold is motivated to handle partial volume effects and marrow inside the bones.
- t_{label}: The t_{label} threshold is used for labeling. The threshold t_{label} is operating on the average density of objects. For implementation purpose the average density of an object is the sum of the densities minus t_{class}.

The basic steps for the segmentation of one slab are as followed:

```
for each slice s in slab t do
    classify voxels in s based on t_class.
    merge regions in s.
label regions in slab t.
for each slice s in slab t do
    classify voxels in s based on t_expand.
    merge regions in s.
(user interactive labelling)
remove regions classified as bone
```

First all slices are classified using a high threshold t_{class} in order to distinguish different objects. The connected regions are merged and finally labeled according to their properties. A second iteration of the whole process (except labeling) is done with a lower threshold t_{expand}. This step improves the quality of the segmented dataset by reducing noise due to partial volume effects and bone marrow. After this step a user defined labeling of objects is possible. Finally the objects labeled as bone are removed.

5.2.2 Classification

Classification is meant to be the process of identifying whether a voxel is a potential object or not. In this step no distinction is done if a voxel belongs to a bone structure or to a blood vessel. As it can be seen in figure 5.9 no clear decision can be made concerning the type of the voxel in the area of the knee if intensity and gradient magnitude are taken into account.

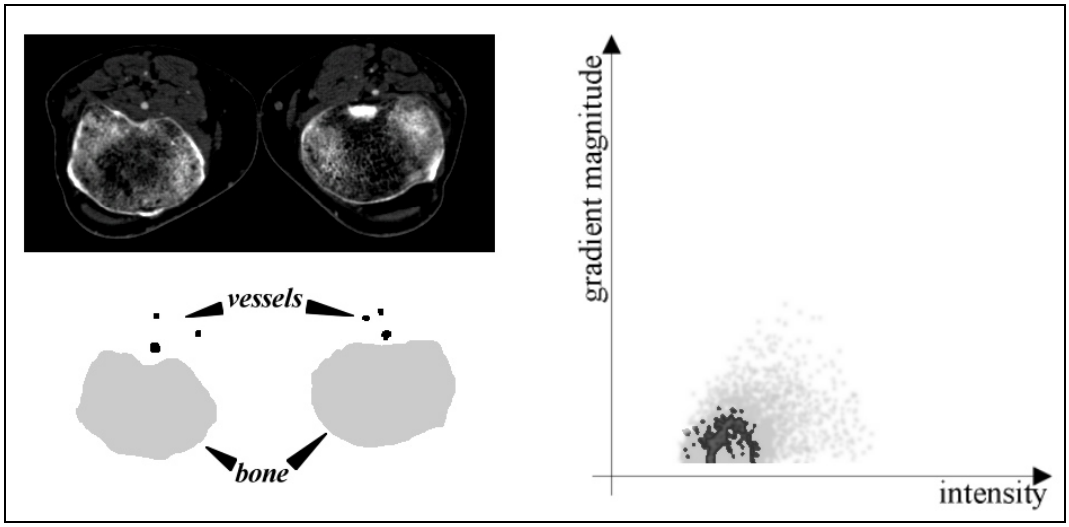


Figure 5.9: Left top: A slice taken from a CTA-dataset at the level of the knees. Left bottom: Manually segmented objects. Right: A scatter-plot showing the gradient magnitude and the intensity value according to the object type for each vessel.

The classification is done threshold based. The decision process is sketched in the following where *f(x)* is the intensity function and *f'(x)* is the gradient magnitude of an voxel *x*.

```
if f(x) > threshold
     if f(x) > (threshold + boundary_area)
          x is OBJECT
     else
          if (f'(x) > max_gradient)
               x is NO_OBJECT
```

```
          else
               x is OBJECT
else
     x is NO_OBJECT
```

The values for *max_gradient* and *boundary_area* are fixed. Empirical default values are 150 and 100 respectively. The *threshold* value is either t_{class} or t_{expand}, depending on the algorithm step.

Using this gradient enhanced method for classification, areas which are likely to be boundaries of objects are not classified. This can be seen in figure 5.10. Regions above the threshold t_{class} are depicted in dark gray or light gray. The dark gray areas do not satisfy the gradient condition in the classification process mentioned above. For this reason only the light gray points are investigated as potential objects in the next step. As can be seen in figure 5.10 vessels very close to the bone remain separate.

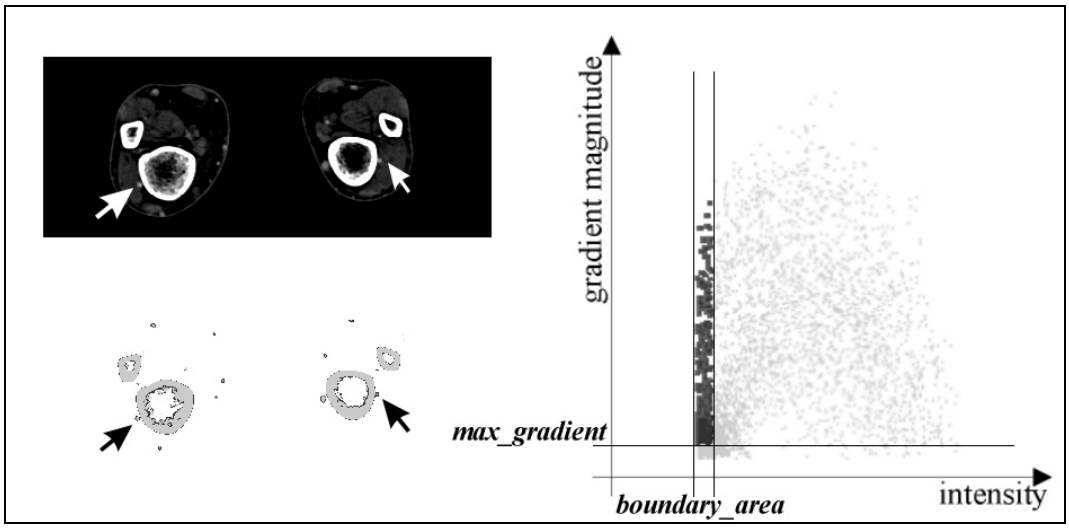


Figure 5.10: Left top: A slice of a CTA-dataset in the lower leg area. Two critical arteries are marked with an arrow. Left bottom: Computed classification. Objects are marked light gray. Dark gray areas are not identified as objects. Note that the two arteries remain separated from the bone. Right: Scatter-plot of the computed classification.

5.2.3 Merging regions.

Region merging is done using the information of the classification step. All voxels which are marked as belonging to objects and which are adjacent to each other are merged into the same object. During this process information about the whole object is collected. The information items are the amount of voxels and the density value. This information is necessary for the labeling step. This merging process is also described in [7].

In the first step of algorithm bone-seg all objects have the same type: *NONE*. As no labeling has been done yet no other type may exist. During the second step of bone-seg neighboring objects with different types may exist. In this case it depends on the object types if a merging is possible or not. The different object types are described below. A summary of the merging rules can be seen in table 5.1.

		Object 1		
Object 2	type	NONE	BONE	VESSEL
	NONE	+	+	+
	BONE	+	+	-
	VESSEL	+	-	+

Table 5.1: Merging rules: + means merging is done, - means merging is not done.

5.2.4 Labeling and removing

The labeling process assigns to each object a unique type. Depending on properties like size and average density the decision on the type of an object is made. Three different types exist:

- *NONE*: Either the object was not labeled yet or the amount of voxels being part of the object is too low. Most common objects of type *NONE* are small fractions of vessels and bones as well as high intensity noise.

- *BONE*: Objects of the type *BONE* are removed from the dataset. These objects are meant to be bones. An object must have a certain size and the average density has to be above a certain threshold t_{label}.
- *VESSEL*: All objects which consists of enough voxels but the average density is below t_{label} are considered to be contrast enhanced arteries. Therefore these objects are not removed from the dataset.

Finally all voxels within objects of type *BONE* are removed from the dataset by setting the value of the voxel to zero.

5.2.5 User intervention

The basic workflow is that the tool first segments the whole dataset using predefined settings. No changes are permanent. At the end of the segmentation process the whole result is stored in a separate data-file. If the algorithm produces incorrect results, the user has to correct the result. Two types of user intervention are possible. First the set of parameters can be changed independently for each slice. Second the labeling of the final set of objects can be altered manually.

The different thresholds used in a slab are displayed in an overlay of the MIP visualization. It's up to the user to change these thresholds. This situation can be seen in figure 5.11. The leftmost line alters t_{label}. The light gray line in the middle sets the t_{expand} threshold. Finally the rightmost line changes the threshold t_{class}. The two rightmost lines are in the same value domain.

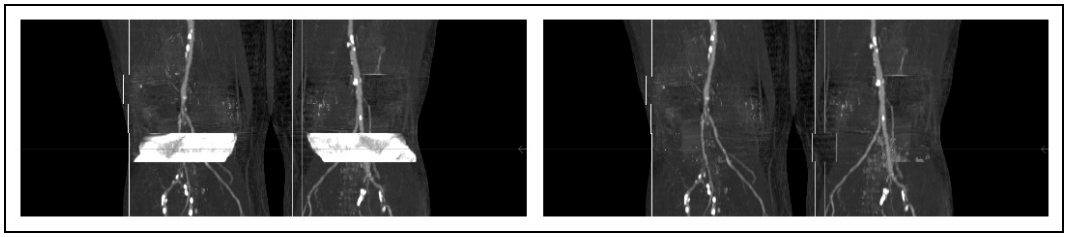


Figure 5.11: Left: Suggestion of the algorithm. Right: Correction of the user by changing t_{expand} and t_{class}.

If an object is not labeled correctly the user is able to select this object. In figure 5.12 such a situation is shown. In the left column a section of the slice window is presented. The right column shows a section of the MIP window. In the first row the wrongly segmented artery is clearly visible. Starting the manual labeling feature shows the objects color encoded as an overlay in the slice window. Clicking on the objects using the mouse cursor alters the type of the object. The white arrow visualizes the mouse cursor. After this step the type of the object changes, which can be seen immediately in the MIP window. This situation is shown in the next row of figure 5.12. Closing the manual labeling feature shows the changes in full quality (final row).

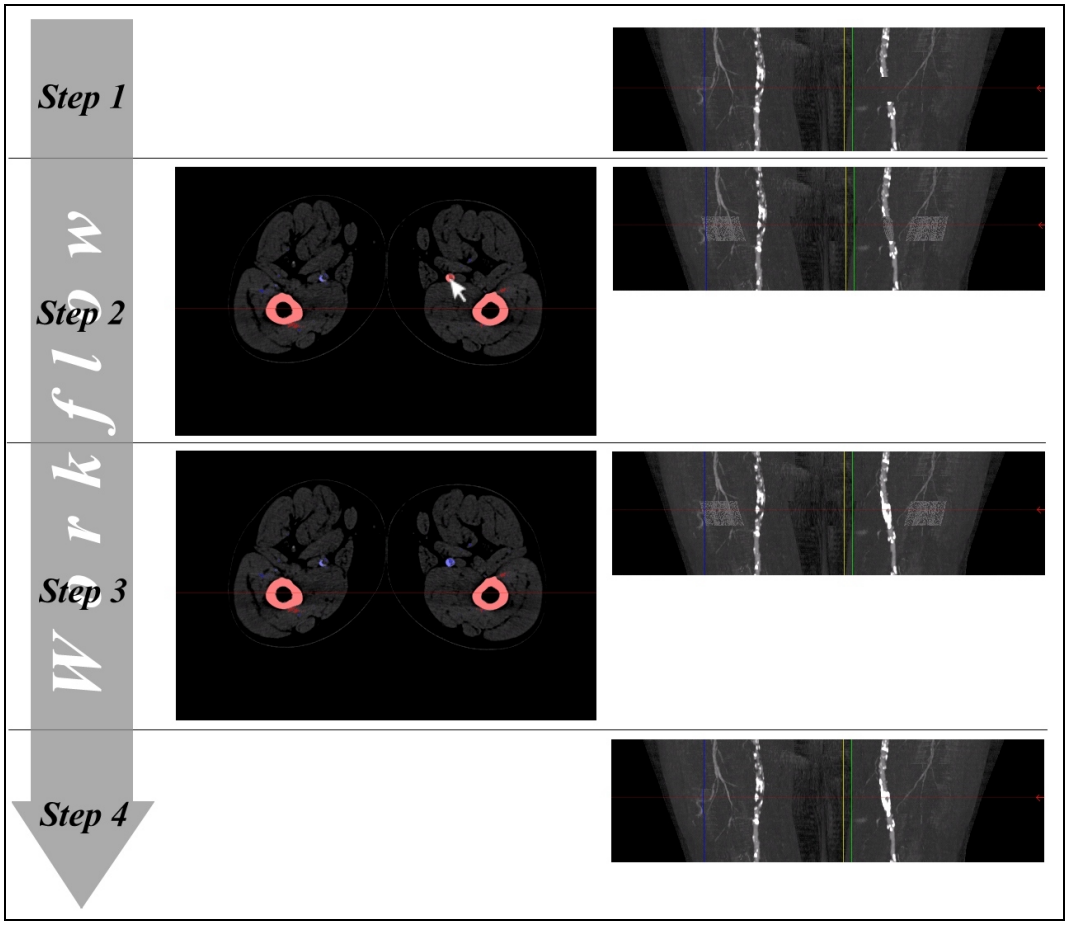


Figure 5.12: Workflow of manual labeling. Left column: slicing window. Right column: MIP. The incorrectly labeled object is corrected using the mouse as input device (white arrow).

5.2.6 Conclusions

Using bone segmentation as pre-processing step and MIP or direct volume rendering as a visualization technique produces quite good results (please refer to the results in chapter 7). The reason is that the vessel tree is visible in its entirety. Radiologists prefer this kind of visualization because the spatial relation is clearly visible and the collateral vessels are also displayed. Especially the second reason is important for investigation purposes as mentioned above. However this technique suffers from two minor drawbacks:

- *Safety and Correctness*: It is by no means guaranteed that vessels are not removed accidentally. Especially big vacant calcifications without connection to the rest of the vessels may be removed incorrectly. The disconnection to the rest of the vessel tree may result from occlusions. This drawback can easily be overcome by manual user interaction, after the obvious incorrectness is identified.
- *Lack of intuitive user intervention*: As this is not a fully automatic method, the user has to be trained to handle the specification of the algorithms parameters. After a training period, the system however can be used intuitively and efficiently. Yet an untrained user will not be able to produce results of any diagnostical value.

Chapter 6

Implementation

The following sections describe the implementation of the algorithms discussed in chapter 5. C++ was the programming language of choice. The development environment consisted of Microsoft Visual Studio 6.0, the graphics interface OpenGL in version 1.2.1 and the OpenGL utility GLUT 3.0. As C++ is an object-oriented language, the code is mainly written in object oriented C. A selection of the most important objects is presented below. Only the most important information is shown for each object. Type identifiers are abbreviated as shown in table 6.1.

i	integer	ui	unsigned integer
s	short	us	unsigned short
c	char	uc	unsigned char

Table 6.1: Abbreviations of the used types.

6.1 CData

The object CData encapsulates the dataset. CData provides a set of access methods to the dataset.

```
class CData
{
    us xSize;
    us ySize;
    us zSize;

    void GetSlab(i slab, i length, i size, us* buff);
    void GetCacheBlock(i hx, i hy, i hz, us* buff);

    CData(c* filename);
    CData(us* daten, us xsize, us ysize, us zsize);
};
```

This object contains the information like the dimensions of the dataset and the management of the raw data. Concerning the algorithms the most important access methods are GetSlab() and GetCacheBlock(). The first one fills *buff* with the data of the slab number *slab*. GetCacheBlock() delivers the data of the cache block defined by *hx*, *hy* and *hz* in *buff*. As the datasets are very large it is not efficient to allocate memory for the whole dataset in main memory. Therefore the dataset is split into so called cache blocks of constant size. The object uses either a filename or a data array with dimension information as construction information. If a filename is given the file must first contain the dimension information and the raw data.

6.2 CPathfinder

```
typedef struct TListstruct
{
    TElement* head;
} TList;

typedef struct TElementstruct
{
    us x, y, z;
    i value;
    struct TElementsrtuct* next;
    struct TElementstruct* before;
} TElement;

class CPathfinder
{
    TList queue[C];
    i queue_pointer;

    int SearchPaths(CPunkte* pkt);
    void CalculatePaths(CPfad* pfds, CPunkte* pkt);

    CPathfinder(CDaten *daten)
};
```

CPathfinder is created with a CData object as constructor parameter. According to the shortest path finding algorithm described in chapter 5 the *SearchPaths()* method takes a set of points as input. The output is kept in the direction information arrays. This method searches the shortest paths from the starting point to the endpoints.

As proposed in [13] a circular queue *queue[C]* of size *C* is used in order to store all vertices that are currently in process (the set *Q* in section 5.1.5). Each item of this queue is a pointer to a double-linked list consisting of elements of the type TElement. A queue pointer *queue_pointer* is initially set to 0.

All values gained from the local cost function introduced in chapter 5 are calculated as integer numbers. For this reason the accumulated cost of each voxel can be stored in an integer type. The voxel *v* is inserted in the queue at the position *cc(v)* modulo *C*. Where *cc(v)* is the accumulated cost of *v*. The value of the local cost function has to be less than *C*. When searching for the voxel with the least cumulative cost the first voxel is taken from the list *queue_pointer* is pointing to. If the list is empty *queue_pointer* is incremented modulo *C*. As all voxels within a list have the same accumulated cost and the value of the cost function is always positive and smaller than *C*, this method yields the voxels with the smallest accumulated cost.

The complexity of inserting a voxel *v* into this list is O(1). As all voxels within the list have the same accumulated cost the new voxel *v* can be inserted at first position for instance.

The complexity of removing a voxel *v* from a double-linked list is O(1).

The worst-case complexity of finding the next voxels with least accumulated cost is O(*C*-1). This is very unlikely, as the local cost function for each vessel has to be of value *C*-1. The number of queue pointer increments to find a new non-empty list is usually less than 1% of *C* [13].

As each voxel is processed exactly once (in the moment when this voxel has the minimum cumulated cost) the worst case complexity of the algorithm is O(*nC*) where *n* is the amount of voxels in the dataset.

CalculatePaths() makes use of the direction information generated in the *SearchPaths()* function to build up the paths as described in chapter 5. The direction information is stored in a single byte. As each voxel starting from the end points determines the next voxel until the starting point is reached this operation is of complexity of O(*l*) for each path *P* with path length *l*.

6.3 CBSpline

```
class CBSpline
{
    void GetNormalVector(i index, TFVector* vkt);

    CBSpline(CPfad* pfade, i which, i k);
};
```

The constructor of *CBSpline* takes a *CPfad* object *pfade*, containing the paths. An integer value *which* defines from which path inside *pfade* the B-spline curve should be constructed. Finally *k* defines the order of the B-spline curve.
The function *GetNormalVector()* returns the tangent vector from the B-spline curve at position t = *index* and returns the information in the structure *vkt*.

6.4 CNormalEbene

```
class CNormalEbene
{
    void GetPlane(i pkt, i xDiam, i yDiam, us* ebene);

    CNormalEbene(CDaten* d, CBSpline* bs,);
};
```

CNormalEbene provides a method which returns in *ebene* a plane perpendicular to the B-spline functon *bs* through control vertex *pkt* with a dimension of *xDiam* to *yDiam*. For this reason the constructor needs data access through *d* and the B-spline function *bs*.

6.5 CCenterFinder

```
class CCenterFinder
{
    void GetCorrectedPath(CPfad* paths, i welcher);
    CCenterFinder(CNormalEbene* nEbene);
};
```

The object *CCenterFinder* takes a *CNormalEbene* object as input for the constructor. *GetCorrectedPath()* corrects the path defined by *nEbene* and stores it in *paths* at position *welcher*.

6.6 CInteractiveSegmentation

```
class CInteractiveSegmentation
{
    us *threshold_high;            // t_class
    us *threshold_low;             // t_expand
    float threshold_avgdensity;    // t_label
    FILE** allready_segmented;

    void WorkSlab(i slab);
    void RemoveSlab(i slab);

    void SegmentSlabStart(i slab);
    void SegmentSlabFinish(i slab);
    void SegmentSlab(i slab);
    void SaveAllSlabs();

    CInteractiveSegmentation(CDaten* d);
};
```

The *CInteractiveSegmentation* object needs to have access to the data, therefore a *CDaten* object is given at creation time.

The arrays *threshold_high* (t_{class}), *threshold_low* (t_{expand}) and *threshold_avgdensity* (t_{label}) store the slab dependent segmentation parameters as introduced in chapter 5. If a user intervention was necessary, a temporary file is generated and inserted in the array *already_segmented* according to the specific slab.

SegmentSlab() segments a slab *i* according to the parameter for the slab *i* in the arrays mentioned above. Among other functions this method is based on *WorkSlab()* and *RemoveSlab()*.

The methods *SegmentSlabStart()* and *SegmentSlabFinish()* implement the user interactive labelling feature. Between these two functions the user may do manual labeling. *SegmentSlabStart()* calls *WorkSlab()* and *SegmentSlabFinish()* calls *RemoveSlab()*.

The *WorkSlab()* does classification, merging and labeling as described in chapter 5. *RemoveSlab()* removes objects labeled as bones from the slab.

If no manual labeling was done the segmentation was only based on the parameters corresponding to each slab. The segmentation step can be reproduced for these slabs. Therefore only the maximum intensity projection visualization is altered. If manual labeling was done the segmented dataset is stored in a temporary file. This method makes an undo functionality possible without decreasing the speed when adjusting only the parameters of the slabs. According to these constraints the *SaveAllSlabs()* method either takes the information of each slab from the according temporary file or restarts the segmentation process for the slab. The processed slabs are stored in a different file.

Chapter 7

Results

This chapter presents the results of the algorithms proposed in this thesis. Three datasets were selected in order to maintain diversity in respect to the dataset size and the types of arterial diseases. The test environment consists of a PII 350 MHz system with 704 MB main memory, running Windows NT 4.0 SR 5 as operating system. The volume rendering was done on the commercial medical image processing system JVision/Space-Vision from TIANI Medgraph [25].

7.1 The test datasets

Table 7.1 gives an overview of the datasets parameters. All datasets are real world datasets. These datasets were anonymized as patient information was deleted.

Name	Spatial resolution	Size in MB	Volume size (mm x mm x mm)
100_198.dat	512 x 512 x 988	494,5	257 x 257 x 1070
004_old.dat	512 x 512 x 550	275,5	240 x 240 x 1102
022_old.dat	512 x 512 x 1000	500	260 x 260 x 1100

Table 7.1: Parameters of the presented datasets

In table 7.2 the time needed for the investigation steps is summarized. As the segmentation process is a highly interactive process no division between user interaction time and computation time is made.

Name	CPR computation time	CPR user interaction time	CPR center finding time	Segmentation time
100_198.dat	16 min 20 s	1 min 30 s	1 min 14 s	25 min 41 s
004_old.dat	8 min 10 s	2 min 10 s	28 s	31 min 25 s
022_old.dat	15 min 40 s	1 min 20	29 s	18 min 11 s

Table 7.2: Investigation time of the datasets

7.2 Dataset 100_198.dat

The patient this dataset has been acquired from suffers from a serious occlusion in the left iliac artery. Several high-grade stenoses caused by soft plaque and serious calcifications can also be found in this dataset (see figure 7.1).

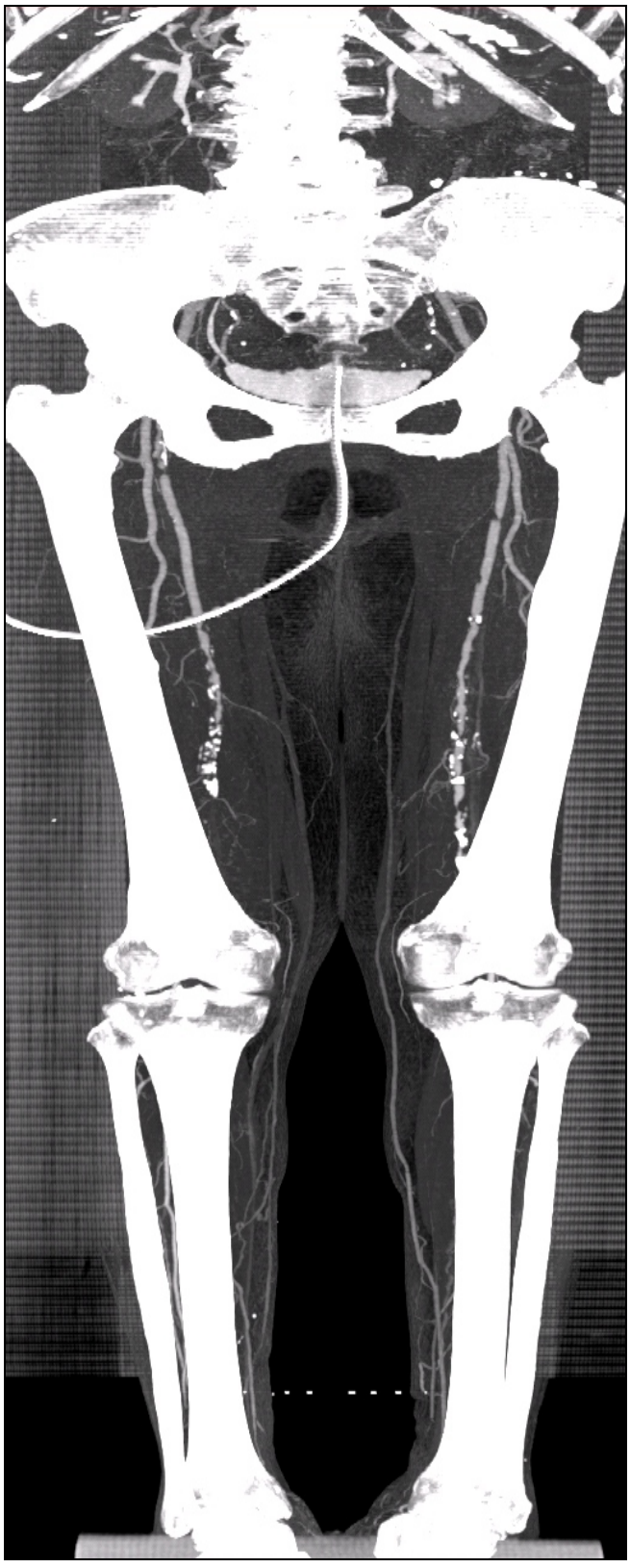


Figure 7.1: MIP of the dataset 100_198.

7.2.1 Curved planar reformation

Different vessels are selected for curved planar reformation. The six endpoints can are shown in figure 7.2. The paths are numbered from left to right. The leftmost shorter path is number 1. The rightmost path is referred to as number 6.

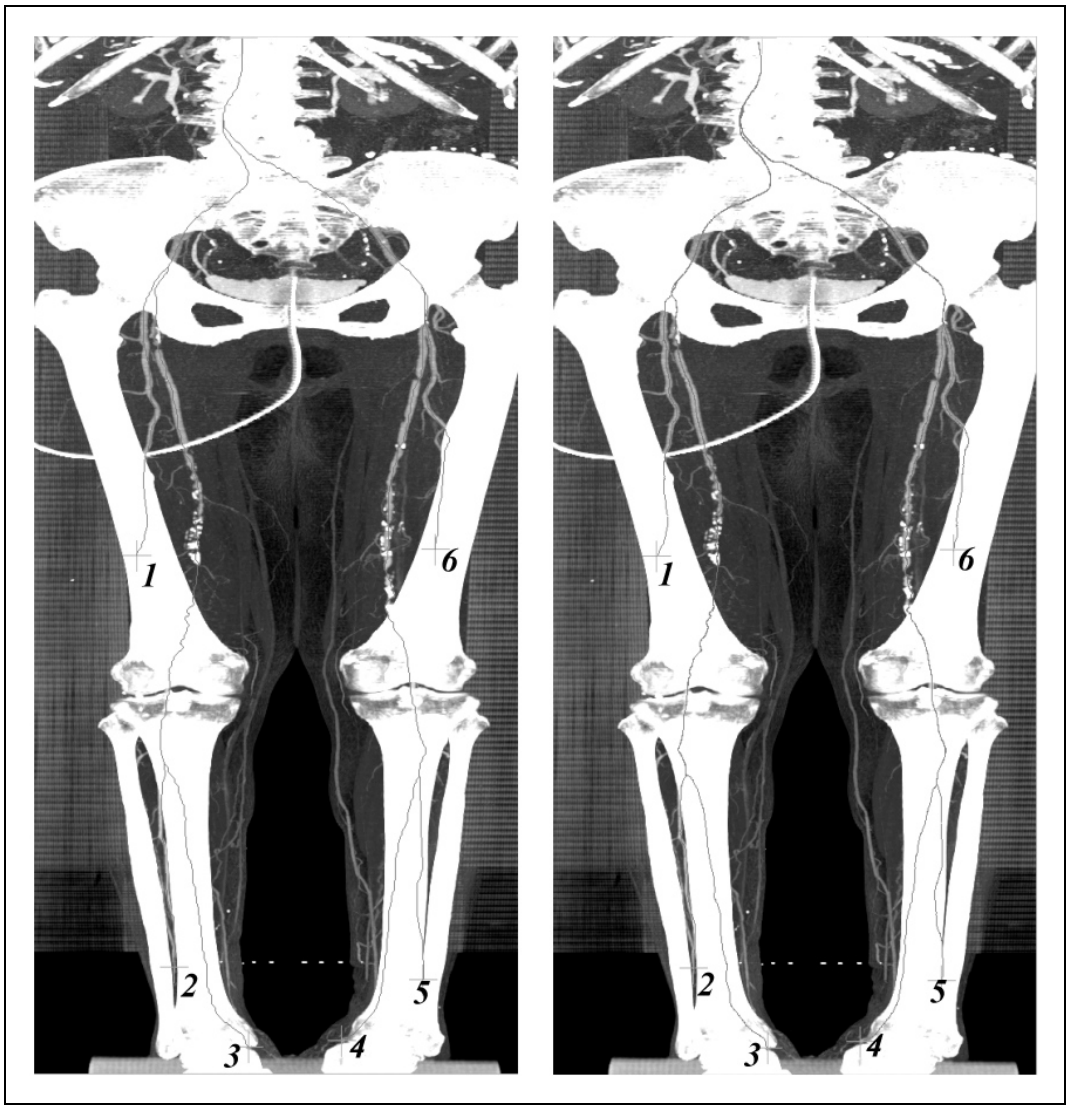


Figure 7.2: Left: Calculated paths. Right: Centered paths.

In the figures 7.3 to 7.8 the curved planar reformation with the original path is shown in the left image. The right image shows the curved planar reformation with the centered path.

Figure 7.3: CPR of path number 1.

Figure 7.4: CPR of path number 2.

Figure 7.5: CPR of path number 3.

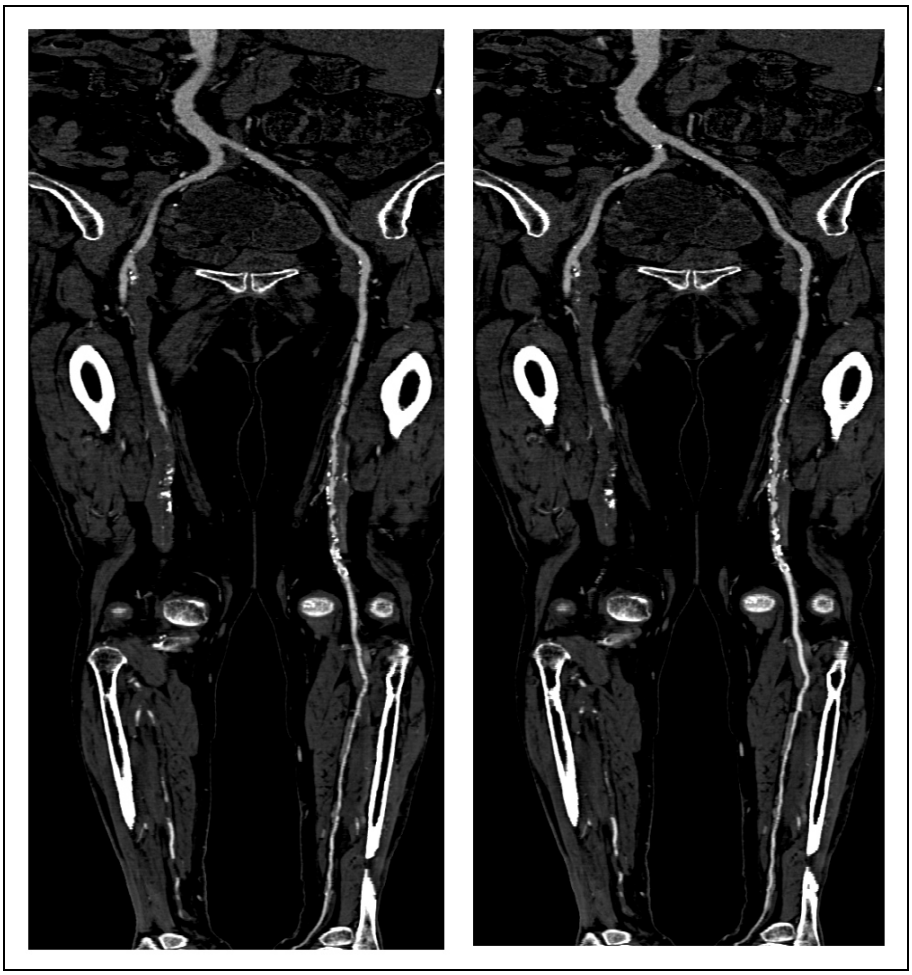


Figure 7.6: CPR of path number 4.

Figure 7.7: CPR of path number 5.

Figure 7.8: CPR of path number 6.

7.2.2 Segmentation results

This section presents the maximum intensity projection (MIP) of the segmented dataset 100_198.dat (see figure 7.9). Also images of direct volume rendering with this dataset are shown in figures 7.10 and 7.11. The color encoding is *red* for contrast-enhanced vessels, *white* for calcifications and *dark cyan* for body and bone contours.

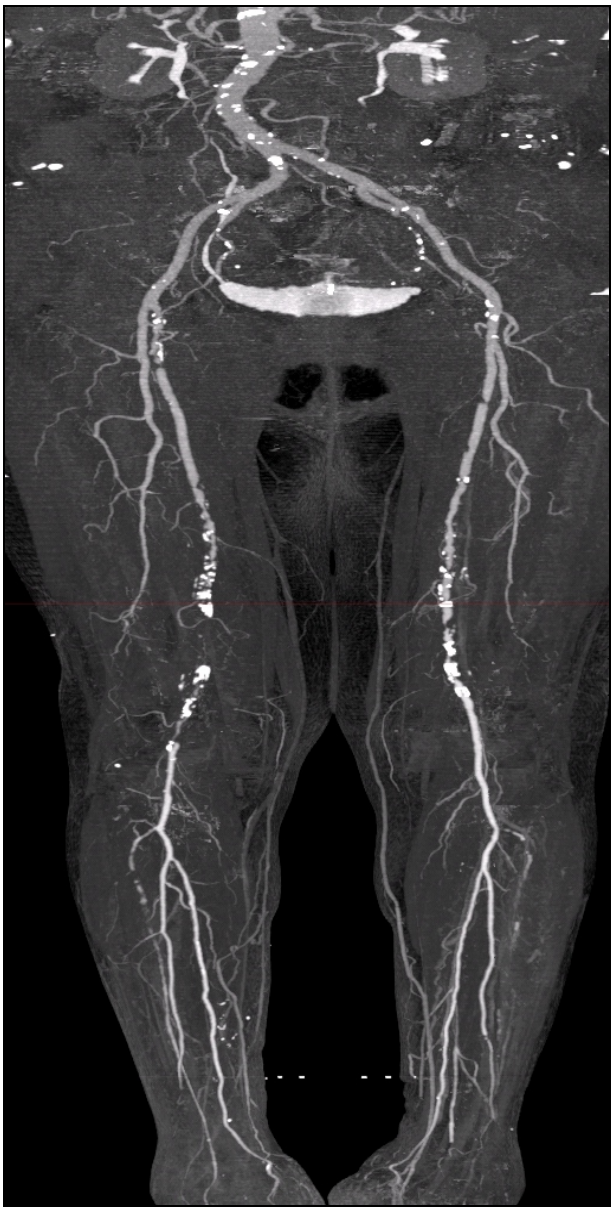


Figure 7.9: MIP of the segmented dataset 100_198.dat.

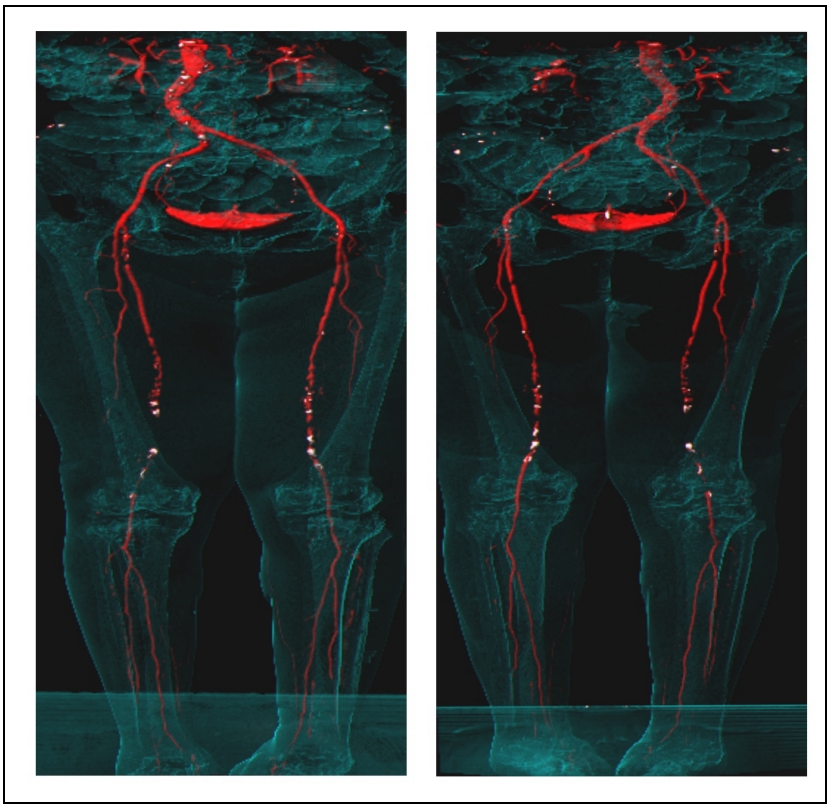


Figure 7.10: The front side (left) and the back side (right) of the dataset.

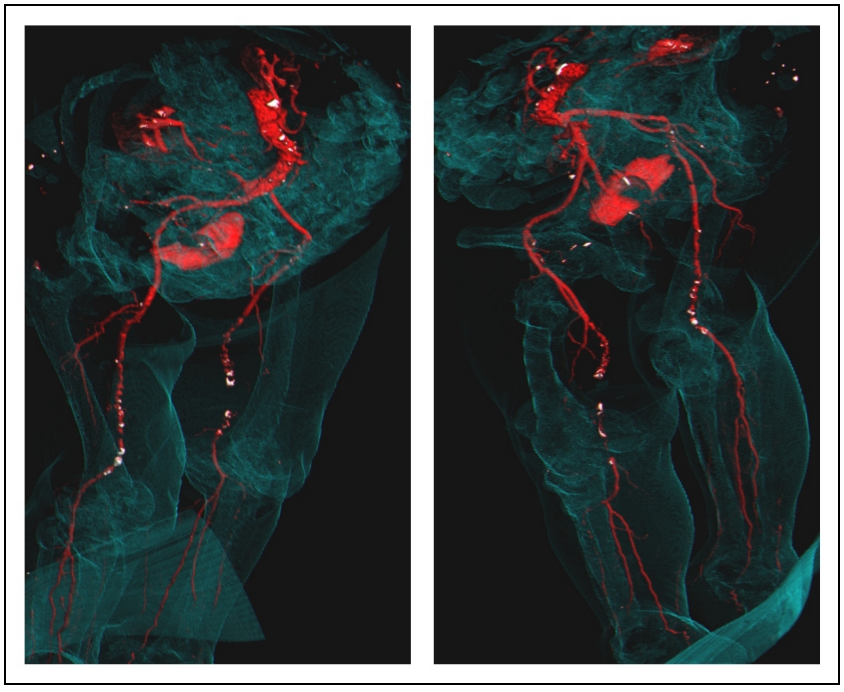


Figure 7.11: The dataset from two arbitrary viewpoints.

7.3 Dataset 004_old.dat

The patient of dataset 004_old.dat suffers from serious calcifications and occlusions. The bypass in the left iliac artery is an indication of surgical intervention. It is very likely that such high-grade arterial diseases lead to a symptomatic disease (see figure 7.12).

Figure 7.12: MIP of dataset 004_old.dat.

7.3.1 Curved planar reformation

The five paths are numbered according to their endpoints from left to right. In the following CPR based pictures the non-centered paths are shown on the left hand side. The centered path reconstruction is shown in the right image of each figure.

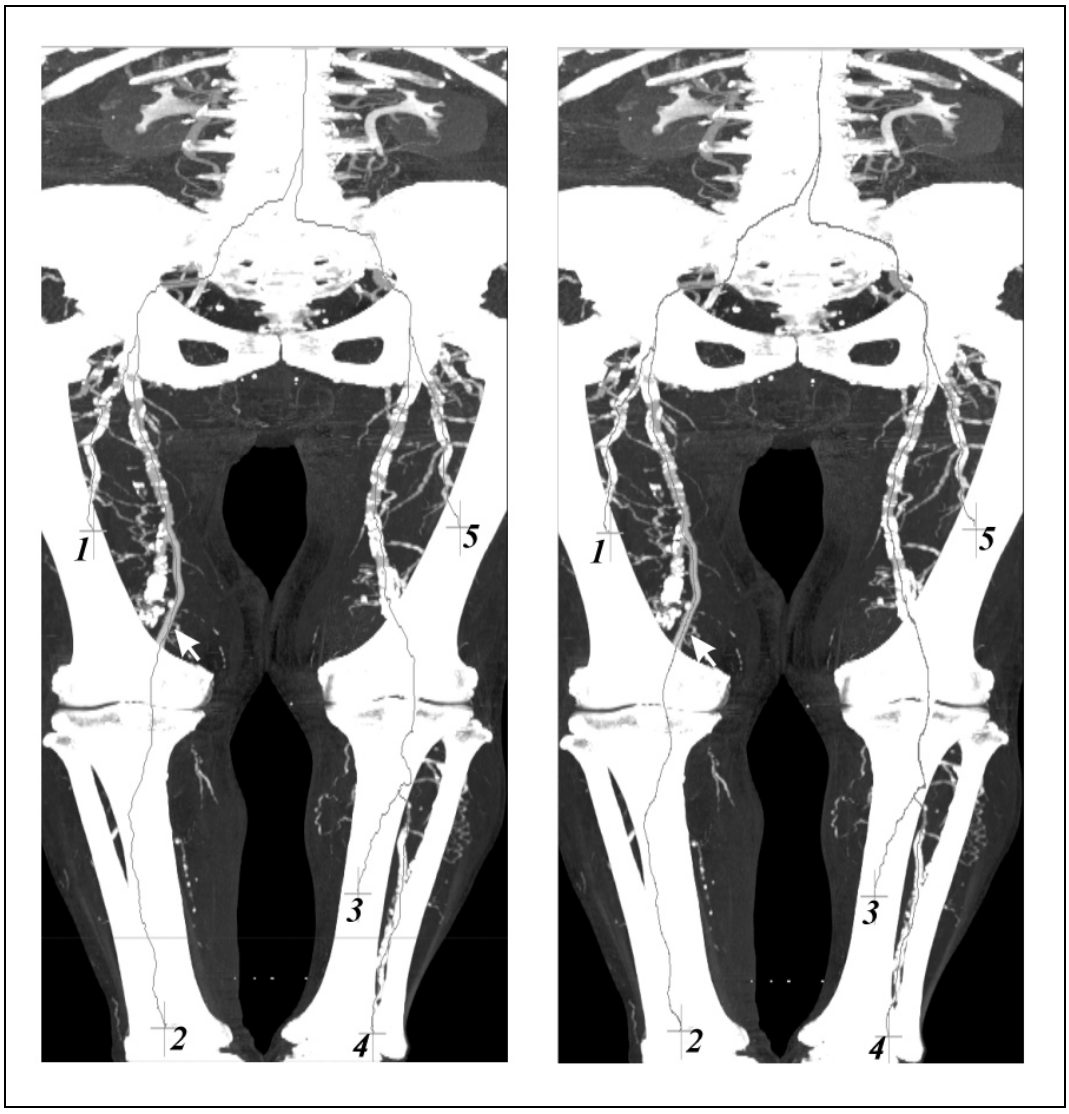


Figure 7.13: Left: Calculated path. Right: Centered path. The white arrow marks a region where the center-finder algorithm improves the image significantly.

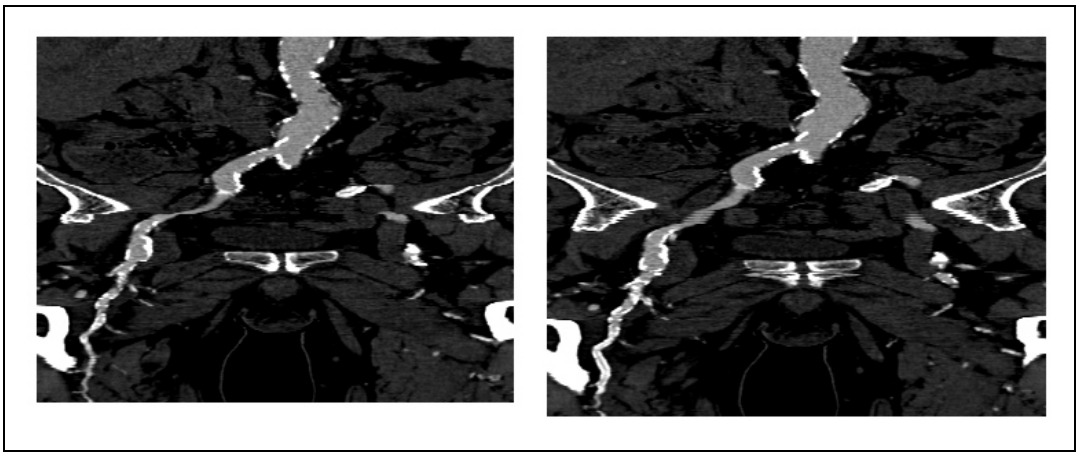


Figure 7.14: CPR of path number 1.

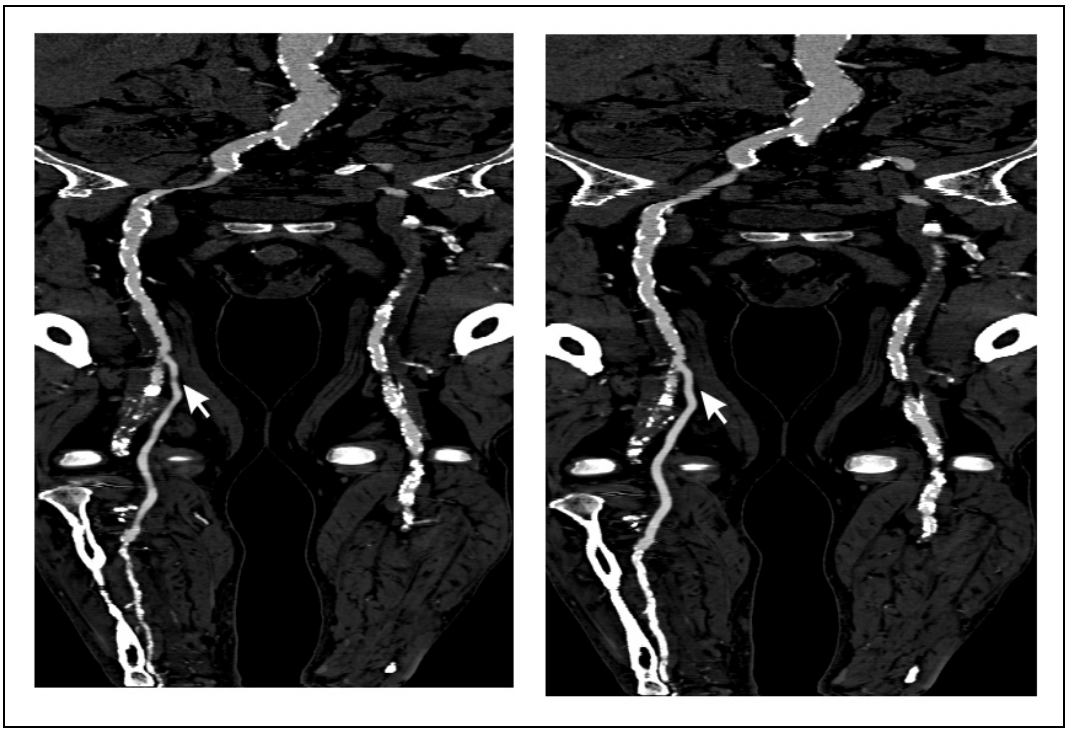


Figure 7.15: CPR of path number 2. A wrong stenosis is pointed out by the arrow in the left image. The improvement by centering the vessel is clearly visible in the corresponding region of the right image.

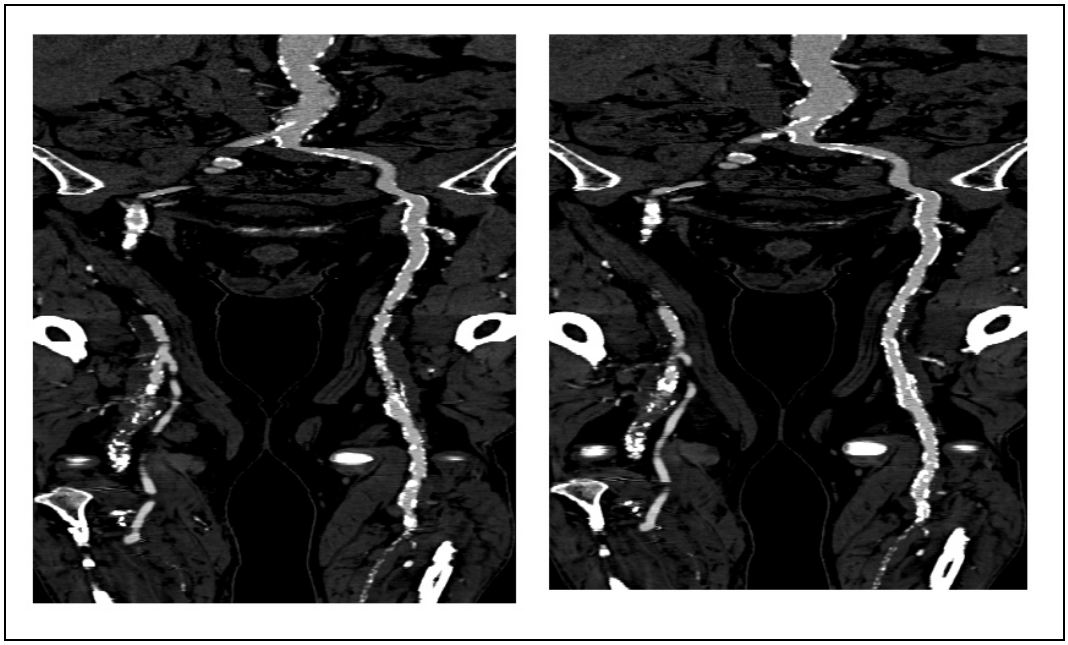


Figure 7.16: CPR of path number 3.

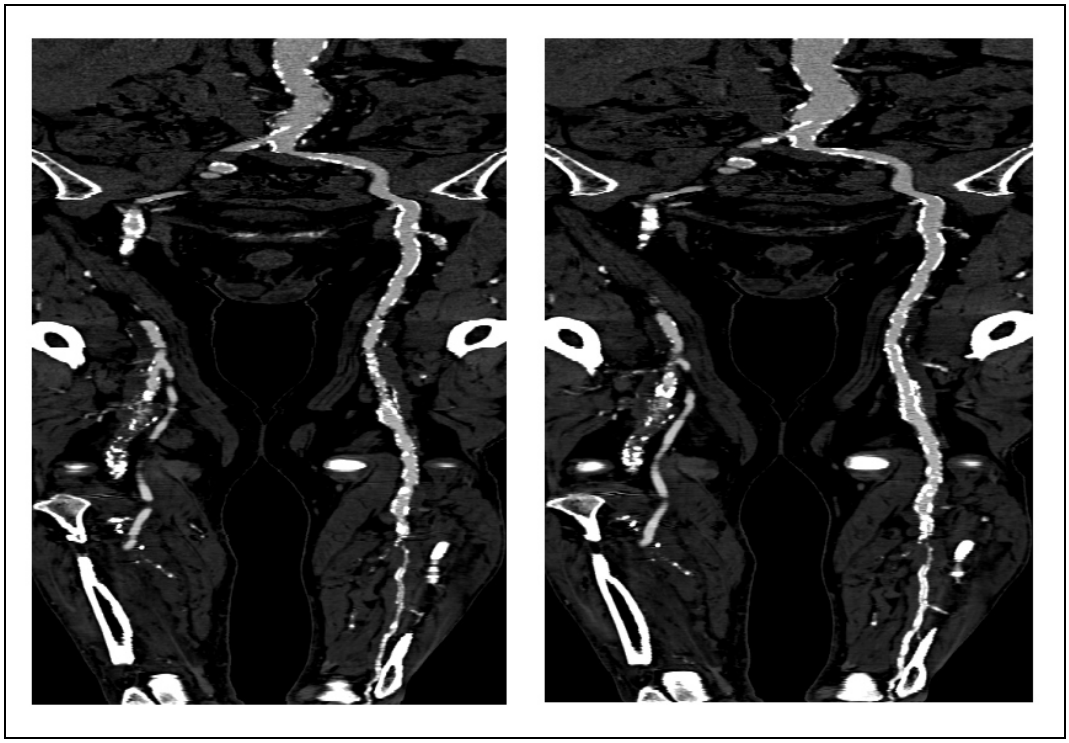


Figure 7.17: CPR of path number 4.

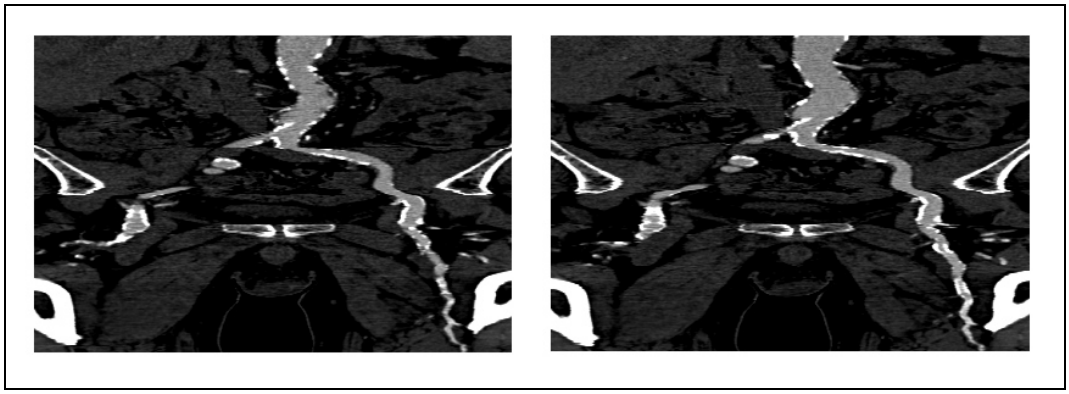


Figure 7.18: CPR of path number 5.

7.3.2 Segmentation results

As can be seen in figure 7.19 in the topmost area the dataset causes some difficulties. Therefore more noise than elsewhere in the dataset remains after segmentation.

Figure 7.19: MIP of the segmented dataset 004_old.dat

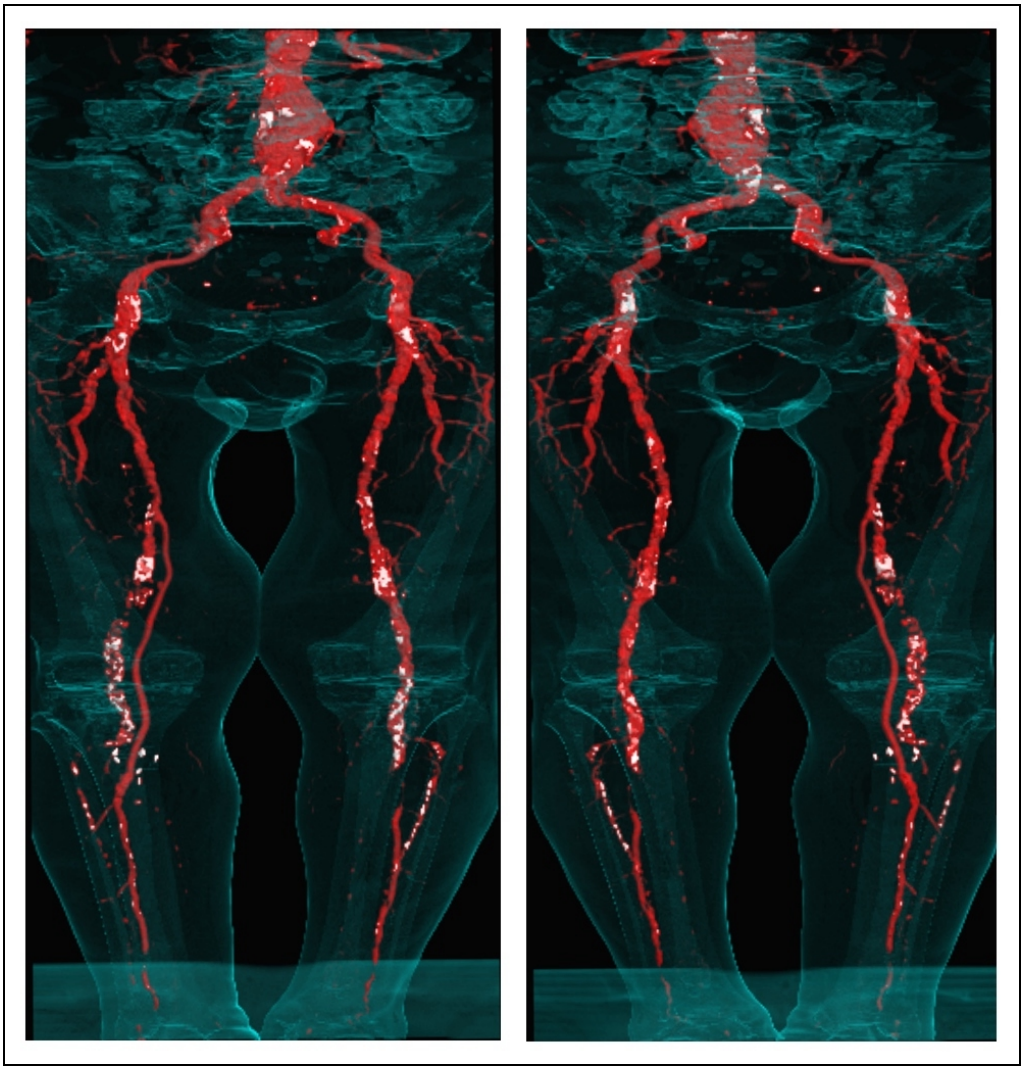


Figure 7.20: Front view (left) and back view (right) of the dataset.

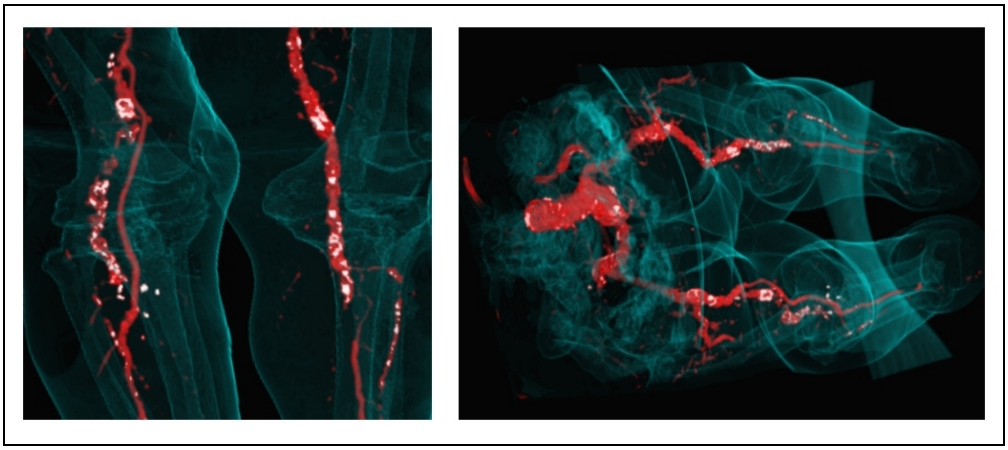


Figure 7.21: Detail inspection and an arbitrary viewpoint of the dataset.

7.4 Dataset 022_old.dat

Dataset 022_old.dat is a very interesting dataset. Even though no major calcifications exist, serious occlusions occur. These long occlusions are caused by soft plaque. This dataset is a challenge for every algorithm trying to identify vessels.

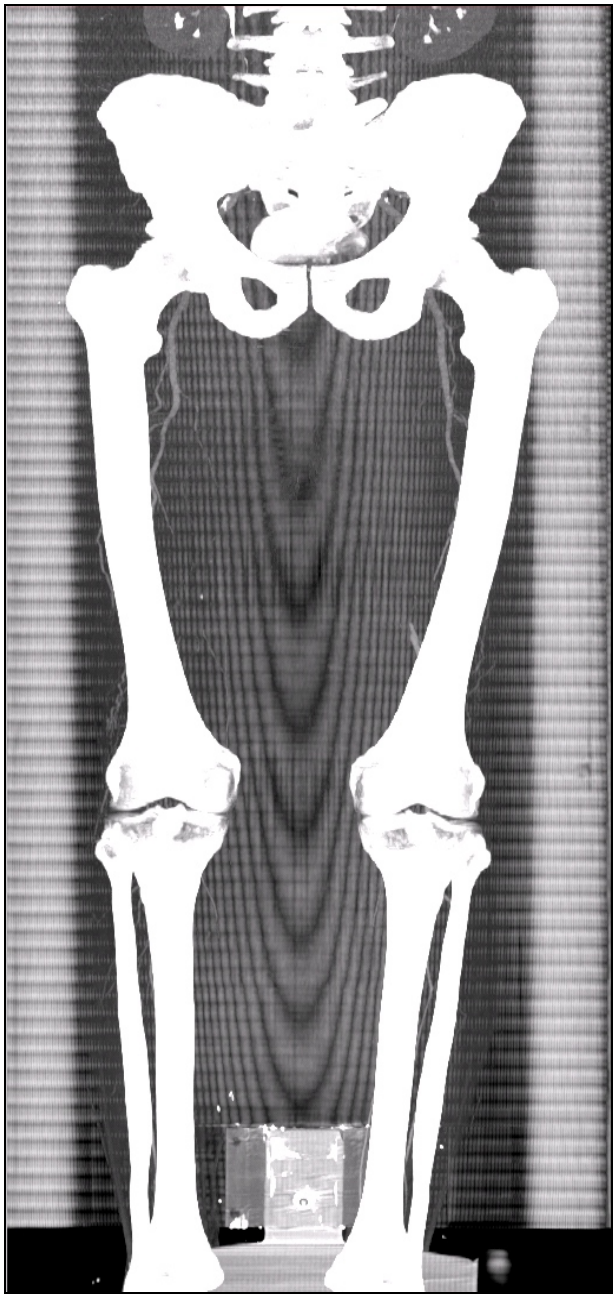


Figure 7.22: MIP of the dataset 022_old.dat

7.4.1 Curved planar reformation

On this challenging dataset the high reliability of the path-finder algorithm is illustrated. Even as no contrast agent is present inside the vessel over long distances the path-finder correctly tracks the vessel. Note how the algorithm finds the connection through collateral vessels (see figure 7.23).

Figure 7.23: Left: Calculated path. Right: Centered path. The black arrow marks a region where the path-finder algorithm finds a connection through collateral vessels.

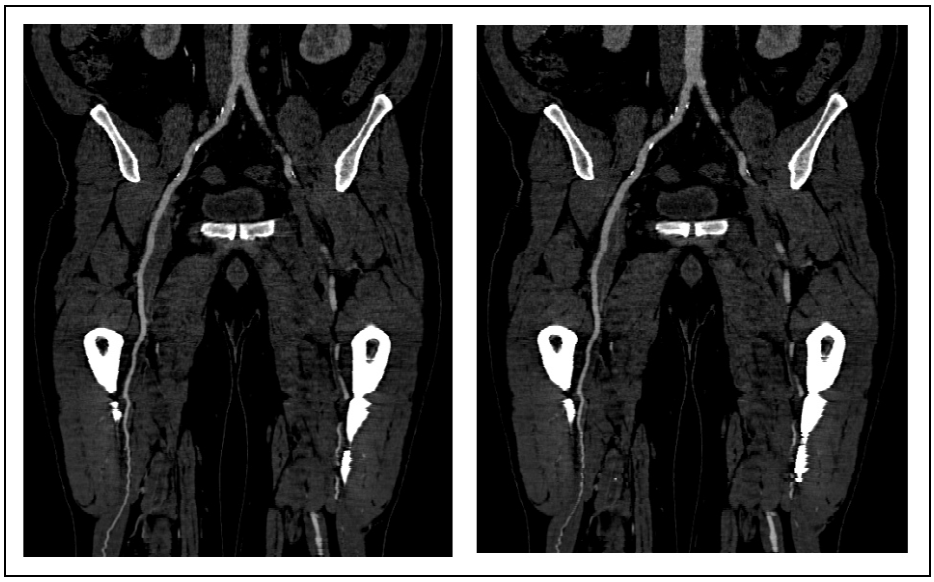


Figure 7.24: CPR of path number 1.

Figure 7.25: CPR of path number 2.

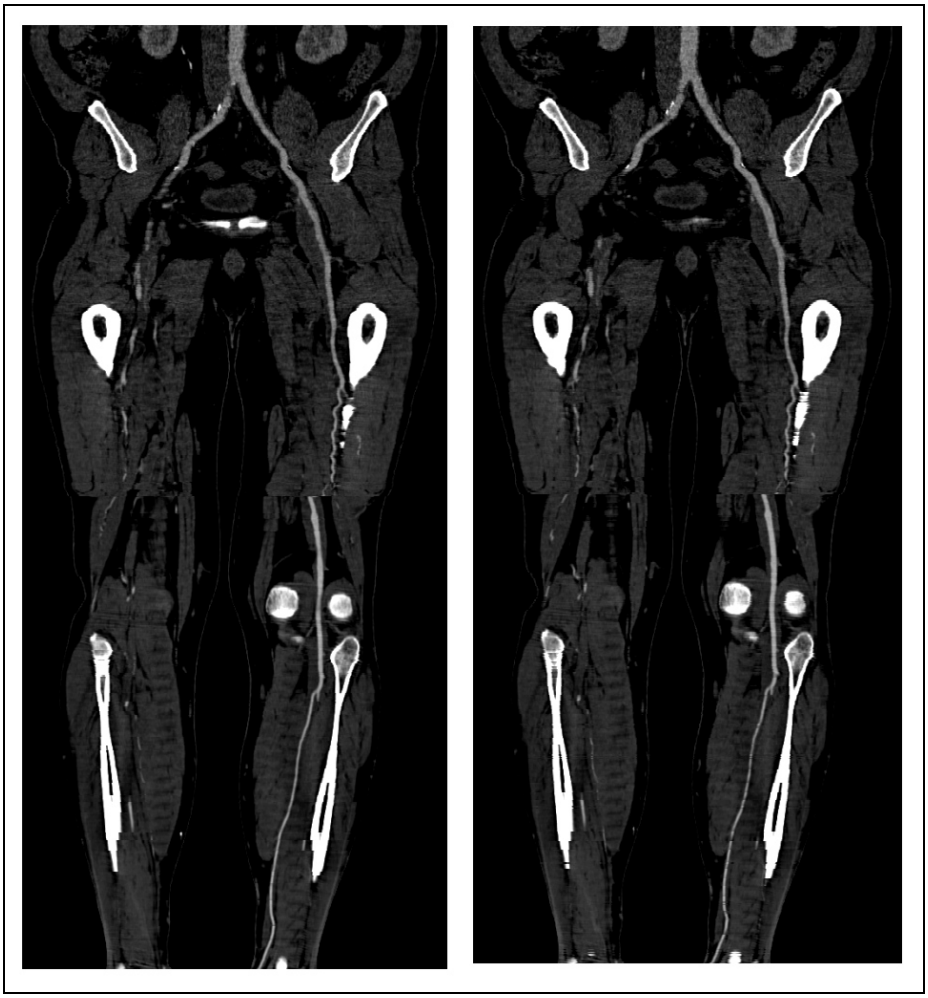


Figure 7.26: CPR of path number 3.

Figure 7.27: CPR of path number 4.

7.4.2 Segmentation results

The very long occlusions are clearly visible in the maximum intensity projection of the segmented dataset. The extended collateral vessels in the area of the end of the occlusions are also very good visible. These are the areas where the artery seems to start again (see figure 7.28).

Figure 7.28: MIP of the segmented dataset.

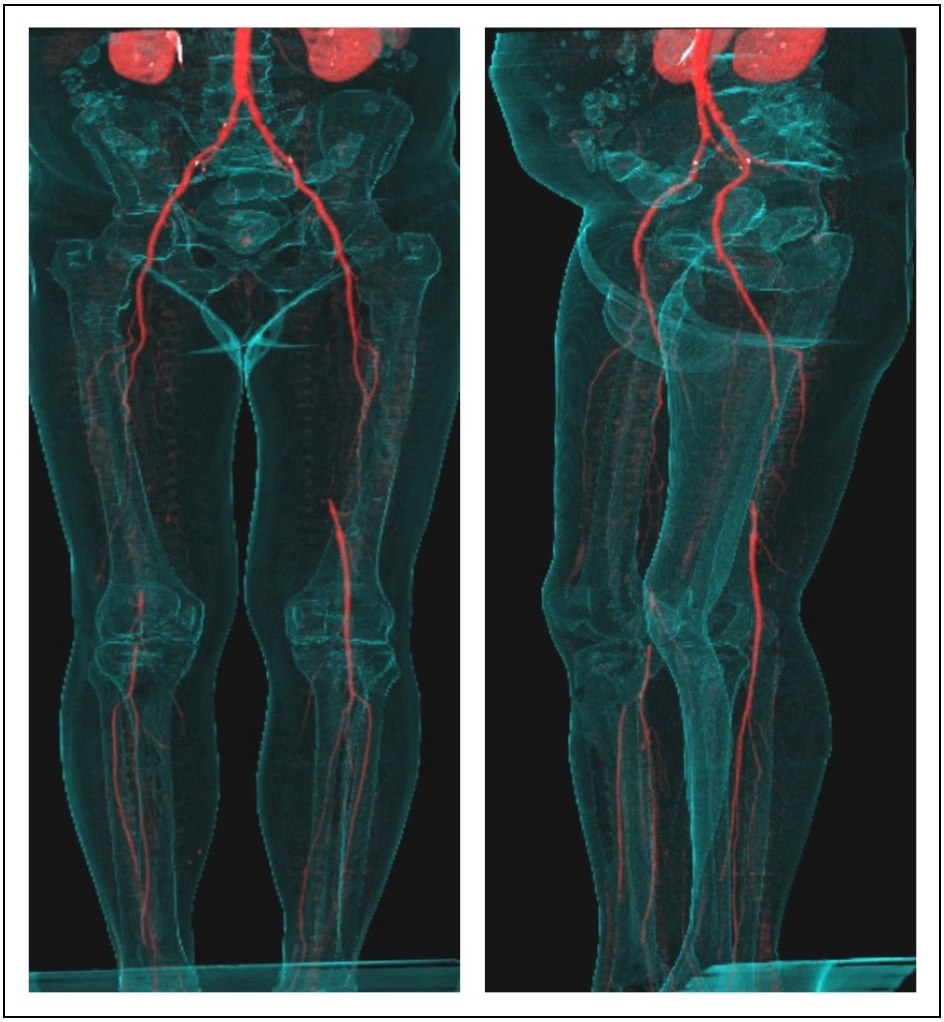


Figure 7.29: Front view (left) and side view (right) of the dataset.

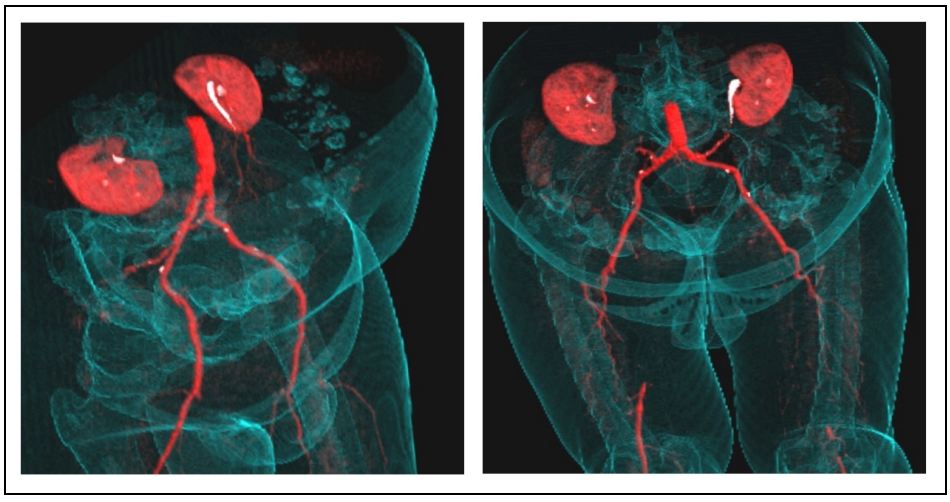


Figure 7.30: Two different views of the dataset

Chapter 8

Conclusion

The goal of this explorative thesis is to find appropriate techniques for vessel investigation of lower extremities and to prove that faster investigation is possible. Both targets were addressed and different solutions were proposed. All solutions were approximately 8 times faster as the methods currently in use.

Different visualization techniques and segmentation methods of computed tomograpy angiography datasets have been investigated. A global optimisation method for the computation of a curved line was chosen because of high reliability. The curved plane reformation based on the semi-automatic computed curved-line was considered as an appropriate visualization technique as this method is known well in the radiology community. Therefore high acceptance of this technique within clinical environments is expected. An enhanced user interactive segmentation method focuses on bone removal. The advantage of this technique is the well-balanced trade-off between high quality segmentation and time requirements.

Chapter 9

Summary

The following chapter provides a summary of the main topics mentioned above. As only the main features and algorithm basics are addressed this chapter provides an overview of the thesis.

9.1 Abstract

Multi-slice helical computed tomography has the potential to accurately show the entirety of the lower extremity vessels with a single intravenous contrast medium injection at unprecedented, near isotropic spatial resolution. Large data size and the visibility of bony structures makes data set post-processing after data acquisition necessary. Two different visualization and segmentation methods are presented: A global optimization method for the computation of a curved planar reformation is introduced. Secondly a user-interactive segmentation approach is proposed.

9.2 Introduction

Medical image analysis is a rapidly growing field in information technology. Especially in the field of computed tomography new areas of application arise due

to the progress in computed tomography modalities. One of such applications is computed tomography angiography (CTA) of peripheral vascular diseases. Latest technical developments in CT – notably multi-slice helical CT – allow an approximately three-fold increase of volume coverage, while maintaining longitudinal resolution, and without incurring more helical artifacts.

Lower extremity arterial disease is a significant health problem in the industrial world. Today, intraarterial digital subtraction angiography (iaDSA) is the pretherapeutic imaging technique of choice. iaDSA, however, is an invasive and costly procedure, which requires arterial catheterisation. A non-invasive technique - as CTA - for imaging the entire inflow and runoff vessels is therefore highly desirable.

In order to visualize the entire arterial system of the lower-limb vessels with CTA, a stack of approximately 900 to 1500 transversal images have to be reconstructed per patient. Currently available editing and rendering techniques require more than 4 hours of manual editing and interaction of a well-trained user (radiologist or technologist) to obtain images that contain diagnostical value. Compared to the acquisition time of 25 to 60 seconds it is obvious that post-processing time has to be reduced. In contrast to MRA or iaDSA, CTA requires some sort of post-processing, as bony structures are visible in the patient's dataset. These bony structures prevent the vessel-tree from being entirely visible.

Basically two different approaches were followed in order to provide a feasible tool for investigating CTA-datasets. The first is to generate a curved planar reformation (CPR) as described in section 9.4.

The second approach is quite different. As the vessel tree in lower extremity areas consists of a huge amount of blood vessels of all sizes it is very difficult to identify every single vessel. The basic idea is that if it is difficult to identify the structures of interest, it might be easier to hide structures of less or no importance. Following this approach the whole vessel tree can be made visible by removing the bones from the dataset.

9.3 Vascular diseases

The arterial tree of interest which supplies blood to the legs includes the abdominal aorta, the pelvic arteries and the arteries of both legs. The entire vascular tree including the abdominal aorta is shown in figure 9.1.

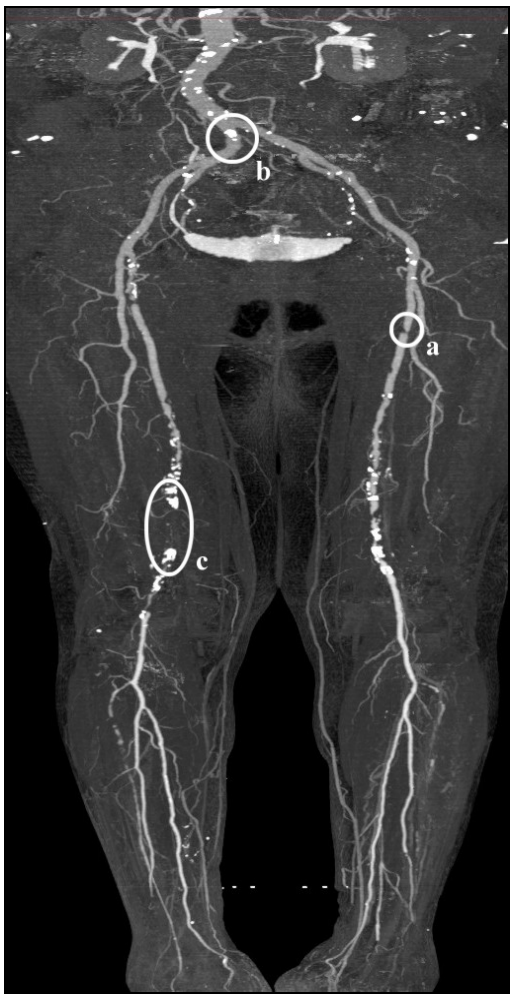


Figure 9.1: The vessel tree of lower extremities. *a*) a short, segmental stenosis in the femoral artery. *b*) calcification near the aortic bifurcation. *c*) a long occlusion in the femoro-popliteal artery.

The most relevant arterial abnormalities are:

- *Stenosis*: A stenosis is a narrowing of the arterial flow lumen. Arterial stenoses are caused by atherosclerotic *plaque* (as can be seen in figure 9.1*a*). Atherosclerotic plaques are soft tissue density lesions within the

vessel wall, but plaques may also calcify. Atherosclerotic plaques often occur near the areas of high turbulences, such as bifurcations and bends. A *bifurcation* is for example the area where the aorta branches into the iliac (pelvic) arteries.

- *Occlusion*: A complete obstruction of a vessel is referred to as an occlusion. The blood flow is redirected through secondary vessels, which circumvent the occluded vascular segment, and which are called collateral vessels. An example of an arterial occlusion is shown in figure 9.1*c*.
- *Calcification*: The vessel wall of diseased arteries, as well as atherosclerotic plaque may calcify. With CT, calcified tissue is of high attenuation. In figure 9.1 several areas of calcification can be seen. One is marked with circle *b*.

9.4 Semi-automated curved planar reformation

A *curved plane* in 3D space is described by a curved line and a vector. For each point of the curved line a straight line is defined which is collinear to this vector. The result is a curved plane in 3D space. Figure 9.2 shows a curved line on the left side. On the right side a curved plane is constructed using a vector that is parallel to the x-axis for each point of the line.

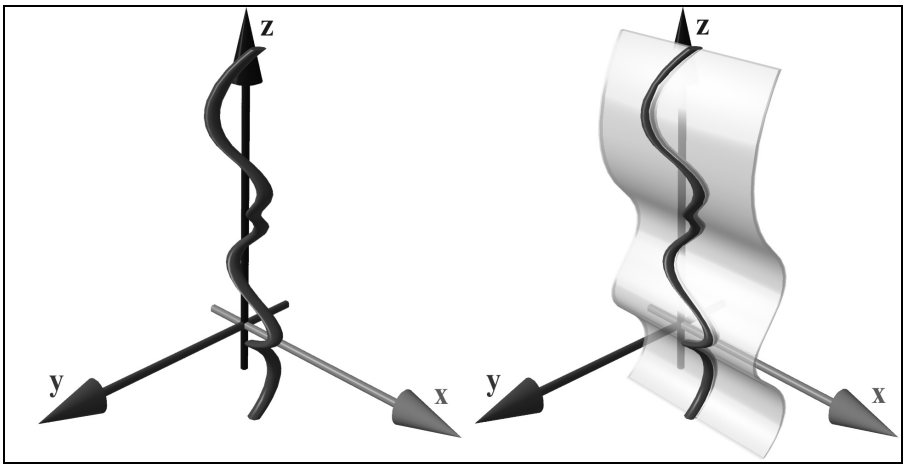


Figure 9.2: Left: A curved spatial line. Right: A curved planar defined by the curved spatial line.

This method is already used in medical environments. Therefore it is a visualization technique that is very likely to be accepted for daily clinical use by the medical personnel in hospitals. One of the biggest disadvantages of this technique is the very time-consuming and error prone manual generation process. For this reason a semi-automatic generation method is desirable. The following section is going to describe such an approach.

9.4.1 User interaction

First the user identifies a starting point and at least one endpoint within each desired vessel. Afterwards the curved line computation can be started. The resulting path can be inspected and finally this path can be centered in order to improve the quality of the resulting curved planar reformation.

9.4.2 Cost function

The curved line computation is based on a modified live-wire algorithm [13]. A proper cost function is defined as to keep the calculated path inside the vessel. In contrast to the live-wire method this algorithm searches in 3D space for the optimal path. As the path depends on the sum of the cost function of each voxel the path is very likely to lie within the vessels.

The local cost function $f_C(x,y)$ for a single step from a voxel x to the adjacent voxel y can be defined as:

$$f_C(x,y) = f_S(x) + f_I(x) + f_G(x,y) + f_L(x) \tag{9.1}$$

Where $f_S(x)$ is a constant cost function in order to keep the path short. This is necessary to avoid undesired high curvature regions of the path.

The *density interval function* $f_I(x)$ ensures that the path stays inside the vessel. This is accomplished by penalizing intensity regions, which differ from the intensity regions that with high probability represents contrast-enhanced vessels.

The gradient function $f_G(x,y)$ gives the difference of the intensity value of the previous voxel x and the intensity value of the subsequent voxel y. This function results from the assumption that in the direction of the central axis of the vessel the gradient magnitude is lower than in the direction to the vessel boundary.

Finally the Laplacian function $f_L(x)$ prevents the algorithm from tracking along and into bones. This function is necessary because of the partial volume effects and inhomogeneous regions of bony structures.

9.4.3 The algorithm

In order to find a path with minimal cumulative cost based on the local cost function mentioned above, Dijkstra's shortest path algorithm is used. As this algorithm generates all optimal paths for a given starting point s a region L can be defined where all shortest paths are computed. The information of all optimal paths is stored in a direction information for each voxel. Following the direction information from the endpoint to the starting point s produces the shortest path. The region L can also be referred to as a set of voxel. The set of voxels located on the boundary of L is defined as Q. Initially the sets L and Q are empty.
If a new endpoint e is defined two possibilities arise:

1. $e \in L$: In this case simply the direction information $dir(e_2)$ has to be followed. The reason for this is that all optimal paths for voxels in L are already computed according to Dijkstra's algorithm.
2. $e \notin L$: The new endpoint was not computed yet. Therefore Dijkstra's algorithm is started with the new endpoint e. It is obvious that the sets L and Q can be reused because the boundary Q expands independent of the endpoints.

9.4.4 Centering the curved line.

As any deviation from the vessel main axis causes errors in the curved planar reformation, a feature is added to correct the path in terms of finding the center of the vessel. The pure path-finder algorithm does not assure the path to be in the center of the vessel. This can be seen in figure 9.3. The main goal of the *center-finder* algorithm is to improve the calculated path.

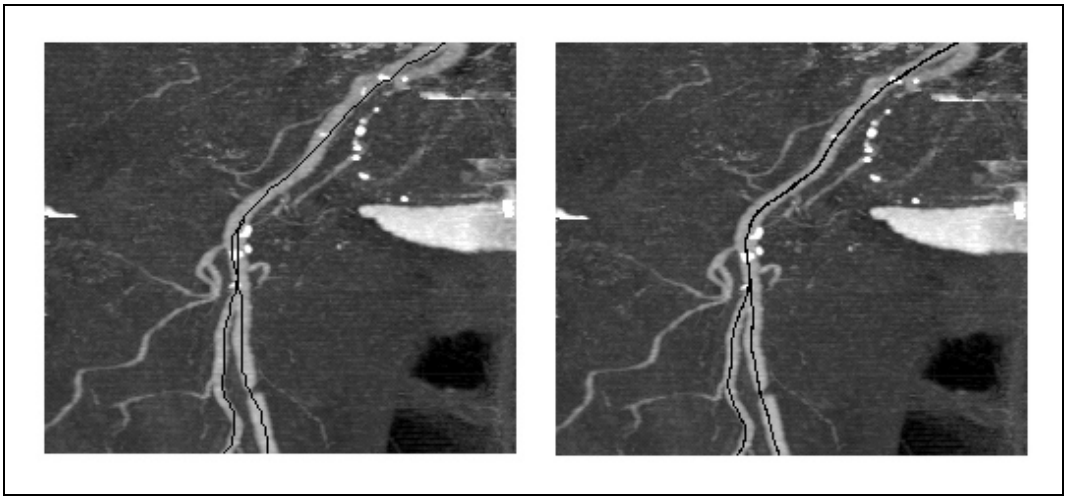


Figure 9.3: Left: Curved line computation without center correction. Right: With center correction.

9.5 User-assisted segmentation

The applied segmentation method is a rather basic segmentation technique. A combination of thresholding and region growing is applied. One of the main reasons for using a rather simple but computationally efficient approach is the large size of the datasets.

The bone-seg algorithm is working on so called *slabs*. A slab is a set of several spatially adjacent volume slices. Typical 30 to 50 slices are combined in one slab. The advantages of this arrangement are:

- *Better response time*: Because the amount of data is significantly smaller than the whole dataset the computation time after a user interaction is reduced.
- *Error containment*: The error inside one slab cannot propagate over the whole dataset.
- *Region dependent parameters*: As the correlations and properties of the objects differ depending on the regions within the dataset, the parameters can be set according to these regions.

The bone-seg algorithm is applied independently for each slab. Basically the algorithm consists of 3 steps. First a rough distinction between the different objects is done. Secondly the objects are labeled. In the final step the correct shape is computed. A predefined set of parameters is used for each slab. The user can change this set of parameters during the segmentation process. For each slab the set of parameters consists of:

- t_{class}: This threshold is used to distinguish different objects.
- t_{expand}: This threshold defines the enhancement of already identified objects. This threshold is motivated to handle partial volume effects and marrow inside the bones.
- t_{label}: This threshold separates between objects considered as bones or vessels. The threshold t_{label} is operating on the average density of objects.

First all slices are classified using a high threshold t_{class} in order to distinguish different objects. The classification process is based on the intensity value and gradient magnitude of the voxels. The connected regions are merged and finally labeled according to their properties. A second iteration of the whole process (except labeling) is done with a lower threshold t_{expand}. This step improves the quality of the segmented dataset by reducing noise due to partial volume effect and bone marrow. As the merging of different object types is prevented bones and vessels remain separated. After this step a user defined labeling of objects is possible. Finally the objects labeled as bone are removed.

9.6 Results

Table 9.1 gives an overview of the datasets' parameters. All datasets are real world datasets. These datasets were anonymized as patient information was deleted.

Name	Spatial resolution	Size in MB	Volume size (mmm)
100_198.dat	512 x 512 x 988	494,5	257 x 257 x 1070
004_old.dat	512 x 512 x 550	275,5	240 x 240 x 1102
022_old.dat	512 x 512 x 1000	500	260 x 260 x 1100

Table 9.1: Parameters of the presented datasets

The test environment consists of a PII 350 MHz system with 704 MB main memory, running Windows NT 4.0 SR 5 as operating system. The volume rendering was done on the commercial medical image processing system JVision/Space-Vision from TIANI Medgraph [25].
In table 9.2 the time needed for the investigation steps is summarized. As the segmentation process is a highly interactive process no division between user interaction time and computation time is made.

Name	CPR computation time	CPR user interaction time	CPR center finding time	Segmentation time
100_198.dat	16 min 20 s	1 min 30 s	1 min 14 s	25 min 41 s
004_old.dat	8 min 10 s	2 min 10 s	28 s	31 min 25 s
022_old.dat	15 min 40 s	1 min 20	29 s	18 min 11 s

Tabelle 9.2: Investigation time of the datasets

In the following the results of the investigation techniques applied on dataset 004_old.dat are presented.

Figure 9.4: MIP of dataset 004_old.dat.

The patient of dataset 004_old.dat suffers from serious calcifications and occlusions. The bypass in the left iliac artery is an indication of surgical intervention. It is very likely that such high-grade arterial disease leads to a symptomatic disease.

9.6.1 Curved planar reformation

The five paths are numbered according to their endpoints from left to right. In the following a CPR based on the non-centered paths are shown on the left hand side. The centered path reconstruction is shown in the right image of each figure. The CPRs of the four leftmost paths are shown in the following.

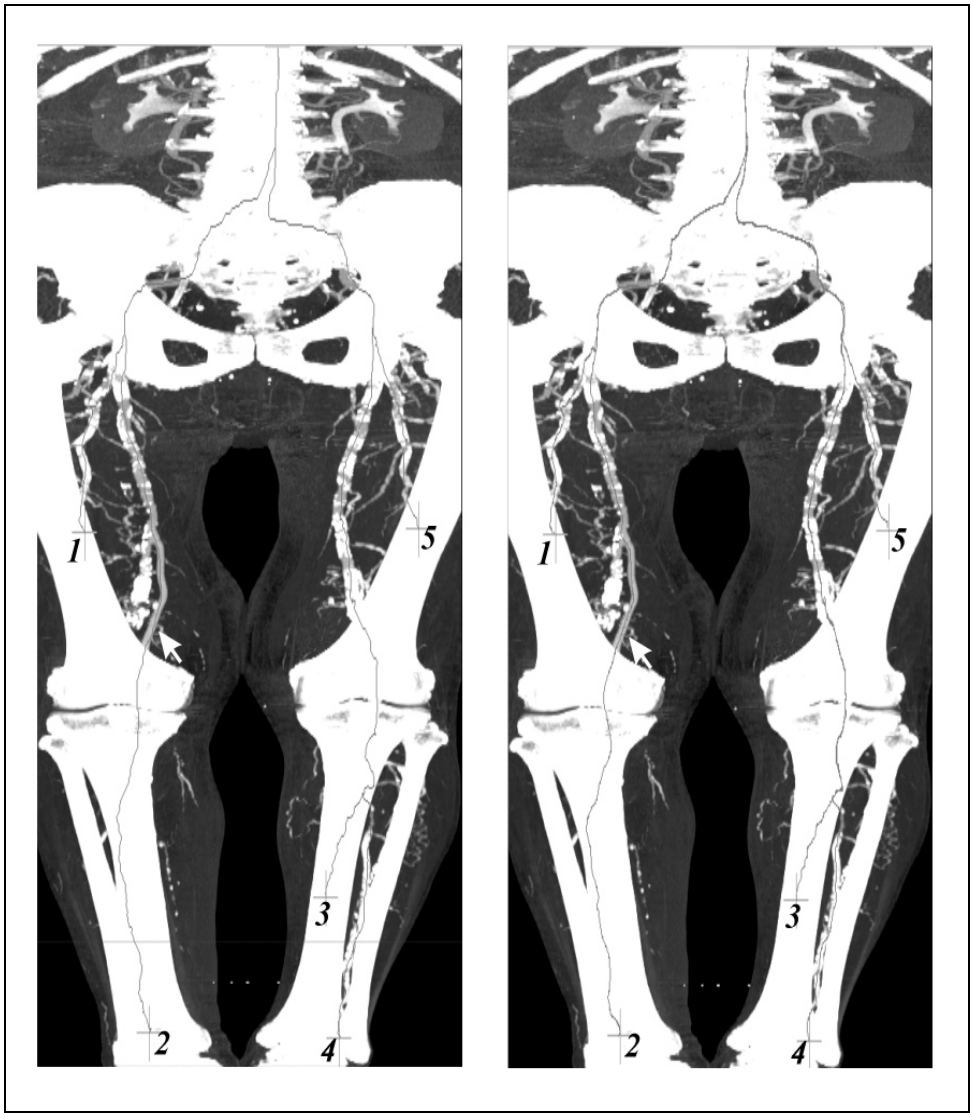


Figure 9.5: Left: Calculated path. Right: Centred path. The white arrow marks a region where the center-finder algorithm improves the image significantly.

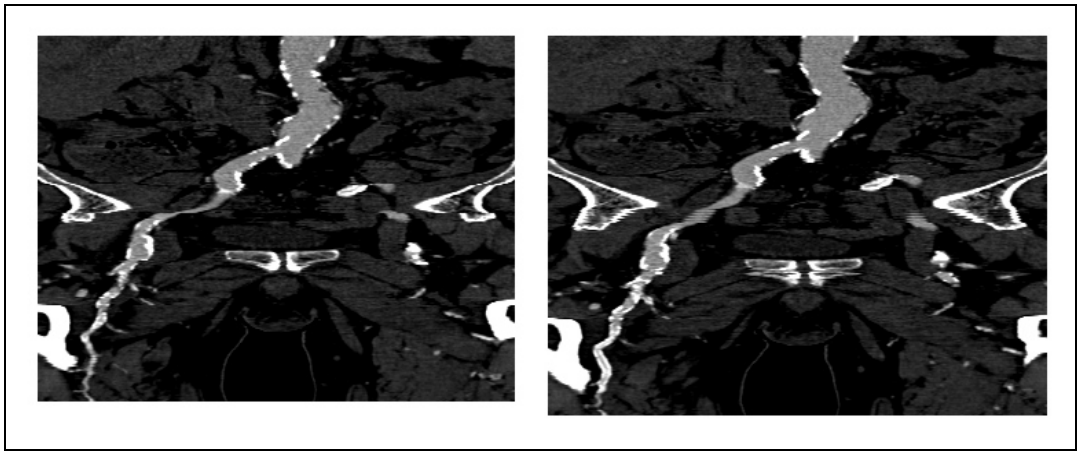


Figure 9.6: CPR of path number 1.

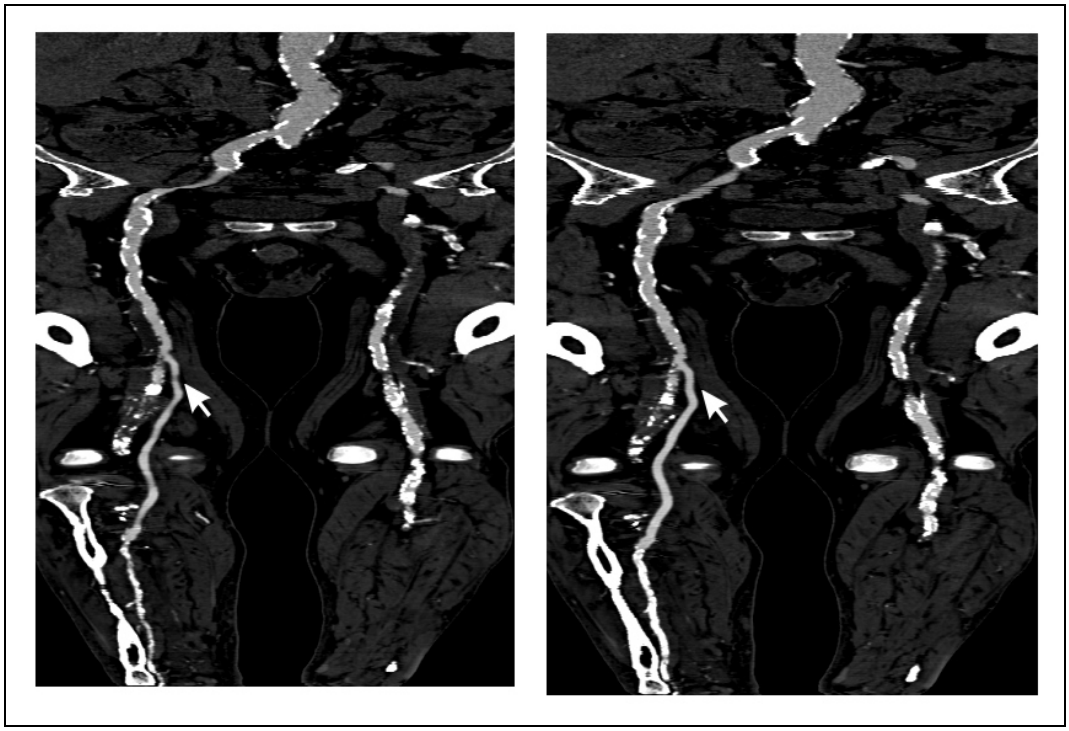


Figure 9.7: CPR of path number 2. A wrong stenosis is pointed out by the arrow in the left image. The improvement by centering the vessel is clearly visible in the corresponding region of the right image.

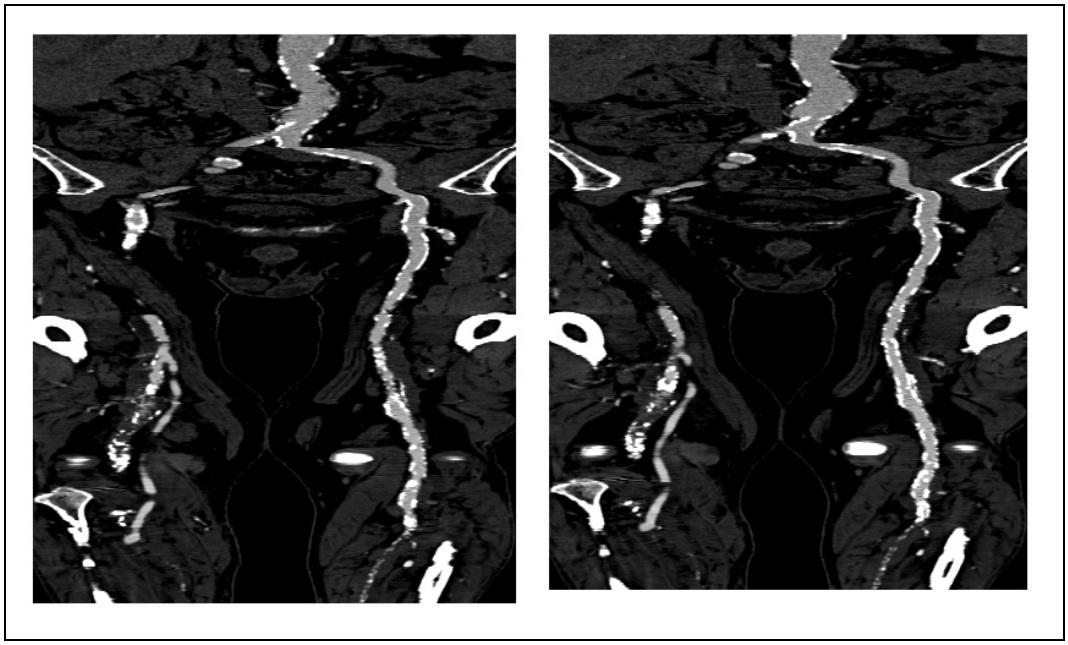


Figure 9.8: CPR of path number 3.

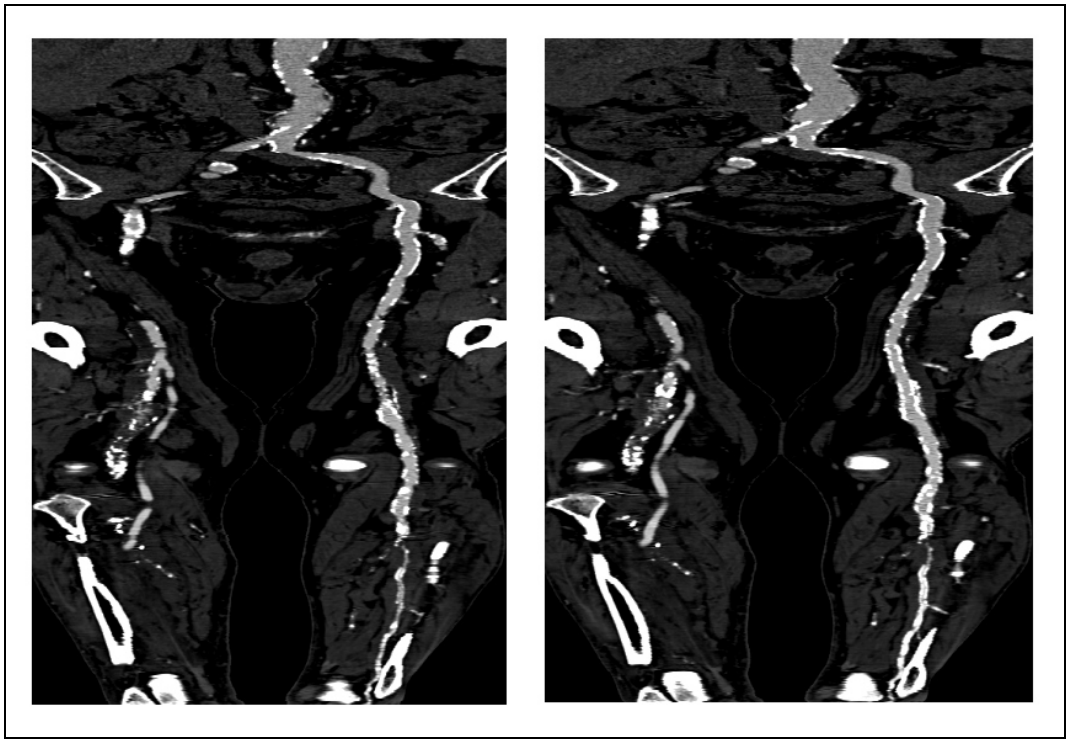


Figure 9.9: CPR of path number 4.

9.6.2 User interactive segmentation

As can be seen in figure 9.10 in the topmost area the dataset causes some difficulties. Therefore more noise than elsewhere in the dataset remains after segmentation.

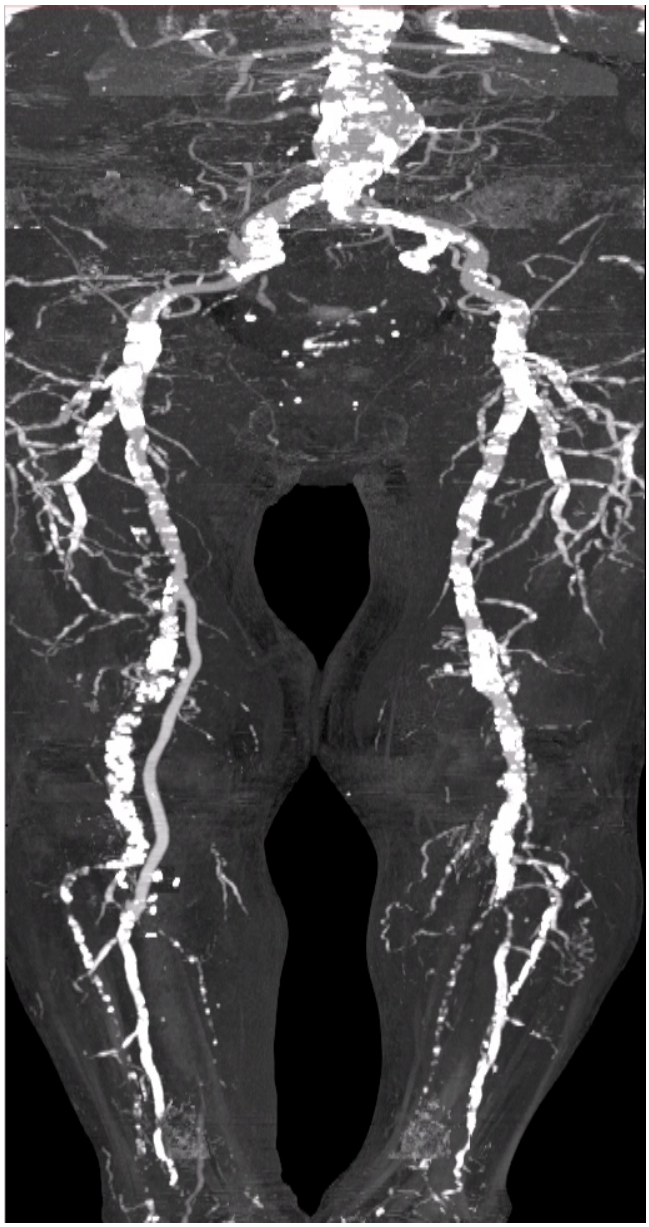


Figure 9.10: MIP of the segmented dataset 004_old.dat

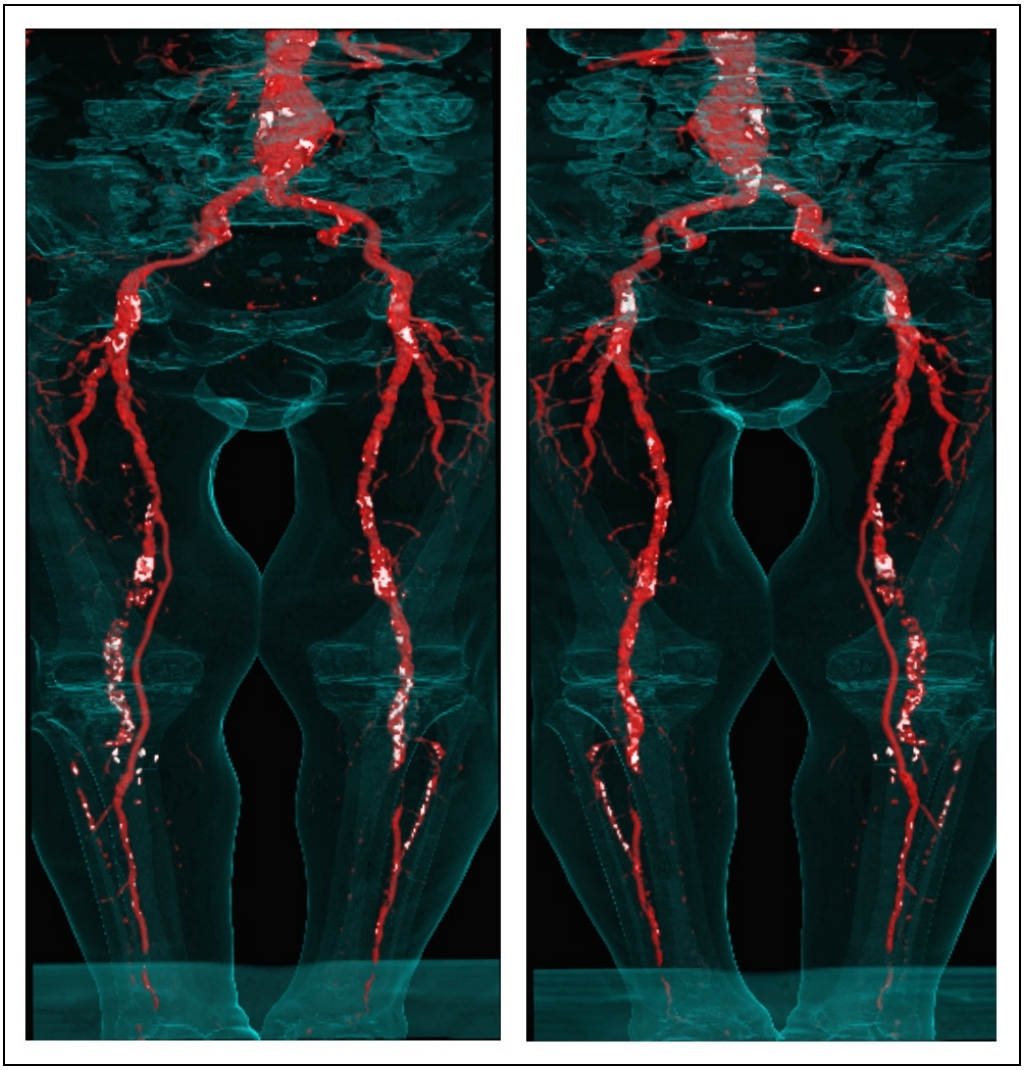


Figure 9.11: Front view (left) and back view (right) of the dataset.

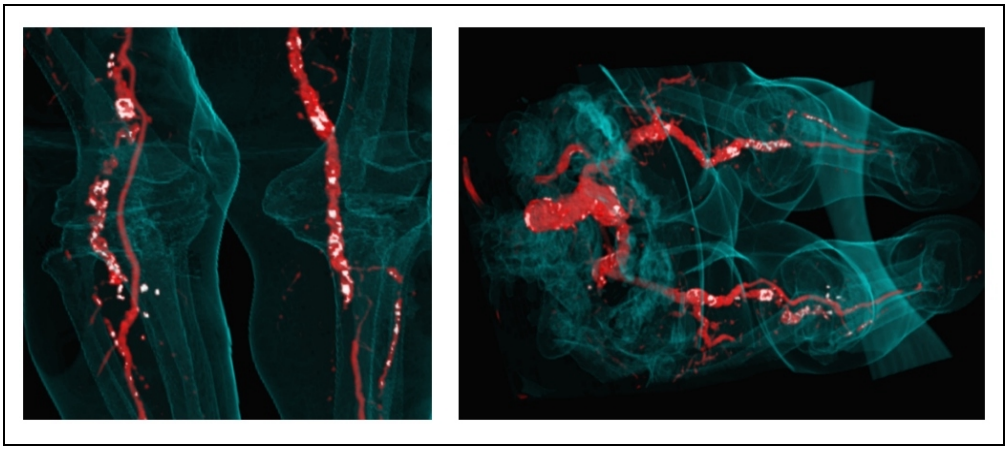


Figure 9.12: Detail inspection and an arbitrary viewpoint of the dataset.

9.7 Conclusions

Two different approaches of vessel investigation were presented. A global optimization method for the computation of a curved line was chosen because of the high reliability. The curved plane reformation based on the semi-automatic computed curved-line was considered as an appropriate visualization technique as this method is known well in the radiology community. Therefore high acceptance of this technique within a clinical environment is expected. An enhanced user-interactive segmentation-method focused on bone removal was proposed. The advantage of this technique is the well-balanced trade-off between high quality segmentation and time requirements.

Compared to methods currently in use all solutions presented here were approximately 8 times faster.

Chapter 10

Acknowledgments

At this point I want to express my thanks to all members of the BBA-group for the constructive support of my work and the great time we had. Especially I want to thank Dominik Fleischmann from the Department of Radiology of the University of Vienna for the assistance in medical topics and for making available various datasets. Special credits go to the guys from TIANI Medgraph, Vienna, Rainer Wegenkittl and Petr Felkel who introduced several outstanding ideas and therefore had a considerable impact on my work. Last but not least thanks go to Andreas König and Eduard Gröller, as their encouragement made this thesis possible.

This thesis was done in corporation with the VRVis – Research Center for Virtual Reality and Visualization – Vienna. VRVis (http://www.vrvis.at) is supported by the Austrian "K plus" research program.

Bibliography

[1] L. Mroz, A. König, and E. Gröller. Real-time maximum intensity projection. In E. Gröller, H. Löffelmann, and W. Ribarsky, editors, *Data Visualization '99*, pages 135–144, Springer, 1999.

[2] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, February 1987.

[3] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. *Computer Graphics (Proceedings of SIGGRAPH), Annual Conference Series*, pp. 451–458, 1994.

[4] VolumePro 500 User's guide, *www.rtviz.com*, 2000

[5] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (Proceedings of SIGGRAPH '87)*, vol. 21, no. 4, pp. 163–169, July 1987.

[6] H. Cline, W. Lorensen, S. Ludke, C. Crawford and B. Teeter. Two algorithms for the three-dimensional reconstruction of tomograms. *Medical Physics,* vol. 15, no. 3, pp. 320–327, May/June 1988.

[7] A. Neubauer and A. Kanitsar. ALVIS - Meeting the tremendous requirements arising with the visualization of aluminum foam samples investigated by high resolution industrial CT-modalities. *Proceedings of CESCG2000*, www.cg.tuwien.ac.at/studentwork/CESCG-2000/, 2000.

[8] J. Xuan, T. Adali and Y. Wang. Segmentation of magnetic resonance brain images: integrating region growing and edge detection. *2nd IEEE Intl. Conf. Image Processing*, Washington, D. C., 1995.

[9] N. Copty, S. Ranka, G. Fox, and R.V. Shankar. A Data Parallel Algorithm for Solving the Region Growing Problem on the Connection Machine. *Journal of Parallel and Distributed Computing*, vol. 21, no. 1, pp. 160–168, April 1994.

[10] A. Falcao and J. Udupa. A 3D generalization of user-steered live-wire segmentation. *Medical Image Analysis*, no. 4, pp. 389–402, 2000.

[11] K. Höhne and W. Hanson. Interactive 3d segmentation of MRI and CT volumes using morphological operations. *Journal of Computer Assisted Tomography*, vol. 16, no. 2, pp. 285–294, 1992.

[12] T. Schiemann, M. Bomans, U. Tiede and K. Höhne. Interactive 3D-segmentation. R. A. Robb (Ed.): *Visualization in Biomedical Computing II, Proc. SPIE 1808*, Chapel Hill, NC, pp. 376–383, 1992.

[13] A. Falcao and J. Udupa. An Ultra-Fast User-Steered Image Segmentation Paradigm: Live Wire on the Fly. *IEEE Transactions on Medical Imaging*, vol. 19, no. 1, January 2000.

[14] W. Barrett and E. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, no. 4, pp 331–341, 1997.

[15] E. Mortensen and W. Barrett. Intelligent Scissors for Image Composition. *Computer Graphics (SIGGRAPH `95)*, Los Angeles, CA, pp. 191–198, August 1995.

[16] O. Wink, W. Niessen and M. Viergever. Fast Delineation and Visualization of Vessels in 3-D Angiographic Images. *IEEE Transactions on Medical Imaging*, vol. 19, no. 4, pp. 337–346, April 2000.

[17] M. Göpfert. Segmentierung und Analyse von Gefäßbäumen in CT-Bildern zur präoperativen Einteilung der Leber in Segmente. *Technical Report*, no. 95, DKFZ, Heidelberg, 1997

[18] Y. Sato, C. Westin, A. Bhalerao, S. Nakajima, et. al.. Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 160–180, April–June 2000.

[19] A. Frangi, W. Niessen, K. Vincken, M. Viergever. Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, 1496, pp. 130–137, 1998.

[20] Y. Sato, S. Nakajima, N. Shiraga, et. al.. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis*, vol. 2, no. 2, pp. 143–168, 1998.

[21] C. Revol and M. Jourlin. A new minimum variance region-growing algorithm for image segmentation. *Pattern Recognition Letters*, no. 18, pp. 249–258, 1997.

[22] S. Beucher and C. Lantuejoul. Use of Watersheds in Contour detection. *International Workshop on Image Processing: Real-time Edge and Motion detection/estimation*. Rennes, France, September, 1979.

[23] S. Beucher. The watershed transformation applied to image segmentation. *Conf. on Signal and Image Processing in Microscopy and Microanalysis*, Cambridge, UK, September 1991.

[24] F. Fowkes, E. Housley, E. Cawood, et. al.. Edinburgh Artery Study: prevalence of asymptomatic and symptomatic peripheral arterial disease in the general population. *Int J Epidemiol*, no. 20, pp. 384–392, 1991.

[25] TIANI Medgraph, *www.tiani.com*

[26] L. Westover. Footprint Evaluation for Volume Rendering. *Proceedings of SIGGRAPH '90*, vol. 24, no. 4, pp. 367–376, August 1990.