Technische Universität Wien

Dissertation

Visualization Techniques for Virtual Endoscopy

ausgeführt
zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller,
Institut 186 für Computergraphik und Algorithmen,

eingereicht
an der Technischen Universität Wien,
Fakultät für Technische Naturwissenschaften und Informatik,

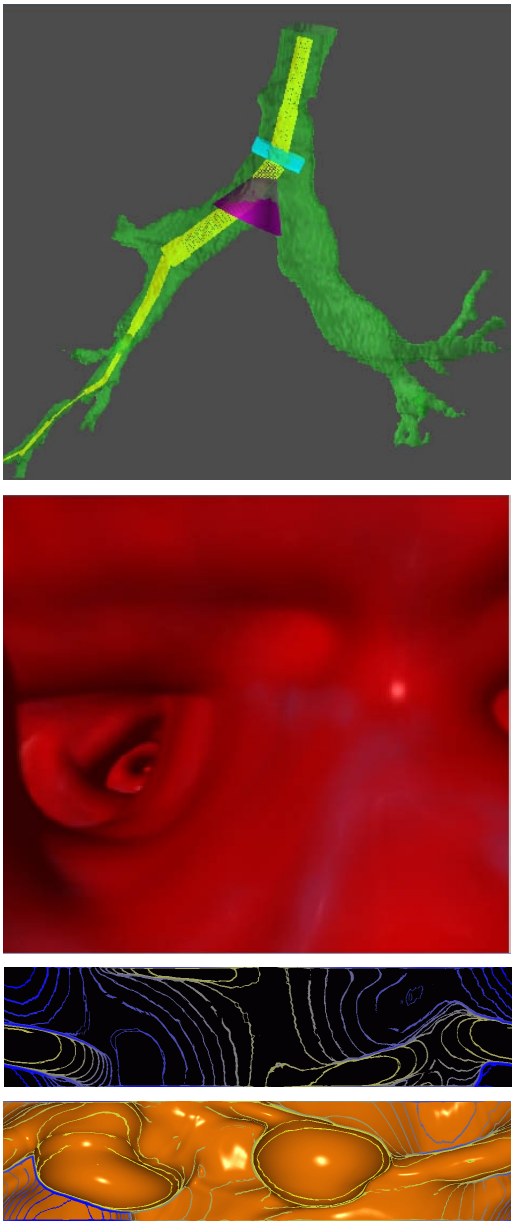
von
Anna Vilanova i Bartrolí
Matrikelnummer 9826871,
Linke Wienzeile 94/13
A-1060 Wien
geboren am 03. Mai 1973 in Olot (Girona) Spanien.

Wien, im September 2001,

Anna Vilanova i Bartrolí

Visualization Techniques for Virtual Endoscopy

(PhD Thesis)



http://www.cg.tuwien.ac.at/research/vis/vismed/
http://www.cg.tuwien.ac.at/research/theses/
mailto:anna@cg.tuwien.ac.at

Acknowledgements

First of all, I would like to thank my supervisor, the Meister Eduard Gröller, for his good advice and support in everything. I express my gratitude to Rainer Wegenkittl for the valuable discussions and his collaboration on the work presented in this thesis. I specially thank the former and current members of the vis-group in the Institute of Computer Graphics and Algorithms of the Vienna University of Technology (Jirí, Balázs, Tom, Armin, Andreas, Lukas, Markus, Helmut, Helwig and Wolfgang) for the great fun it was working with them, the ice creams, the beer and, of course, the scientific discussions.

I thank my friends in Vienna for the good meals, the turkish lessons, the chats, enjoyment and their major contribution in making my stay in Austria pleasant. I thank my friends who were far away from me but never forgot me and gave me the moral support which enabled me to finish this thesis.

I specially dedicate this thesis to my parents without whose effort this work would have never been possible, and to my complete family for fully supporting me in everything I undertook.

My immense gratitude goes to Paul for his company, help, patience and comfort whenever I needed it.

I would like to thank Dr. Martin C. Freund and the Department of Radiology at the Leopold-Franzens-University of Innsbruck for their collaboration and for providing part of the data used in this thesis.

I thank Dr. Erich Sorantin from the Department of Radiology in Graz for his collaboration and for providing the data sets used for the virtual colon unfolding and the images of the dissected colon.

The work presented in this thesis has been funded by the V^{is}M^{ed} project. V^{is}M^{ed} is supported by *Tiani Medgraph*, Vienna (*http://www.tiani.com*), and the *Forschungsförderungsfonds für die gewerbliche Wirtschaft*, Austria. See *http://www.vismed.at* for further information on this project.

Kurzfassung

Virtuelle Endoskopie ist eine Technik hohle Organe und anatomische Aushöhlungen mittels Methoden der medizinischen 3D Visualisierung und Verfahren aus der Computergraphik zu erforschen. Virtuelle Endoskopie ist eine vielversprechende Technik, welche die tatsächliche Endoskopie verbessert, oder in manchen Anwendungen sogar ersetzen kann. Endoskopie ist eine invasive und für den Patienten risikoreiche Methode. In den letzten Jahrzehnten erfolgte viel Forschungsarbeit in diesem Bereich. Diese Dissertation reflektiert einen Teil dieser Forschung und im speziellen die Untersuchung von Visualisierungstechniken für die virtuelle Endoskopie.

Die meisten Methoden der virtuellen Endoskopie simulieren das Verhalten echter Endoskopie. Der erste Teil der Dissertation konzentriert sich auf die Betrachtung dieser Methoden und stellt die Struktur eines solchen Systems vor. Ein Prototyp angelehnt an diese Struktur wird entwickelt. Ein zentrierter Pfad innerhalb der Organe wird verwendet, um eine Kamera durch das Organ zu führen. Wir verbessern einen existierenden Algorithmus, um den zentrierten Pfad durch das Organ zu finden. Zwei neue Techniken werden beschrieben, um hochqualitative, perspektivische Darstellungen zu beschleunigen. Die erste Methode ist eine neue Space-Leaping-Technik für Volumsdarstellungen. Die zweite Methode nützt Hardware-Beschleunigung (VolumePro) zur orthographische Volumsdarstellung, um perspektivische Volumsdarstellungen zu erzeugen.

Simulationen von wirklicher Endoskopie ist in vielen Anwendungen nicht die am besten geeignete Visualisierungs-technik. Eine Endoskopie ist an gewisse physikalische Beschränkungen gebunden, welche für die virtuelle Endoskopie nicht gelten. Der zweite Teil der Dissertation präsentiert zwei Techniken, welche den Dickdarm virtuell entfalten, um die Oberfläche zu untersuchen, und Polypen entdecken zu können. Wir konzentrieren uns auf den Dickdarm, obwohl diese Techniken auch für andere Organe einsetzbar sind.

Die erste Technik entfaltet den Dickdarm lokal und erzeugt eine animierte Abfolge von aufeinanderfolgenden entfalteten Regionen. Die Bilder werden mittels einer Projektionstechnik erzeugt, welche dem Mediziner die Visualisierung eines Großteils der Oberfläche erlauben. Diese Methode erlaubt es Polypen einfach zu erkennen, welche aus dem Blickwinkel einer Endoskopie von Falten verdeckt, oder schwer zu lokalisieren wären. Der Nachteil der lokalen Dickdarmentfaltung ist die Notwendigkeit die Untersuchung anhand eines Films durchführen zu müssen, um die gesamte Oberfläche zu visualisieren.

Die zweite Technik — Nonlinear Colon Unfolding — erlaubt dem Mediziner in einer einzelnen Darstellung die Oberfläche des Organs zu untersuchen und möglichst viel Information darzustellen. Man erhält ein einzelnes Bild des vollständig entfalteten Dickdarms. Auf diese Weise können problembehaftete Bereiche schnell identifiziert und anschließend genauer untersucht werden.

Um die Anwendbarkeit zu zeigen wurde jede Technik wurde an mehreren Datensätzen getestet. Jedoch sind diese Tests nicht ausreichend, um die Methoden bereits in einem klinischen Umfeld einsetzen zu können. Die Resultate zeigen jedoch das Potential der entwickelten Methoden auf.

Abstract

Virtual Endoscopy is a technique to explore hollow organs and anatomical cavities using 3D medical imaging and computer graphics. Virtual Endoscopy turns out to be a promising technique to improve, or even in some procedures substitute, real endoscopy. A real endoscopy is invasive and usually implies some risk for the patient. In the last decades, much research has been done in this field. This thesis reflects a piece of this research concentrated on investigating visualization techniques for virtual endoscopy.

Most common methods in virtual endoscopy simulate the behavior of a real endoscope. In the first part of this thesis, we concentrate on studying these methods and present a general framework for such a system. We develop a prototype according to this framework. A central path of the organ is used to move the camera through the organ. We improve an existing algorithm to find the central path of the organ. We describe two new techniques to accelerate high quality perspective volume rendering. The first method is a new space leaping acceleration technique for ray casting. The second method uses hardware acceleration (i.e., VolumePro) of orthographic volume rendering to generate perspective volume rendering.

Simulating a real endoscopy is not the most suitable visualization technique in many endoscopy procedures. A real endoscopy is restricted due to physical limitations that a virtual endoscopy does not have. The second part of the thesis presents two techniques which virtually unfold the colon to inspect its surface and detect polyps. We concentrate on the colon although these techniques could be used with other organs too.

The first technique, locally unfolds the colon and generates an animation sequence from consecutive unfolded regions. The images are generated with a projection technique that allows the physician to visualize most of the surface, and to easily recognize polyps that in an endoscopic view would be hidden by folds or would be hard to localize. The drawback of the local colon unfolding is that the physician has to inspect a video to be able to visualize the whole surface.

The goal of the second technique, nonlinear colon unfolding, is to enable the physician to inspect and get as much information as possible of the inner organ surface at a first glance. It obtains a single image of the complete unfolded colon. In this way, the problematic areas can be identified quickly and inspected later in more detail.

Every technique have been tested with several data sets to show their feasibility. Although, these tests are not enough to use the methods in the clinical environment, the results show their potential to achieve such an state.

Contents

List of Figures

Chapter 1

Introduction

> Faith is to believe what you do not see; the reward of this faith is to see what you believe.
>
> *Saint Augustine (354 - 430)*

1.1 Scientific Visualization

”A picture is worth a thousand words.” This common expression symbolizes the extraordinary ability of the human mind to extract information through visual stimulation. Illustrations and drawings are an established didactic technique. Aristotle (384-322 B.C.) already drew figures to teach geometry. Leonardo Da Vinci (1452-1519) sketched fluid flows to be able to understand their behavior. Andreas Vesalius (1514-1564), one of the pioneers in the study of human anatomy, mentioned that the only way to really learn anatomy was visually. His atlases [Vesa43] communicated his idea that you must actually see to learn. Vesalius supported his theory, using pictures which showed what no text could explain (see figure 1.1).

In the last decades, technology has allowed scientific researchers to do simulations and computations that in previous times were impossible. These computations generate a huge amount of numerical data which is not the best way for humans to understand them. New algorithms and techniques had to be explored, since existing techniques were not enough to represent those complex results. Scientific Visualization has become the union of complex computations and computer graphics to help researchers to better understand complex structures in a meaningful way. With computer graphics, you can take the data and represent it in three dimensions. You can travel through the objects and represent multidimensional data in various ways. You are able to observe the beauty and complicated behavior of mathematical models like fractals. ”Scientific Visualization is the art of making the unseen visible” (Clifford Pickover).

The birth of Scientific Visualization as a discipline is usually placed with the publication of the 1987 Report of the National Science Foundation’s (NSF’s) Advisory Panel on Graphics, Image Processing and Workstations. The report contained the term ”Visualization in Scientific Computing”, which was later shortened to ”Scientific Visualization”. Visualization was seen as having the potential for fostering important scientific breakthroughs [Rose94].

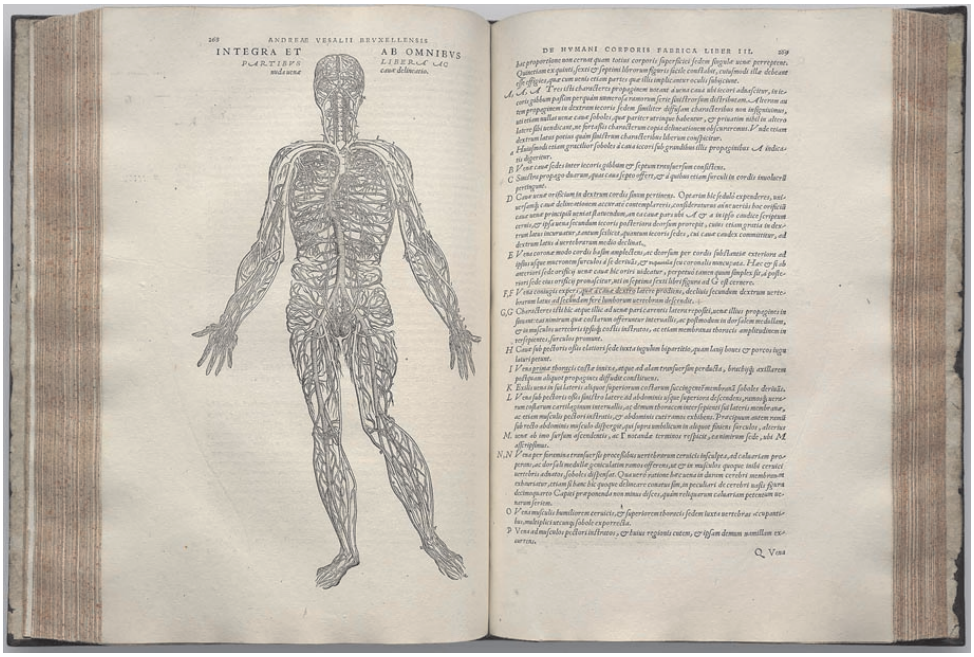


Figure 1.1: Two pages of the *De Humani Corporis Fabrica* from Vesalius, where the veins of the body are illustrated [Vesa43]

Scientific Visualization techniques can be divided in different branches depending on the data to visualize. Some of the branches are the visualization of medical data, flow dynamics, geographic information, biological microscopic data, molecules, large-scale data and architectural data.

This thesis is concerned with Medical Visualization which deals with visualizing human anatomy from real human data.

1.2 Medical Visualization

In previous time, the only way to look inside the body was by dissection, irrespective whether the goal was diagnostic or educational. As Vesalius said, ”to learn anatomy it is necessary to see it.” A usual anatomy lesson in the seventeenth century can be seen in Rembrandt’s famous painting ”The anatomy lesson of Dr. Nicolaes Tulp” (see figure 1.2). Doctor Tulp needed a real dissection of a corpse to teach the students the human internal organs.

Nowadays, this is not necessary anymore. We can look at the internal part of a body without the need of dissection, thanks to the evolution of 3D medical imaging techniques (i.e., 3D scanners) combined with computer graphics.

3D scanners measure the physical properties of an object at determined locations (i.e., by sampling). The measured locations are usually distributed over a regular grid. The spacing of the samples taken is called spatial resolution.

Figure 1.2: Rembrandt Harmenszoon van Rijn (1606-1669) ”The anatomy lesson of Dr. Nicolaes Tulp”

3D scanners are classified depending on the acquisition technology and physical principles they are based on:

Computer Tomography (CT) was the first 3D imaging technique. It consists of projecting X-rays through transversal cuts of the body (i.e., slices). The radiation that penetrates the body is measured by an array of detectors that are able to see the beam at a particular orientation. For each slice, the X-ray tube rotates around the slice and several beams are measured. Then an image is generated by using the different measurements. The technique to reconstruct the image from these measurements is called back projection.

Moving the scanner along the body, we can obtain a stack of 2D slices of a part of the human body (see figure 1.3).

Since CT is based on X-rays, each slice image encodes the tissues impermeability to X-rays. One disadvantage of this method is that it charges the patient with a dose of irradiation and this limits the quality of the results.

Magnetic Resonance Imaging (MRI) can selectively measure different soft tissue characteristics, but it is also one of the most complicated imaging modalities techniques. This technique is based on the high sensitivity of the hydrogen protons to align with a magnetic field. In the alignment, the protons tend to fluctuate about a magnetic field. This resonant oscillation is called magnetic resonance. By applying short radio frequency (RF) pulses to a specific anatomical slice, the protons in the slice absorb energy at this resonant frequency causing them to spin perpendicularly to the magnetic

field. As the protons relax back into alignment with the magnetic field, a signal is received by a RF coil. This signal is processed by a computer to produce images of the anatomical area of interest. Since the tissues have different hydrogen atom densities, they generate energy at different levels. Depending on the type of RF pulse sequence used, different tissue characteristics can be measured. Furthermore, MRI has the advantage over CT of the absence of irradiation.

Emission-Computed Tomography does not measure anatomy, as CT and MRI, but body functions like metabolism. There are basically to methods: PET (Positron Emission Tomography) and SPECT (Single Photon Emission Computer Tomography). Radioactive components are introduced into the body, which react with the body tissues producing gamma rays. These rays are picked up by a receiver and translated into an image. This technique can be used to detect the unusual metabolic activities that take place in growing tumors or to image the flow of blood through coronary arteries.

Ultrasound scanners use high-frequency sound waves (1MHz-20MHz) to visualize internal structures of the body. These sound frequencies interact with the tissues and their resulting echoes are used to create the images. Ultrasound is used to assess fetal development, blood flow, the heart and vascular system. The resulting images suffer from a high level of noise.

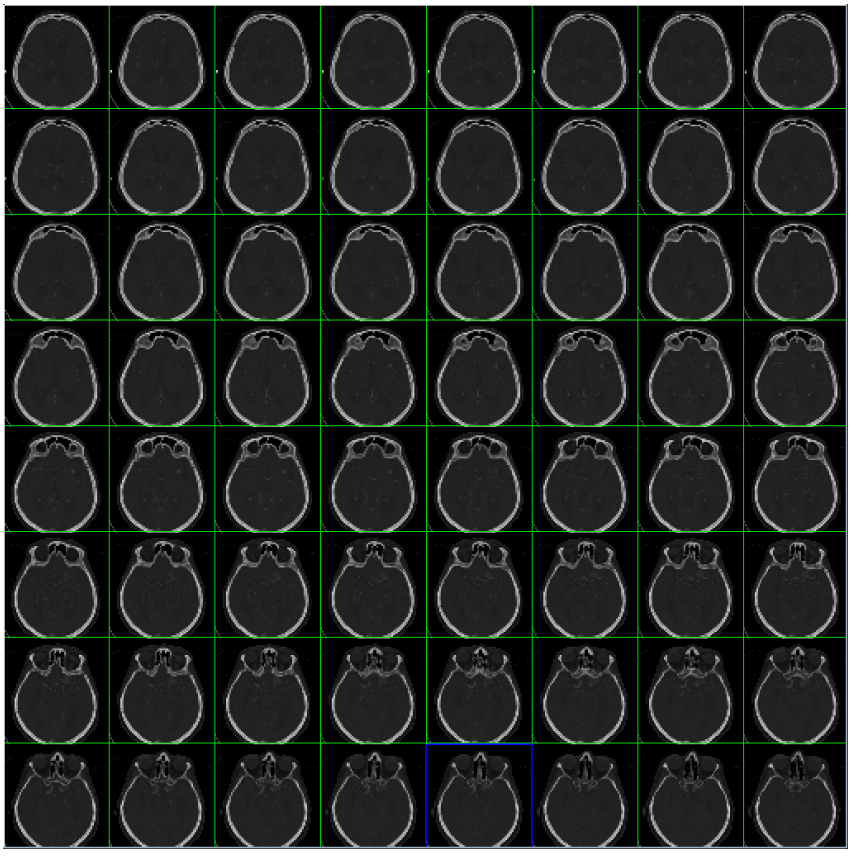


Figure 1.3: 2D slices result of a computer tomography of a head

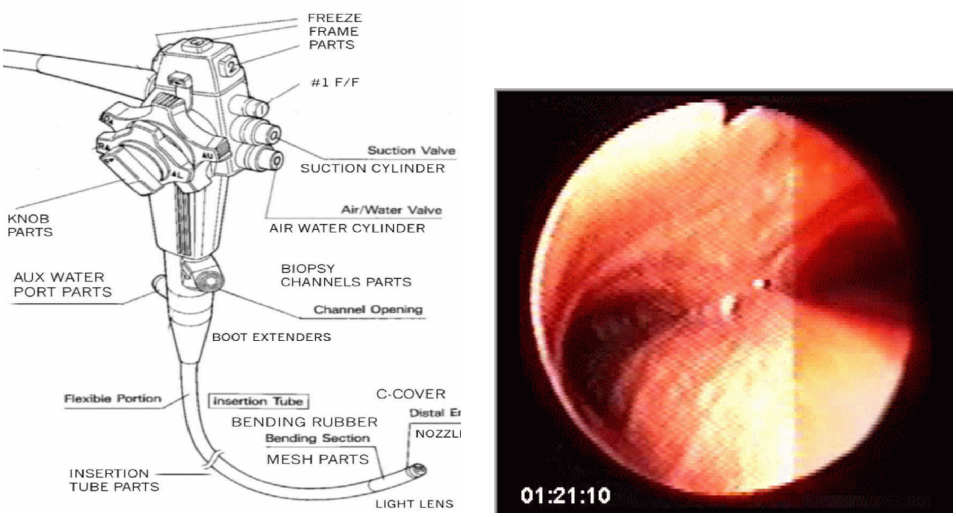
The acquisition techniques presented provide different information which in most of the cases has to be combined to get a meaningful result.

The 3D scanners provide 3D volumetric data. In the case of MRI, CT and PET or SPECT the result is a stack of 2D slices, or a set of cross-section images of the body. Nowadays, radiologists are trained to look at 2D images and do a mental reconstruction of the 3D organs. Although the ability of radiologists to perform such a task is amazingly high, in some cases this reconstruction is too difficult and even impossible for the human brain (e.g., in case of blood vessels).

The 2D slices can be aligned and stacked to produce a so-called 3D image or volume data. The volume data is represented by the regular grid formed by the locations at which the scanner sampled the object. This volume data can be visualized directly obtaining a 3D projection of the organ.

Medical visualization is concerned with the visualization of volume data obtained from 3D medical imaging techniques. In the next chapter, we describe the most common techniques to visualize volume data (i.e., Volume rendering techniques).

Usually, visualization techniques are based on the requirements of the application where it should be used. In this work, we concentrate on techniques to be used in virtual endoscopy.



a) b)

Figure 1.4: **a)** Real endoscope system. **b)** Image of the lumen of the trachea captured by a real endoscope.

1.3 Virtual Endoscopy

In a medical environment, an endoscopy is a procedure where an endoscope (i.e., a slender tubular optical instrument; see figure 1.4) is used for examining the interior of a body cavity or hollow organ through natural orifices or small incisions. Electronic endoscopes use a video chip and strobe light mounted to that tube, to capture an image which is then displayed on a video monitor. This procedure is applied for diagnostic as well as for surgery purposes. Endoscopy is uncomfortable for the patient and sedation and

anaesthesia may be necessary. Furthermore, it involves a degree of risk for the patient since it can cause perforation, infection and hemorrhage.

Virtual endoscopy is a promising new technique to explore hollow organs and anatomical cavities using 3D medical imaging and computer graphics (i.e., medical computer visualization techniques). The fields of application of virtual endoscopy are manifold.

- Non-invasive diagnostic endoscopy (e.g., colonoscopy) avoids the risks associated with real endoscopy.
- Virtual endoscopy can be used for educational purposes like endoscopists training (e.g., sinus surgery).
- A special field of application is the usage of virtual endoscopy for surgery planning.

Furthermore, virtual endoscopy does not suffer from some limitations that real endoscopy procedures have. Special parts of the human body, which are impossible to access with a real endoscope (e.g., blood vessels, thoracic aorta) can be investigated with virtual endoscopy. Endoscopes display only the inner surface of hollow organs and yield no information about the anatomy within or beyond the wall which can be visualized in virtual endoscopy (see figure 1.4b). Moreover, new visualization techniques can be developed which do not simulate an endoscopic view and achieve more meaningful information.

On the other hand, virtual endoscopy is limited to visualize the information that is captured by 3D medical imaging techniques. This implies that virtual endoscopy cannot visualize real colors which in some cases is necessary to determine whether a polyp can develop in malignancies.

1.4 Thesis overview

This thesis concentrates on visualization techniques for virtual endoscopy, mainly for diagnosis purposes. However, some of the results can be used in a more general scope.

In chapter 2, the basic concepts about volume rendering, which is the basic visualization technique used for 3D medical data, are described.

The thesis is then divided into two parts. The first part contains the description of a virtual endoscopy system whose goal is to simulate a real endoscopic view. Chapter 3 presents a general framework of a virtual endoscopy system with all the required modules. This thesis deals with two specific modules of this system: navigation and rendering. In chapter 4, we present a method for finding the central path of an organ which is used during navigation. Chapters 5 and 6 propose two new techniques to accelerate high quality perspective volume rendering.

The second part of the thesis concentrates on new visualization techniques which try to overcome the limitations of a real endoscopy for some procedures and use the flexibility of the virtual manipulation of objects. These methods give a more adequate visualization than a real endoscopic view would provide. The main idea is to unfold the organ. In this way, physicians can explore the surface of the organ where polyps can be detected in a more adequate way. In chapters 8 and 9, two new methods to visualize unfolded organs are presented.

Finally, conclusions derived from the work presented in this thesis are discussed in chapter 10.

Chapter 2

Volume Rendering

> The difficulty lies not in the new ideas, but in escaping the old ones, which ramify, for those brought up as most of us have been, into every corner of our minds.
>
> *John Maynard Keynes (1883-1946)*

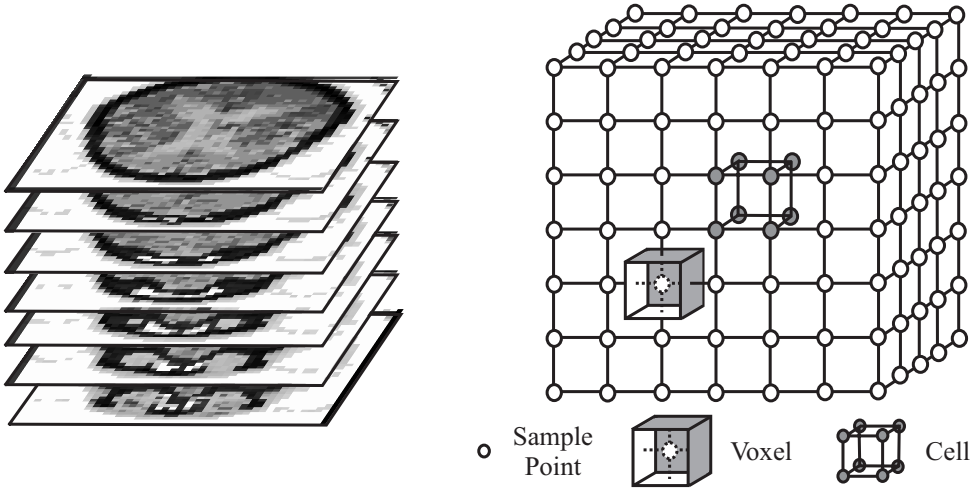
2.1 Introduction

Volume rendering concerns all possible projection or visualization techniques that are used to inspect volume data.

As we have mentioned before, one of the main sources of volume data is 3D medical imaging. Nevertheless, it is not the only source, and medicine is neither the only application where volume rendering can be used. For educational proposes, it is acceptable to cut the original object in slices and take pictures producing a volume data. An example is the data set of the Visible Human Project [Vis86]. In microscopic analysis, confocal microscopes produce high-resolution optical slices of microscopic objects. Geoseismic data is also volumetric data which is used, for example, in oil exploration by finding the correct location where to drill. Physical simulations of fluid dynamics also produce volume data for the visualization of the fluid behavior. There are many more applications and acquisition techniques besides the ones mentioned above where volume-data visualization techniques are used.

Depending on the source, volume data might be given on a cartesian rectilinear grid, or on a curvilinear grid, or maybe even completely unstructured. Depending on the grid structure different visualization algorithms exist. In this thesis, we concentrate on data coming from 3D medical imaging. This means, we deal with visualization techniques for cartesian rectilinear grids (see figure 2.1). The grids are isotropic if the distance between consecutive samples is constant for all dimensions. Otherwise they are anisotropic grids.

Volume data represented by a cartesian rectilinear grid is a three-dimensional array of data elements. For volume data, each data element is called a voxel (volume element). A digital 2D picture is a two-dimensional array of data elements. Each data element is called pixel (picture element) and represents the measurement of a physical property at a defined location (e.g., intensity or color). Volume data can be seen as a 3D image and a voxel as a 3D extension of a pixel. From this follows that several algorithms



a) b)

Figure 2.1: **a)** Stack of 2D slices which will generate the 3D volume data represented by a cartesian rectilinear grid. **b)** Regular grid structure of volume data illustrating the concept of voxel and cell.

used in image processing for 2D images have been extended to 3D and have been used for volume data manipulations [Lohm98].

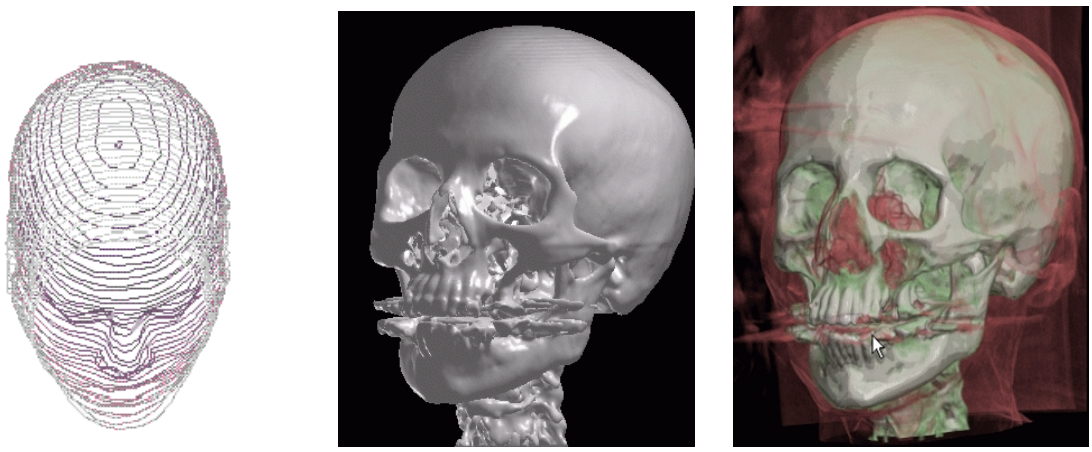
Literature does not agree on the definition of a voxel. In this thesis, a voxel is the minimal division of a 3D image (i.e., volume data). A voxel is represented by a cuboid whose value is constant in its volume. The center of a voxel corresponds with a sample point position (see figure 2.1). A cell is defined as the cuboid whose vertices correspond with neighboring sample points.

2.2 Classification of Volume Rendering Techniques

Through the last decades, a large number of techniques to visualize volume data has been proposed. The criteria by which these techniques could be classified are also diverse. One possible classification is to group the techniques according to the dimensionality of the geometric entities that are visualized: lines (1D), surfaces (2D) and volume data (3D).

Since usually 3D scanners produce a collection of 2D slices, one of the first visualization methods consisted of extracting contours within each slice. The contours are obtained by segmentation (e.g., manual or thresholding) of the desired object and finally visualized in 3D. It is obvious that the quality of such a visualization is considerably low, and it is difficult to obtain relevant information from such an image (see figure 2.2a).

One step further resulted in the visualization of surfaces. Various algorithms were created to approximate the surface by polygons (i.e., triangles or squares) defined between the contours. These methods are called tiling algorithms. Various algorithms and criteria were proposed to obtain the optimal surface between the contours [Kepp75, Fuch77]. Tiling algorithms suffer from not being able to guarantee a generally correct surface model. This occurs in areas where branching appear or elements of the



a) b) c)

Figure 2.2: Examples of volume visualization according to the dimensionality of the entities that are visualized: **a)** (1D) contour lines, **b)** (2D) surface obtained with the marching cubes algorithm **c)** (3D) direct volume rendering.

structure start or terminate. Due to ambiguities in the association of contours, these algorithms cannot execute completely automatically.

Other researchers [Artz81, Udup83] proposed the so called cuberille model to solve the topological ambiguities of tiling algorithms. This model is based on the assumption that the elements that belong to an object must be connected. The algorithm based on the cuberille model extracts the surface formed by the faces of the cells that constitute the connected surface of the object.

The most popular algorithm for surface extraction was presented later on under the name of marching cubes algorithm [Lore87]. The polygonal surface obtained from the algorithm is an approximation of an isosurface. An isosurface is an implicit surface defined by equation $f(x, y, z) = T$, where $f(x, y, z)$ represents the continuous function sampled by the volume data, and T is a threshold value which defines the isosurface. The algorithm goes through each cell and checks whether part of the isosurface is found within the cell. This can easily be checked by looking at the values of the eight vertices of a cell.

If some of the values of the vertices are bellow the value of T and some are above T, then the surface crosses the cell, since we suppose f to be continuous. Afterwards, usually by means of interpolation, the cell-surface intersection points are detected, and the points are joint forming triangles. There are 256 possible configurations of a cell, which are reduced by symmetries to 15 patterns. This fact simplifies the algorithm into looking for the cell pattern in a lookup table and then defining the triangles using linear interpolation between the vertices of the cell. An example of such a surface can be seen in figure 2.2b. The initial marching cubes algorithm can lead to ambiguities and erroneous holes in the surface. Several authors proposed solutions to these ambiguities [Bloo88, Niel91, Ning93].

Using one of the previous methods, we obtain a polygonal surface that can easily be rendered with any of the standard graphics hardware and software tools available. This allows an easy and fast rendering of the object. The speed depends on the number of polygonal primitives that are generated. Marching cubes usually generates a large amount of polygons. This is also a problem for interactive rendering.

Decimation and simplification algorithms for polygonal meshes can be used to achieve fast renderings of such data at the cost of accuracy.

Another group of algorithms, which we will call binary direct volume rendering algorithms, do not generate a polygonal model of the surfaces of the object to visualize. However, they deal with the objects of the volume as if they were just surfaces. The important point is to be able to detect the ray surface intersection as precise as possible. There are mainly two possibilities to define the surface: the volume is segmented and the voxels are labelled according to which object they belong to, or it is determined the intersection point of the isosurfaces with each viewing ray.

The methods mentioned until now reduce the dimensionality of the volume data from 3D to 2D (i.e., surface rendering) before the data is projected to the image. This means that a lot of information is lost during this reduction. Furthermore, there are objects whose surface cannot be easily defined (e.g., tumors) since the value of the object changes along its surface. Therefore, threshold-based algorithms, like marching cubes, fail. This brings us to the last category of techniques which directly project the 3D volume data to the image without the extraction of any intermediate structure (see figure 2.2c).

These visualization techniques, called direct volume rendering, are explained in more detail in the remainder of this chapter, since they are of major concern for this thesis. These techniques consider the volume data as a discretization of a continuous function defined by $f(x, y, z)$. With interpolation, this function is reconstructed from the sampled data (i.e., volume data) for all points in the space within the volume.

As has been mentioned before, 3D scanners measure some physical properties which do not correspond to optical properties needed in a visualization, like color or opacity. Usually in direct volume rendering, a function which maps the properties of the volume data to optical properties is necessary. The optical properties that are defined by this function depend on the optical model used for the visualization. The most common optical models will be described in section 2.4.

To compute an image, the effect of the optical properties must be integrated along each viewing ray. This integration can be done in several ways which produce different visualization algorithms for direct volume rendering.

Although allowing high quality renderings, direct volume rendering has the disadvantage of being computationally expensive. As a consequence, several acceleration techniques in software and hardware have been proposed in the last decades. The most common techniques are described in section 2.8.

2.3 Reconstruction

Volume data is considered to be a rectangular grid resulting from sampling a continuous functions $f(x, y, z)$ using one of the acquisition techniques presented in chapter 1. A value of the measured property is determined for each vertex of the grid. Usually, volume visualization algorithms need to be able to have values of the function f at any point of space. It is usual that the calculation of first, second or higher order derivatives is also necessary. To reconstruct the function f, interpolation is used rather than approximation.

The most common interpolation filters for function reconstruction are nearest neighbor interpolation and trilinear interpolation. Nearest neighbor is fast but inaccurate, while trilinear interpolation is considered

accurate enough but slow to compute. More sophisticated reconstruction filters have been proposed by different authors [Mars94, Mitc88, Möll97b, Theu00]. The main disadvantage of these reconstruction filters is that quality is achieved at the cost of computational complexity.

Usually, there is also a need of reconstruction of higher order derivatives. The gradient vector, which is based on the first derivative, is interpreted as the normal to the isosurface that passes through a point. The gradient is used as the surface normal in the shading models and can highly influence the quality of the results. The most commonly used gradient estimation is central differences, since it has a low computational cost. However, the quality of such a technique is quite low too. Other quite simple filters (e.g., Sobel filters) are also commonly used when better quality results are required with reasonable speed. Like in function reconstruction, more complex filters for gradient estimation have been studied [Möll97b]. The gradient filters compute the value of the gradient in grid vertices. Therefore, they must be combined with some interpolation filters in order to be able to estimate the gradient in any point of the volume [Möll97a].

In binary direct volume rendering of labelled volume data the previous schemes cannot be used. Reconstructing directly from labelled volume data produces staircase artifacts. Several techniques have been proposed to achieve smooth surface visualizations of labelled volume data [Gibs98, Tied98, Neum00].

2.4 Optical Models

In volume rendering, realism is not as important as obtaining meaningful information. However, images that resemble reality help to interpret the scene, since they are similar to what the observer is used to see. To resemble reality, it is important to define how the light interacts with the media that are going to be rendered. In the case of surface based volume rendering, the optical model used is the same that has been used and proposed for common computer graphics rendering. Usually, Lambert or Phong local illumination models are used. Global illumination models have also been proposed for volume data [Yage92a].

The previous models deal with the visualization of objects which are made up just from surfaces. Another widely used model which was introduced by Blinn [Blin82] and extended by Kajiya [Kaji84] sees the volume as a jelly mass formed by different types of particles which differently affect the light passing through differently. The models differ in the level of realism by modelling absorption, emission and/or scattering of light [Max95]. This model is expressed in the so called Volume Rendering Integral (VRI).

The simplest model is a medium containing particles which absorb incoming light and do not emit or scatter light. Given a ray traced from the view point, the absorption model is expressed by the following equation:

$$I(s) = I_0 \, e^{-\int_0^s \tau(t)dt}$$

where $\tau(t)$ is the extinction coefficient defining the rate of light that is occluded per unit length due to absorption of light. $e^{-\int_0^s \tau(t)dt}$ corresponds with the transparency of the material between 0 and s. We will express this transparency as $T(s)$. $I(s)$ is the light intensity at distance s and I_0 is the background intensity when $s = 0$ (i.e., when the ray enters the volume).

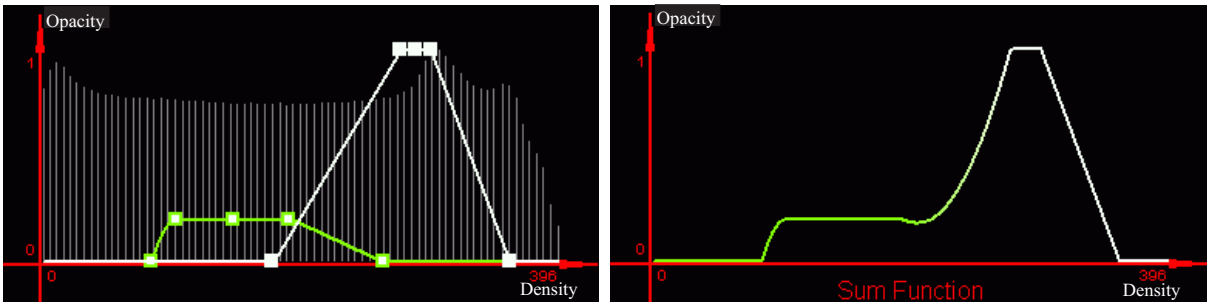


Figure 2.3: Transfer function definition mapping the scalar values of the volume represented by its histogram to color and opacity values. The method is the method presented by König et al. [Cast98]

It can also be that the medium adds light to the ray by emission or reflection of external illumination. The model is expressed as follows:

$$I(s) = I_0 + \int_0^s g(t)dt$$

where $g(t)$ is the source term which contains the intensity emitted and reflected per unit length; $g(t) = C(t)\tau(t)$ where $C(t)$ corresponds with the intensity per unit projected area of each particle. If our model includes both emission and absorption, it is represented by the following equation:

$$I(D) = I_0 T(D) + \int_0^D g(s) e^{-\int_s^D \tau(t)dt} ds \tag{2.1}$$

The function $g(s)$ can include local illumination models like the Lambert or Phong model. This would represent a special case of a general single-scattering term. This is often used, since it produces the visual effect of shaded contours or surfaces. However, these shading effects are unrealistic since they ignore the transparency of the volume data between the light and the shaded point.

Models for realistic single-scattering (i.e., just one reflection event from the illumination ray to the observer) and multi-scattering in a particular medium have also been proposed [Max95, Dach00, Sobi94]. These methods are less often used due to their complexity.

2.5 Transfer Functions

To be able to apply an optical model to a volume, it is necessary that the values of its properties are defined for each point within the volume. The volume is constituted by scalar or vector value which define some physical properties at each location. Usually, these properties do not correspond with optical properties like color, opacity or intensity. The function that maps these properties to optical properties is called transfer function.

How to define a transfer function in order to generate meaningful results is a common problem in volume rendering [Pfis01]. Determining a function depends on many parameters, and it is difficult to predict the results of modifications in the parameters, partly due to the nonlinearity of the VRI.

The most common transfer function maps the scalar values of the volume, using its histogram, directly to color and opacity values [Köni01] (see figure 2.3). More sophisticated transfer functions map the spatial

domain combined with the scalar values and their derivatives [Kind98]. This gives a high dimensional problem. Therefore, the definition of a transfer function to obtain desired visualization results is far from trivial.

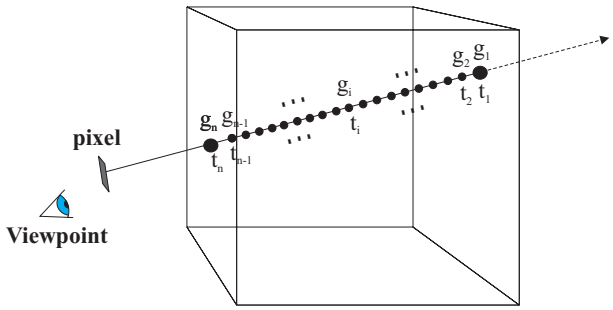


Figure 2.4: Ray sampling to numerically approximate the VRI; g_i is the light intensity emitted at a sample point and t_i represents its transparency.

2.6 Compositing

In section 2.4, we defined the Volume Rendering Integral (VRI) as a continuous integral that should be evaluated along each viewing ray. To evaluate the VRI, it is necessary to use numerical approximation. The numerical approximation of equation 2.1 using the Riemman sum is:

$$I(D) \approx I_0 \prod_{i=1}^{n} t_i + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j$$

This equation is computed by sampling the viewing ray and accumulating the intensity values incrementally (see figure 2.4). The values of t_i and g_i are determined by the reconstruction filter, the transfer function and the illumination model. There are two incremental ways to accumulate the values: back-to-front and front-to-back compositing. The difference between them is the direction in which the ray is traversed to accumulate the values.

The following pseudo code represents both back-to-front and front-to-back compositing.

```
FRONT-TO-BACK COMPOSITING
I = 0
T = 1
i = n
while (T > SmallT and i ≥ 1 ) {
    T = T * t[i]
    I = I + T * g[i]
    i = i − 1
}
I=I + T * I_0
```

```
BACK-TO-FRONT COMPOSITING
I = I_0
for(i = 1; i ≤ n; ++i) {
    I=t[i] * I + g[i]
}
```

Back-to-front compositing is a bit simpler than front-to-back compositing. The advantage of using front-to-back compositing is that when the transparency accumulated along the ray is low enough (i.e., the ray portion is already almost opaque), the algorithm can stop (i.e., early ray termination).

The sampling step-size along a ray determines the quality of the approximation of the continuous integral.

The presented compositing is not the only possibility. There are other operators that can be used, like the maximum intensity projection (MIP). MIP takes as a value the highest within a ray. This method is used in cases where the structures to visualize have the maximum scalar value of the volume. An example is angiography where the contrast medium injected into the vessels corresponds to the highest value of densities.

2.7 Direct Volume Rendering Methods

From the initial direct volume rendering algorithms [Levo88] until recent ones, many different methods have been proposed. In this section, we shortly present and classify the most important ones.

One classification concerns the order in which the shading and the reconstruction are applied: pre-classification and post-classification. Pre-classification means that the optical properties are assigned to each voxel, and afterwards the values of the optical properties are reconstructed to assign colors to the points sampled through the ray. Post-classification means that values are reconstructed in the volume properties domain. For each sample point the reconstructed value is calculated and its optical properties are obtained applying the transfer function to this value. A correspondence can be seen between these two methods and classical shading models. Phong shading corresponds to post-classification and Gouraud shading corresponds to pre-classification.

Another typical classification criterion of the viewing algorithms, not just in volume rendering but in common computer graphics in general, is the space in which the algorithm works. In this way, we can distinguish between image space (e.g., ray casting) and object-space algorithms (e.g., Z-buffer).

2.7.1 Image-Space Methods

Image-space algorithms are also called backward viewing algorithms. These techniques trace rays from the image plane which are sampled and the intensities are composed using front-to-back or back-to-front compositing. The most common algorithms are called ray casting or ray tracing algorithms which correspond to the algorithms with the same name used in traditional computer graphics.

2.7.2 Object-Space Methods

Object-space methods, or forward viewing algorithms, project the voxels onto the screen instead of casting rays.

One of the methods consists of using the Z-buffer algorithm. However, this algorithm is very inefficient and does not allow the implementation of semitransparent objects, since the voxels are projected in an arbitrary order, and compositing is not a linear function.

Another technique consists of pre-sorting the voxels according to the distance to the observer. The voxels are projected afterwards to the screen in a back-to-front or front-to-back manner using the painter's algorithm. This method allows semitransparent objects since the composite values can be calculated incrementally.

The image quality of these algorithms is low if the voxel interdistance is larger than the distance between pixels. One well known method to solve this problem is splatting [West90].

In the splatting algorithm, each voxel is projected to the screen, as if a snow ball, i.e., a radial symmetric interpolation kernel, would be thrown from the voxel position in the direction of the ray. The voxel density is spread to neighboring pixels based on a pre-calculated 2D footprint or splat (i.e., the projection of the interpolation kernel). These values are composed with the values in the image. The original splatting algorithm uses pre-classification which produces a blurring effect. Müller et al. [Müll99] present a technique to perform splatting without blur, where post-classification is used. The bottleneck concerning time is that each voxel must be convolved with a 2D footprint.

2.7.3 Hybrid and other Methods

The shear-warp algorithm presented by Lacroute and Levoy [Lacr94] is an hybrid between image and object-space algorithms. It is recognized as the fastest software renderer until now. The algorithm is based on the shear-warp factorization of the viewing matrix [Yage92b].

In the first step of the algorithm, a base plane is chosen such that it corresponds with the face of the volume data that is most parallel to the viewing plane. Instead of tracing rays, the volume is divided in slices parallel to the base plane. The slices are sheared according to the factorization of the viewing matrix. Afterwards, the slices are projected to the base plane using a back-to-front or front-to-back compositing. The projection is performed using bilinear interpolation. To accelerate the algorithm, the volume is stored three times in run-length encoding along the major viewing directions. The 2D base-plane image is then transformed to the image plane using a warp operation. To resample the image, 2D texture mapping from common graphics hardware can be used. The use of bilinear interpolation results in a degradation of the image quality. Thus, it can result in a very low image quality, if the resolution of the final image is significantly bigger than the volume resolution.

Fourier volume rendering is a volume rendering algorithm which was proposed independently by Dune et al. [Dune90] and Malzbender [Malz93]. This method makes use of the Fourier projection-slice theorem which states that the inverse transform of a slice from the frequency domain representation of a volume yields a projection of the volume in a direction perpendicular to the slice. The first step of this method is to convert the volume data to the frequency domain. For each view, a 2D slice is computed in the frequency domain. Finally, the 2D slice is converted back to the spatial domain to obtain the projection of the volume.

Fourier volume rendering can only produce attenuation-like images (e.g., X-Rays) [Tots93]. Its complexity, however, is $O(N^2 \log N)$ for an N^3 sized volume in comparison to $O(N^3)$ of conventional methods. Therefore, fourier volume rendering can be calculated much faster.

2.8 Acceleration Techniques

Direct volume rendering is still a very time-consuming process and the cost is multiplied if we have to generate more than one picture like in the generation of stereoscopic images or animations. Many acceleration methods have been proposed over the last years [Mroz01, Cséb01]. We subdivide these methods into software and hardware acceleration techniques.

2.8.1 Acceleration Techniques in Software

One of the most commonly used acceleration concept to avoid unnecessary calculations is coherence [Gröl92]. Coherence may occur in several ways: e.g., homogeneous areas in voxel space or the similarity of two consecutive images in an animation.

Pixel-Space Coherence

The acceleration techniques that use pixel space coherence assume that between two similarly shaded pixels, only pixels with a similar color exist, and these pixels can be computed using interpolation.

One approach [Levo90b] generates an initial grid by casting a uniform but sparse grid of rays into the volume data. The image is obtained by interpolating between the resulting colors, and resampling at the screen resolution. Subsequent images are generated by discarding interpolated pixels, casting more rays. Recursive subdivision based on color differences is used to concentrate these additional rays in regions of high image complexity.

Object-Space Coherence

Object-space coherence tries to avoid over sampling in 3D regions having uniform or similar values.

Laur and Hanrahan [Laur91] present a technique based on a pyramidal volume representation to accelerate splatting. An octree is fit to a pyramid given a user supplied precision. The octree is then drawn using a set of splats each scaled to match the size of the projected octree node. This allows a progressive refinement of the image according to the desired tradeoff between quality and speed.

Danskin and Hanrahan [Dans92] propose a method that efficiently detects regions of low variation by employing a pyramid of volumes that encodes the maximum and minimum voxel value in a neighborhood and the distance between these values.

Several techniques avoid sampling in empty space, since empty space does not contribute to the final image. These techniques are called space leaping techniques. The voxels that do not contribute to the final image usually occupy connected regions. These voxels can be joined into macro regions or into hierarchical structures (e.g., pyramids and octrees [Levo90a]). A space leaping approach based on a distance transform was introduced by Zuiderveld et al. [Zuid92]. The distance transform operation [Lohm98] is applied to a binary volume (e.g., segmented volume) and its result is a distance map. A distance map is a volume of the same size as the original volume which, for every background voxel, saves the distance to the nearest object voxel. The distance map can be used to accelerate ray casting, increasing the distance

between samples along a ray when there is empty space. Later on, the distance maps used for space leaping were called proximity clouds [Yage93].

Sobierajski et al. [Sobi95] propose a polygon assisted ray casting for volume rendering. Their approach generates a polygon bounding box from the faces of the cells that contains the external surface of the object. The bounding box is rendered in Z-buffer graphics hardware. The Z-buffer is then used to quickly identify the first voxel which contributes to the result image.

Lorang [Lora01] proposes a technique which efficiently index the volume and, therefore, speed up the ray-casting algorithm.

Inter-Ray Coherence

Yagel and Kaufman [Yage92b] propose a method called template-based volume viewing. Parallel rays are cast into the volume by repeating a sequence of steps specified by a discrete ray. This method exploits the fact that in parallel projection, all the rays have the same slope. If all the rays have the same form, there is no need to reactivate the discrete line algorithm (e.g., Bresenham) for every ray. The ray templates are computed just once. The rays just differ in the exact initial position. Therefore a plane parallel to one of the volume faces guarantees a complete and uniform tessellation of the volume. A mapping from the base plane to the screen plane by warping is needed to achieve the correct image.

Frame-to-Frame Coherence

These acceleration techniques exploit the coherence between successive images in animations. For orthographic projection, Gudmundsson and Randen [Gudm90] present an algorithm which incrementally reprojects the calculated pixels from one view to the next and just calculates in the areas where new features could appear. These methods accelerate binary direct volume rendering. An improvement to the approach is presented by Yagel and Shi [Yage93]. In their approach, not just the pixels' color values are saved, but also the coordinates of the first non-transparent voxel in the so-called C-buffer. Then, these values are reprojected and the rays start at the position that corresponds to the reprojected coordinates. Just a few rays need to be traced from the image plane. In this way, the frame-to-frame coherence is used for space leaping techniques.

Brady et al. [Brad98, Brad97] present a method to accelerate perspective rendering using frame-to-frame coherence. A two-phases rendering is presented. In the first phase, short ray segments are cast and the composite color is computed for the small segments. The segments are divided in levels depending on the distance to the viewer. In the second phase, the levels are composed to generate the final image. The algorithm is based on the reuse of phase one for small translation and rotation of the camera.

The generation of stereoscopic image pairs can be accelerated by generating the second images using the information obtained from the first fully-rendered image [Adel94, He96].

2.8.2 Acceleration Techniques in Hardware

Software-based volume rendering techniques can achieve interactive frame-rates. However, it usually implies a compromise between speed and quality, or parameters that can be changed interactively like the transfer functions.

To achieve real time frame rates without compromising the quality too much, hardware acceleration is necessary. In this section, we introduce hardware acceleration techniques which make use of already existing common graphics hardware (e.g., 3D texture mapping) and dedicated graphics hardware techniques.

3D Texture Mapping

The use of 3D texture mapping for volume rendering was introduced by Cabral et al. [Cabr94]. This technique was presented for non-shaded volume rendering. It is based on using the 3D texture features of graphics hardware like SGI's RE 2 and IR architecture. The volume is interpreted as a 3D texture and loaded into the 3D texture memory of the graphics hardware. Polygonal slices parallel to the viewing plane are re-sampled using hardware-implemented trilinear interpolation. The reconstructed scalar values of the slices can be converted to colors via a look-up table. The slices are then correctly blended using back-to-front or front-to-back compositing. The distance between slices can be chosen freely, thereby controlling the quality of the final rendering.

Extensions to this technique have allowed to generate shaded images [Geld96, West98]. The main drawback of 3D texture mapping is the limited size of the 3D texture memory which may require to divide the data set into bricks. Another problem is that just specific graphic hardware supports 3D textures. Approaches similar to 3D texture mapping, but for low-end PC's using the 2D texture features of some specific graphics boards have also been proposed [Resz00].

Dedicated Hardware Techniques

Several dedicated hardware architectures have been proposed in the last years, but just a few of them have been implemented. VIRIM [Guen94], which implements ray casting, achieves a frame rate of 2.5 f.p.s. for volumes of size 256^3. VIZARD [Knit97] is a system which implements true-perspective ray casting achieving up to 10 f.p.s. for 256^3 volumes. VolumePro [Pfis99] is the first single-chip real time volume rendering system for PCs.

VolumePro is based on the rendering architecture of Cube-4 [Pfis96] developed at SUNY, Stony Brook. This card allows 30 f.p.s. for a 256^3 volumes.

VolumePro implements ray casting for orthogonal views. The algorithm is based on the volume viewing shear-warp algorithm. The volume is projected on a base plane, which is coplanar to the face of the volume closer to the image plane. Then, the resulting base plane image is warped to the image plane. The main difference to the shear-warp algorithm is that it performs trilinear interpolation and allows rays to start at sub-pixel locations which increases the quality. In summary, VolumePro provides high quality real time rendering with compositing, classification with density based transfer functions and Phong shading.

Part I

Virtual Endoscopy System

Chapter 3

VirEn: A Virtual Endoscopy System

> The whole is more than the sum of its parts.
>
> *Aristotle (ca 330 BC)*

3.1 Introduction

Several virtual endoscopy systems have been proposed in recent years. In spite of the fact that all of these systems are constrained to concrete applications, they do have similar components.

In section 3.2, we present a generic framework for the development of a virtual endoscopy system. We call this system VirEn [Vila99]. In section 3.3, other proposed virtual endoscopy systems and their components are discussed. Finally, the current implementation of the VirEn framework and its components are explained.

3.2 Structure of a Virtual Endoscopy System

The main idea of VirEn is that with volume data as input of the system, after some preprocessing, the user can explore the data in a way similar to a real endoscopy or, at least, obtain similar information. In this section, we explain the structure and elements of this system. A global structure of VirEn is shown in figure 3.1. The system basically consists of preprocessing and interaction modules. The modules in the preprocessing group are less time critical, since they are calculated just once for every input data set. Computations carried out by the interaction modules are done on the fly, during user navigation, and therefore execution speed is crucial.

The virtual endoscopy system consists of the following main elements (see figure 3.1):

Acquisition and enhancement of the volume data. This component provides the initial input to the system, which is a 3D image data set. The data is acquired by one of the usual 3D medical imaging modalities (see section 1.2). Frequently, there is the need to apply image processing techniques (e.g., noise removal) to the 3D image to increase the quality of the data and facilitate the following processing steps. The resulting output of this module is an enhanced volume data set.

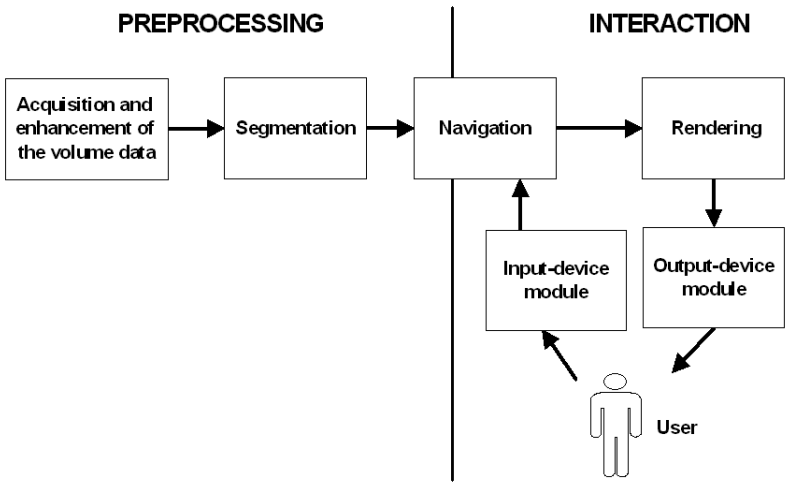


Figure 3.1: VirEn: System overview.

Segmentation. The segmentation module separates an object from its surroundings in a volume data. The result of this process can be data of the same dimension, a subvolume (3D), or data of lower dimensions, such as surfaces (2D) and border lines (1D).

The segmentation module is used for the definition of the object within the entire data volume which the user is interested in (e.g., colon, trachea).

Navigation. The navigation module is concerned with the on-the-fly inspection of narrow tubular structures typical in endoscopy. It includes the interaction of the user to control camera movement, and deals with mapping the input device movements to camera parameter modifications. The user should neither get a "lost-in-space" feeling because of dealing with too many parameters nor a frustrating feeling due to a heavily constrained navigation environment.

A near real time frame rate is also an important goal for a feasible navigation. Imaging devices are producing increasingly higher quality volume data sets, but at the expense of increased storage requirements. In virtual endoscopy, the viewpoint is moving inside the data set and just a little part of the whole data will be seen. The navigation component is taking advantage of this fact to reduce the amount of data that is sent to the rendering module. The navigation module provides the data for the rendering module and also the camera parameters like camera position and orientation.

Input-device module. This module is handling the hardware devices used in the interaction with the user. There are many choices for interaction with a virtual endoscopy system. Different classes of devices range from common desktop hardware (keyboard and 2D mouse), 3D devices with six degrees of freedom and haptic feedback, to a special virtual endoscope device [Hand97].

Rendering. Once the camera position is determined and the data is prepared, rendering has to be performed. Volume rendering techniques can be applied to render the data set depending on the accuracy and the desired frame rate. Because of the segmentation step the data does not necessarily have to be volume data. It is important to point out that rendering should use perspective

projection since the viewpoint is inside the data set. Stereo viewing can also be applied in order to enhance the realistic impression.

Output-device module. This module handles the devices used to present the result of the rendering module to the user. Similar to the input-device module, there are several choices (e.g., monitors, head mounted displays, etc).

We can distinguish two main issues in virtual endoscopy: accuracy and user interaction. Virtual endoscopy can be used in applications like diagnosis and surgical planning and therefore the data must be accurate enough not to lead the physicians to a wrong decision . The accuracy problem concerns acquisition, segmentation and rendering. Another main topic is user interaction. In order to get clinical acceptance, the user must be able to deal easily and fast with the system. This concerns the navigation and the rendering components.

In the remaining part of this chapter we will explain in more detail the most important modules.

3.2.1 Segmentation

Segmentation is a research topic in many areas (e.g., computer vision, pattern recognition, image processing). There is a high degree of complexity in automating the segmentation process. Furthermore, a general, fully automatic segmentation tool is very difficult to achieve. Today the degree of automatization of a segmentation procedure relates inversely proportional to the accuracy of the result. On the other hand a manual-only segmentation is tedious and time-consuming.

Therefore, most of the general segmentation techniques are semiautomatic. The user introduces some clues (threshold, a seed point [Zuck76], contour guiding [Kass87, Mort95], and others) in order to guide the algorithm, which automatically calculates a part of the segmentation. These methods are a compromise between accuracy due to user supervision and being not as time consuming as a manual segmentation. In the VirEn system, the data is segmented using one of the previous cited techniques depending on the type of organ which is going to be inspected.

3.2.2 Rendering

This module includes the rendering techniques for volume data that have been described in detail in chapter 2. In virtual endoscopy, it is mandatory to use perspective projection to get the correct depth cue, as the camera is situated inside the object. Orthographic projection assumes the observer is situated at infinite distance and this cannot be assumed in the case of virtual endoscopy. Therefore, several volume rendering acceleration techniques cannot be used since they are limited to orthographic projection.

Obtaining interactive frame rates and high quality renderings are two contradictory constraints. Surface volume rendering can achieve real time frame rates but it relies on a previous segmentation step. The segmentation produces as result the surface of the object of interest, and it is sometimes very difficult to achieve it in a correct way, even manually. Furthermore, the segmentation step may introduce artifacts that might result in wrong visualizations [Jefr98, Lore96]. On the other hand, using direct volume rendering generates high quality results but the frame rate that it can achieve is low.

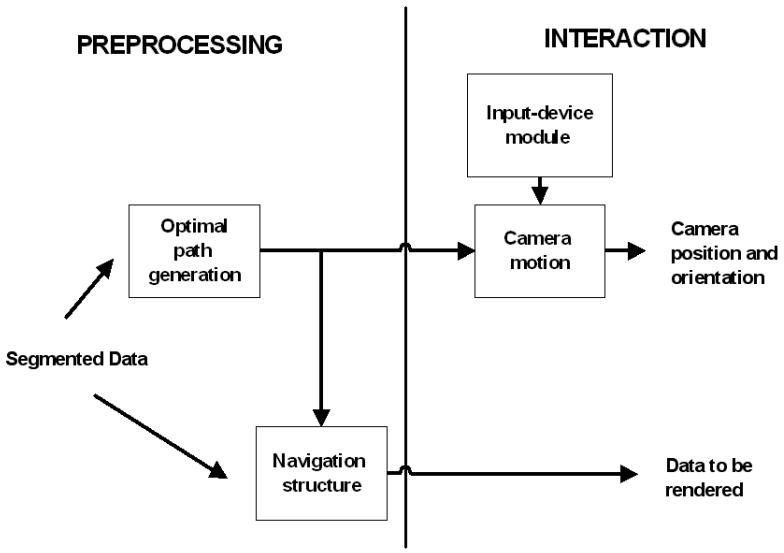


Figure 3.2: VirEn: Navigation module

3.2.3 Navigation

The navigation module deals with 3D interaction and with the generation of data structures that can accelerate the rendering process, using the locality of the viewing frustum within the spatial extent of the data set (see figure 3.2). Navigation in virtual endoscopy has some peculiarities that have to be considered:

- In general, the physicians want to inspect the internal part of an organ, so they are not interested in going through its walls. Nevertheless, special applications like angioscopy of brain vessels might make this option desirable.
- The user would like to have a wide-angle view during navigation, which can be achieved by keeping the camera as close to the center of the hollow organ as possible.

  When the camera position is moving, the camera will usually be pointing towards the end of a tubular structure and not directly to the walls.

If an approximation of the optimal path can be predicted, it could be used for automated navigation or to improve the degree of interactivity.

As shown in figure 3.2, the navigation module has been divided into different components. In the following, these components will be described.

Optimal path generation. The optimal path has to be connected and should reside as close to the center of the hollow object as possible. In order to obtain this path, several approaches (from manual specification to automatic path generation [Lohm98]) can be used.

Camera motion There are different approaches to control the camera in a virtual environment. Although a lot of research effort has been invested in this area, it is not yet fully explored [Hand97, Hinc94, Mack90].

There are basically three groups of user interaction techniques.

- **Planned navigation** is a technique already used in computer animation and robot path planning. Movements along the path are calculated off-line. The user defines a certain number of keyframes where camera parameters are specified. Smooth camera movements between the keyframes are calculated automatically (e.g., using interpolation).

  Another possibility is to automatically calculate the navigation path from a starting point to a target point previously defined by the user.

  The drawback of planned navigation is the lack of interactivity, which may require a tedious amount of work for the user to achieve the desired results.
- **Manual navigation.** With manual navigation, the user has complete control over all parameters of the virtual camera without any constraints. The problem with this technique is the large number of parameters that the user has to control, which can get him easily disoriented.
- **Guided navigation** is in between the two previous techniques. The user has control over the camera parameters but some constraints are added, such as keeping the position of the camera to the optimal path. This implies losing some freedom in the interaction. Nevertheless, this approach avoids the lost-in-space feeling and interactivity is maintained [Galy95]. Guided navigation is actually an extension of both planned and manual navigation. When the motion is too constrained, it is equivalent to planned navigation, and when there are no constraints, it will be like manual navigation.

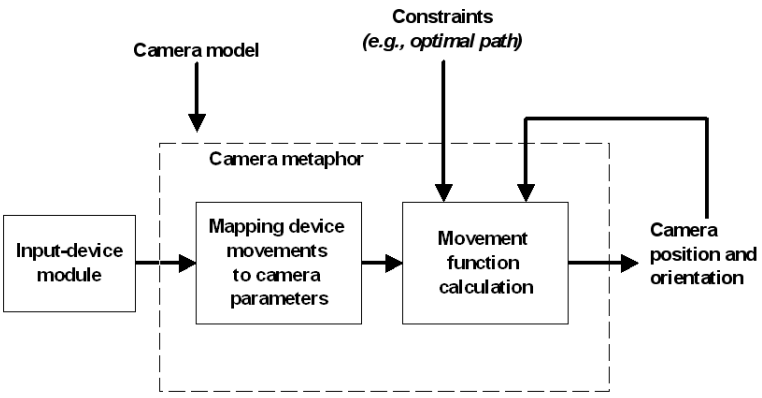


Figure 3.3: VirEn: Camera motion

The parameters that specify the position and orientation of the camera determine the camera model [Druc92]. Some possible camera models include PointLookAt [Blin88] with the position of the camera and up vector; using Yaw/Roll/Pich angles for the orientation plus the camera position, or using quaternions and the camera position.

Viewpoint-manipulation techniques are often called camera metaphors, which usually have a real-world analogy (e.g., flying, walking, ”scene-in-hand”, ”eyeball-in-hand” [Hand97, Hong97]).

Camera motion in VirEn has been defined as shown in figure 3.3. It includes mapping of the device movements to the camera model parameters and can also include the calculation of movement functions. These functions can be used for example to provide smooth transitions or to apply constraints when guided navigation is used (e.g., keep the camera position close to the optimal path or avoid the camera to penetrate the walls using collision detection between the camera and the walls).

Navigation structure. A navigation structure is any data structure used in navigation in order to either accelerate rendering or to guide the camera motion. Navigation structures to accelerate rendering are broadly used in computer graphics applications like architectural walkthroughs and games [Lueb95, Tell91, Yage96a], where polygonal rendering is used. These techniques could also be used in virtual endoscopy, if surface rendering is used.

In the case of volume rendering, using the locality of the endoscopic view and the optimal path generation, subvolumes can be created to avoid having to process the whole volume in the rendering step.

In the case of branching organs, like the trachea, having a tree structure can be useful to decide the navigation path and for accelerating the rendering process.

An endoscopic view just shows the local neighborhood of the camera position in detail. To provide the user with global information on his position within the entire object, a context image is desirable. A sketchy representation of the entire object is sufficient for this purpose. The endoscopic view illustrates the focus area, while the overview illustrates the context area.

Another important option in virtual endoscopy is to provide the user with the possibility to perform measurements (e.g., length, area, volume, distance) of interesting or pathological features.

The presented general framework is the result of investigating the different already existing endoscopy systems.

3.3 Related Work

Much work has been done in the last years related to virtual endoscopy. In this section, some of the developed approaches are described.

Hong et al.[Hong97] describe an endoscopy system which is used for colonoscopy. It is based on surface rendering. They use a modified region growing algorithm and marching cubes as segmentation techniques. Making use of the bent shape of the colon, they employ potential visible sets and portals as navigation structures. As navigation metaphor, they use guided navigation with physically-based camera control. A potential field is calculated using a distance transformation. This potential field is an attracting force to the target point while the walls act as repulsive force. A friction field is added in order to control the speed. The same research group improved the system in a later article [You97] using direct volume rendering, which is accelerated by a distance transformation previously calculated for navigation. The

algorithm is parallelized for acceleration. Using similar techniques a virtual ventricle endoscopy system is proposed by Bartz and Skalej [Bart99].

Yagel et al. [Yage96b] propose a system for the training of surgeons in virtual sinus surgery and for teaching in virtual endoscopy. Volume rendering is done using splatting [West90] for high quality rendering and 3D texture mapping [Cabr94] in hardware is used for fast movements. They use special devices for interaction, like an endoscope device.

Geiger and Kikinis [Geig95] present a virtual endoscopy system used in bronchoscopy for training and simulation of complicated cases. In order to segment the organ they use statistical segmentation or thresholding, as well as manual 2D contour editing. A Delaunay volume and surface reconstruction from the contours is done to be used in surface-based rendering. For navigation, a cylinder metaphor with the degrees of freedom of a real endoscope is used. The camera is constrained to rotate around the axis of the cylinder and to rotate from 90 to -90 degrees around another axis perpendicular to the cylinder axis. Backward and forward movements are also allowed. Additionally, the system detects wall penetration in which case a corrective motion is applied.

Two different virtual endoscopy systems are presented by Satava et al. [Sata97]. They deal with both direct volume rendering and surface rendering. They also use two navigation methods: one is a planned navigation method used for direct volume rendering. For surface rendering, they use a fly-through method or a manual navigation method. In order to enhance the visualization, stereoscopic vision and 3D devices are used.

Darabi and et al. [Dara97] use volume rendering with ray casting and planned navigation. They suggest that stereographic projection gives a more realistic view since perspective projection fails because symmetric objects appear asymmetric near the boundaries.

In the proposal done by Shahidi et al. [Shah96], segmentation is realized by thresholding, region growing and marching cubes. They utilize both manual navigation and planned navigation (i.e., keyframes), as surface rendering is used in this approach.

Paik et al. [Paik98] describe a planned navigation approach using path planning. One of the main issues of this article is the automatic calculation of the medial axis using a thinning algorithm.

Mori et al. [Mori96] propose the usage of surface rendering to implement a bronchoscopy system. A simple marching cubes algorithm is used for segmentation. A thinning algorithm generates a tree structure to obtain a skeleton. This skeleton is used to select the branch corresponding to the current viewpoint.

Brady et al. [Brad98, Brad97] describe a special type of perspective volume ray tracing which makes use of coherence between frames. A subvolumes system is utilized in order to deal with the huge data sets that the current imaging techniques are producing.

Wan et al. [Wan99b] present a space leaping method for virtual colonoscopy to accelerate direct volume rendering.

Wegenkittl et al. [Wege00] describe a system for planning and training for a blind endoscopic transbronchial biopsy. Moving the camera through a pre-calculated central path, a sequence of cubic environment maps is computed. Afterwards, the physician explores the organ in realtime. The camera is restricted to the path positions where the environment maps have been calculated, but it can point in an arbitrary direction using image-based rendering techniques. Different layers of maps are calculated, in

order to look through the lumen of the trachea. The cubic environment maps are calculated by using volume rendering.

Hietala and Oikarinen [Hiet00] introduce a visibility computation method to achieve interactive surface rendering for virtual colonoscopy. The triangles are calculated during the navigation using an optimized ray-casting algorithm.

3.4 VirEn Prototype

A prototype based on the VirEn framework has been developed. This prototype implements mainly the navigation and the rendering modules of the VirEn system. Figure 3.4 presents a diagram illustrating the processes that have been developed. We assume that the input volume data to the system has been already labelled by some of the segmentation methods presented in section 3.2.1.

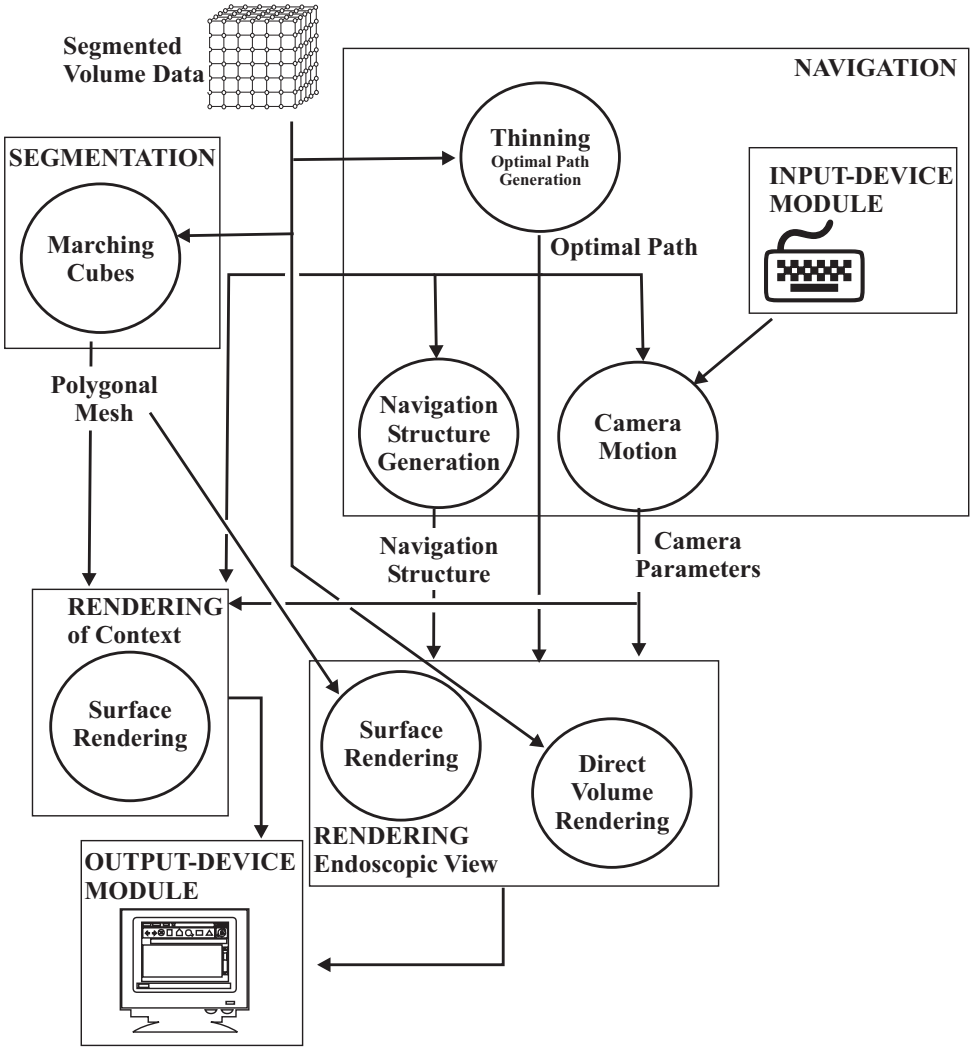


Figure 3.4: VirEn: Prototype structure

The marching cubes algorithm [Lore87] has been implemented to extract a polygonal surface of the segmented object or/and of an implicit surface defined by a threshold. The polygonal mesh allows to generate a fast visualization for the context view (see figure 3.5B).

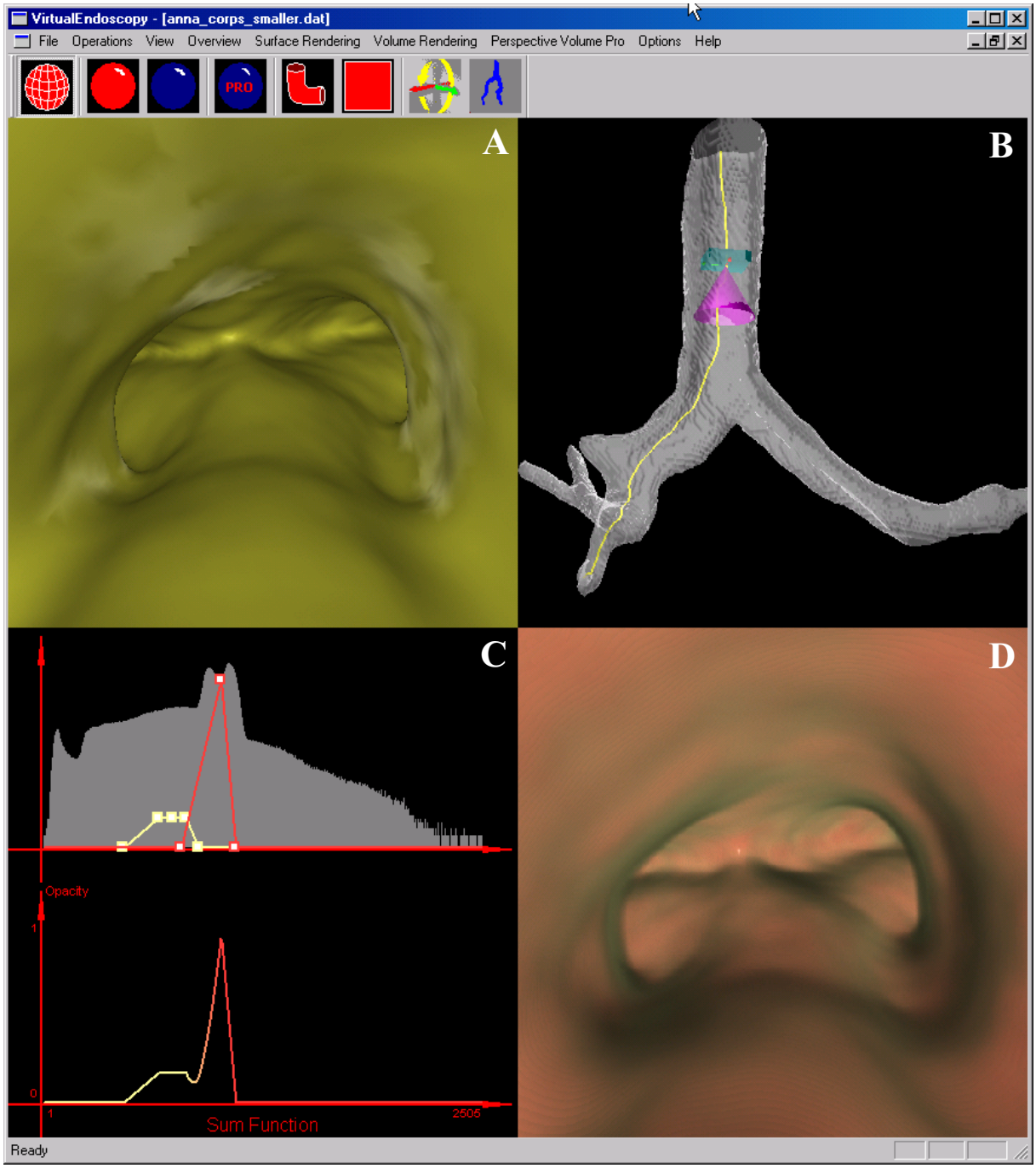


Figure 3.5: VirEn Prototype: Application snapshot, where the windows show: A) endoscopic view using surface rendering, B) context view, C) transfer function, D) endoscopic view direct volume rendering.

The number of polygons generated can be bounded obtaining a less accurate approximation of the surface, but ensuring fast rendering. The resolution of the volume is reduced in order to reduce the amount of triangles.

For the optimal path generation, a topological thinning algorithm has been chosen. A detailed explanation of the algorithm is presented in chapter 4. The camera motion follows the principles of guided navigation with free rotation movements. The position of the camera is restricted to the calculated optimal path. The user can also change to a free navigation metaphor to move the camera away from the path.

We concentrated on achieving high quality renderings since the main goal of the prototype is to investigate visualization techniques to be used for diagnosis. Therefore, direct volume rendering has been implemented. Volume ray casting with the absorption and emission model as presented by Max [Max95], and trilinear interpolation is used. A new space leaping acceleration technique for the ray casting step has been investigated, which will be presented in chapter 5. A new technique to obtain perspective high quality volume rendering using hardware acceleration (i.e., VolumePro) has also been implemented. This technique is presented in detail in chapter 6.

A snapshot of the current application can be seen in figure 3.5. Window 3.5A shows an endoscopic view using surface rendering of the polygonal mesh obtained by marching cubes. This allows fast navigation but the quality is poor. Direct volume rendering, whose results are shown in window 3.5D, is used to get a high quality visualization. Window 3.5C shows the transfer function which is used in direct volume rendering. The specification of the transfer function is based in the method presented by Castro et al. [Cast98]. Window 3.5B is the context view with the organ surface, the path, the camera position and an iconic representation of the camera to illustrate the current position and orientation.

Chapter 4

Optimal Path Calculation

> I knew him when he was nothing and he hasn't changed a bit
>
> *Tom Waits (1976)*

4.1 Introduction

In the previous chapter, we presented that a crucial precondition for intuitive navigation in the described virtual endoscopy framework is to obtain the optimal path. The optimal path is used as a guiding element for the navigation and to generate data structures to accelerate the rendering.

The optimal path has to be connected and has to reside as close to the center of the hollow object as possible. In order to obtain this path, several approaches (from manual specification to automatic path generation) can be used.

An automatic way to generate this path is to calculate the so-called skeleton. One definition of a skeleton is the locus of points that are geometrically centered with respect to the object boundary.

Usually, methods for the generation of the optimal path require a segmentation step first, as the input to these approaches are binary volume data-sets (i.e., the points within the object have value 1 and the background points have the value 0).

There are basically two techniques to extract a skeleton in discrete space [Gagv97]:

Topological Thinning is an operation that iteratively peels the object layer by layer without destroying its topology, i.e., preserving its Euler number [Lohm98]. The basic procedure of the algorithm is the deletion of simple points. A simple point is a point with neighborhood characteristics that ensure topology preservation of the object after the point is deleted. Testing the simple point characteristic is a computationally expensive operation. Therefore, the topological thinning approach is time-consuming. On the other hand, this algorithm can be parallelized and the simple point calculation can be avoided in some cases. Thinning has the advantage that it has a mathematical foundation and the connectivity preservation can be proven. In section 4.2, this approach will be discussed in further detail.

The Distance-Transform Method is based on the fact that the center of an object coincides with points having maximal distance to the borders. A distance transform method converts the binary data image into an image containing for every object point the minimum distance to the nearest background point. The local maximum of the transformed image represents a skeleton point. These methods have the disadvantage that they are not topology preserving. Post-processing must be applied to achieve topology preservation [Bitt00, Zhou98]. It is computational however less expensive than the thinning operation.

In the next section, we describe the optimal path extraction algorithm that has been used for the implementation of VirEn [Vila99].

4.2 Topological Thinning

The approach we use is an extension of the thinning technique described by Ma and Sonka [Ma96]. We define a good skeleton as the skeleton which is kept in the center of the hollow structure and preserves the connectivity of the original object. Therefore, the topological thinning approach is used despite of its computational effort. As the path generation can be done in a preprocessing step, execution time is less crucial. A thinning algorithm is an iterative process which deletes points with a special neighborhood characteristic, called simple points or border points [Lohm98]. It can be proven that deleting simple points preserves the Euler number and therefore the connectivity of the object.

In order to be geometry preserving (e.g., avoiding that a line is reduced to a point), a simple point is not deleted if it does not meet special geometric constraints. Geometric preservation is quite vaguely defined, and the different thinning algorithms differ in the way in which they define these geometric constraints.

The naive sequential thinning algorithm deletes one simple point at a time and it is quite straightforward to show that it preserves connectivity. A parallel thinning algorithm deletes a set of points in each iteration. This enables its simple parallelization and acceleration. Proving that a parallel thinning algorithm is topology preserving can be tricky. It has to be shown that when two adjacent points are deleted they can be deleted in a sequential way without affecting the simple point condition.

Various approaches on parallel thinning are described in the literature [Gagv97, Kong93]. Our thinning algorithm has been based on the fully parallel 3D thinning algorithm proposed by Ma and Sonka [Ma96]. Fully parallel means that the algorithm is applied in parallel to all the object points of the image.

This algorithm, instead of calculating the simple point condition, checks the 26-neighborhood of the points with specific templates. The templates contain object points, background points and “do not care” points, which can go either way.

The templates are obtained by rotations of the 4 so-called template cores *(A, B, C, D)* shown in figure 4.1. A template belongs to class A, B, C, or D if it has been generated by the template core A, B, C or D respectively. Checking the 26-neighborhood is not sufficient to preserve connectivity and several templates must check extra points in an extended neighborhood. In figure 4.2, the six templates of class A (figure 4.1) and the extra tests needed in those templates are illustrated.

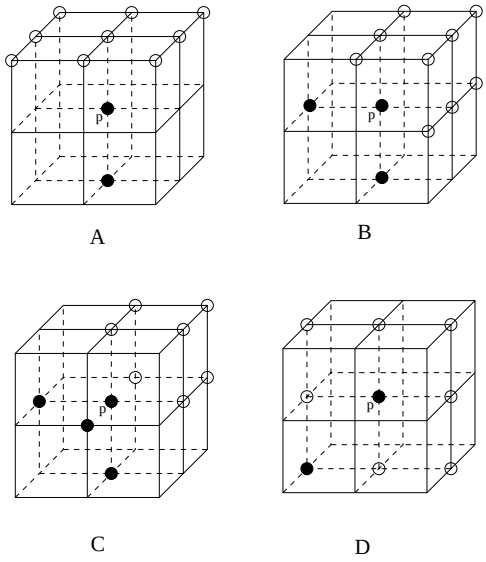


Figure 4.1: The template cores of the parallel thinning algorithm [Ma96]: the unmarked points are ”do not care” points, white points are background points and black points are object points. For the D core template, p must be a simple point.

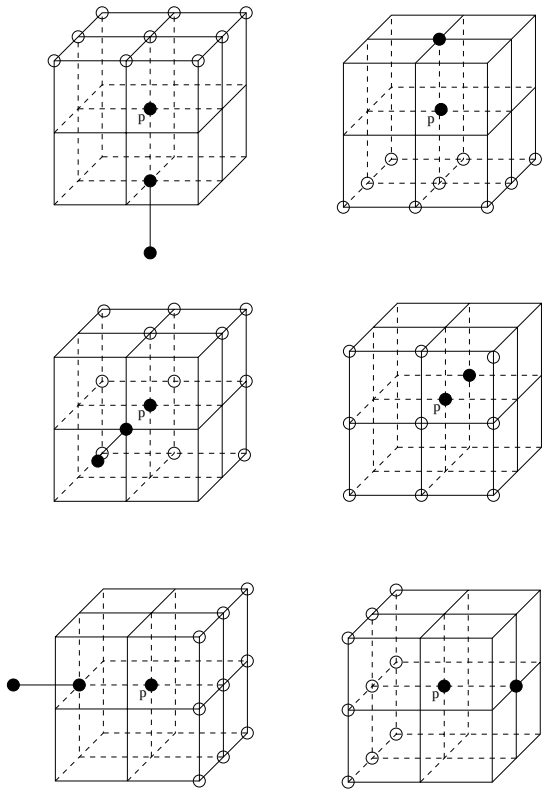
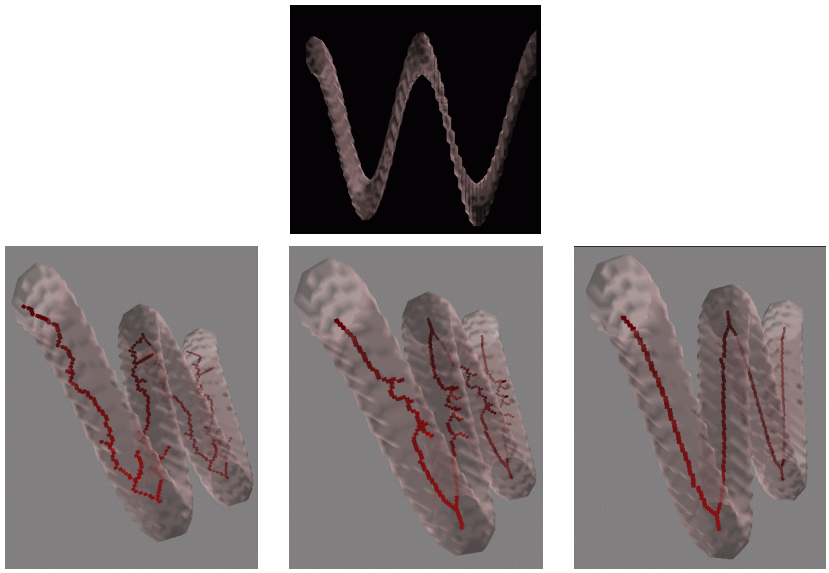


Figure 4.2: The six templates for deleting border points obtained from the core template A.

In our implementation, the 26-neighborhood of a point and the templates are represented by masks. The masks are a chain of 27 binary values (26-neighborhood and the point) where values are 01 for object points, 10 for background points and 11 for "do not care" points. A binary operation *(AND)* and a comparison are enough to find out whether a point fulfills a template condition or not. In some cases tests in an extended neighborhood have to be done. This fact increases computation time. Nevertheless, only the templates generated from the template core D (figure 4.1) require the time-consuming simple point test.

The algorithm by Ma and Sonka presents some problems. One problem arises when a point that fulfills the conditions of the class D templates is deleted in parallel with some of its adjacent points. It should be proven that they can be deleted in a sequential way without breaking the class D template conditions and consequently are simple points. This cannot be proven without restricting the simple point condition to a subgroup of all possible cases. The effect of this restriction can be seen in figure 4.3a compared with figure 4.3b. In figure 4.3b, the templates of class D are treated in a separate iteration. Although this partly loosens up parallelism, the results are better, since the skeleton is smoother (see figure 4.3b). Therefore, it was decided to use this last approach.



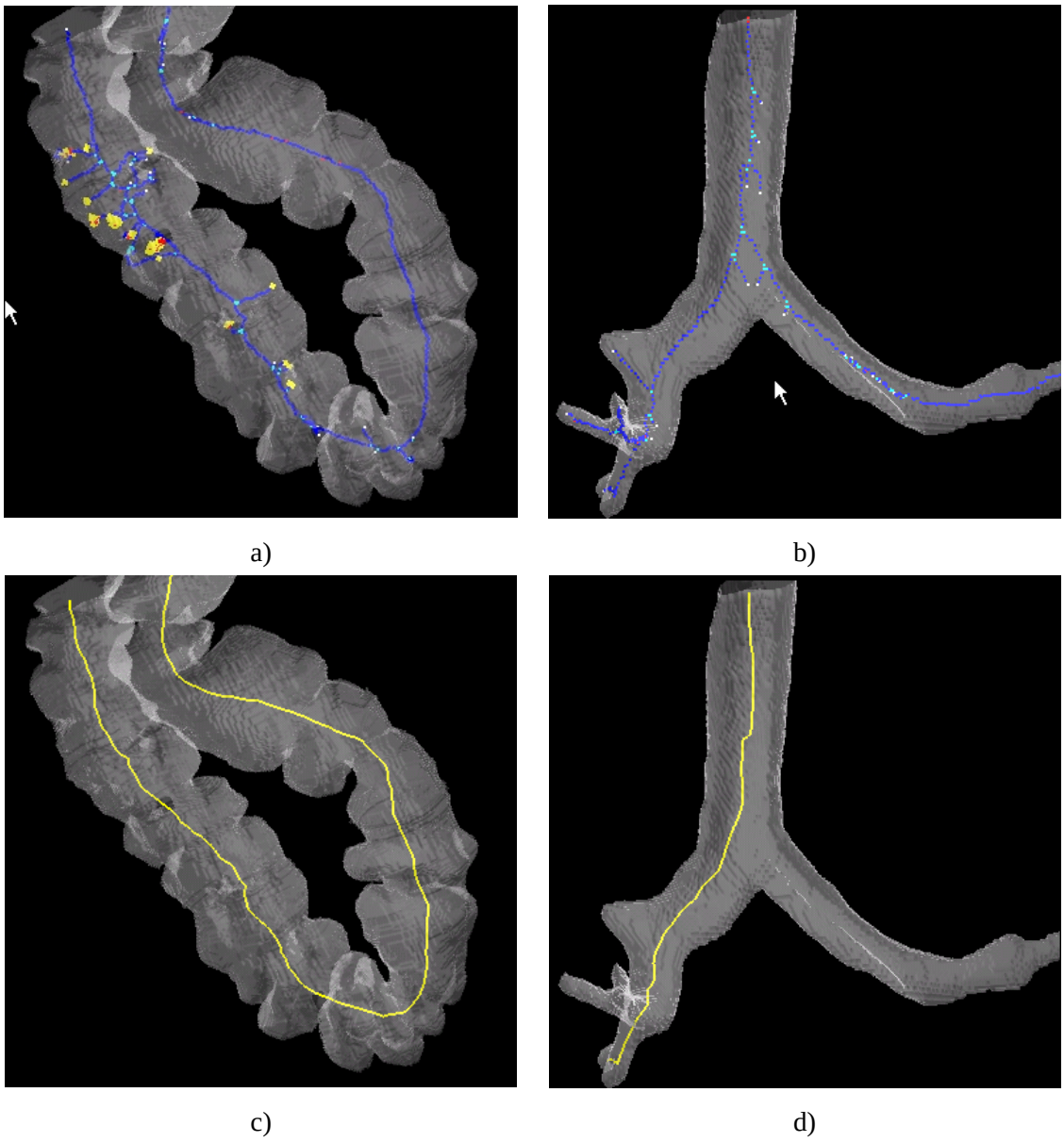
a) b) c)

Figure 4.3: Comparison between the different modifications applied to the thinning algorithm proposed by Ma and Sonka [Ma96]: in the top the artificial data set used for the comparison; **a)** *left* original algorithm [Ma96] result. **b)** *middle* not fully parallel algorithm result. **c)** *right* the same algorithm as *b* plus avoiding orientation dependencies.

Another problem of the original algorithm is that the extended neighborhood tests are not symmetric and therefore the algorithm is orientation-dependent. The algorithm does not treat borders of one side of the object and borders of the opposite side in the same way. This problem is illustrated in figure 4.3b. Although the object represented is symmetric, the thinning result is not. This orientation dependency can be weakened by permuting the orientation of the extended templates in each iteration of the algorithm

(see figure 4.3c). It is straightforward to see that this change does not affect the mathematical verification of the connectivity preservation presented by Ma and Sonka [Ma96].

Two skeletons obtained using real data sets are shown in figure 4.4a and 4.4b. Figures 4.4a and 4.4c correspond to a CT data set of a colon segment with resolution 198x115x300 and figures 4.4b and 4.4d correspond to a CT data set of a trachea with resolution 292x136x114.



a) b)

c) d)

Figure 4.4: Skeleton (**a** and **b**) and optimal path (**c** and **d**) obtained by the topological thinning together with a transparent surface of the binary segmented data. (**a** and **c**) CT data set of a colon segment. (**b** and **d**) CT data set of a trachea.

The color coding shown in the skeleton of figure 4.4a and 4.4b corresponds to a topological classification of the points [Lohm98] where blue corresponds to curve points, yellow to surface points, cyan to curves intersection points (junction points), white to border points and red to those points that could not be classified in any of the previous categories.

4.3 Optimal Path Extraction from the Skeleton

Since real objects are not regular, the skeletons are usually noisy and have some small branches (see figure 4.4 top). This is due to geometry preservation. Therefore an algorithm must be applied to remove these branches.

The skeleton resulting from the algorithm described in section 4.2 is a sequence of 26-connected voxels. We define the best path between two points as the shortest 26-connected path between two end points. Currently, the user has to define the two end points that should be joined and then the path is calculated automatically using Dijkstra's shortest path algorithm (see figures 4.4c and 4.4d). The topological classification is used to detect bifurcations and end points.

Chapter 5

Cylindrical Approximation of Tubular Organs

> Inventions have long since reached their limit, and I see no hope for further development.
>
> *Julius Sextus Frontinus (40-103 A.D.)*

5.1 Introduction

In most applications, dealing with medical data implies that the quality of the rendering results is a major concern. Unfortunately, the volume rendering methods which produce high quality images are also computationally expensive.

There are basically two methods to visualize volume data, as has been described in chapter 2, i.e., surface-based techniques and direct volume rendering. Surface-based implies the loss of quality and information due to the segmentation step. Furthermore, as just the surface of the organ is visible, there are medical procedures which cannot be performed in a surface based model (e.g., decide whether a polyp is benign or not). Moreover, the surface volume rendering looks artificial especially in places where the polygons are near the view point (e.g., narrow tubes).

Therefore, direct volume rendering is used when high quality is a main issue. This technique produces a projected image directly from the volumetric data without using a segmentation step or intermediate constructs. Due to the high computational cost of these techniques, several acceleration methods have been proposed. However, most of these methods imply a compromise between speed and image quality. Just a few preserve quality while speeding up the algorithm (e.g., space leaping techniques).

Currently, most of the systems which use high-quality direct volume rendering for virtual endoscopy compute an off-line movie sequence.

In this chapter, an algorithm [Vila00] that generates cylindrical structures for tubular shaped organs (e.g., colon, vessels) is presented. These structures roughly approximate the organ cavity. These structures are used to accelerate high quality direct volume rendering for virtual endoscopy using space leaping.

The next section gives a brief description of related work done in this field; in section 5.3, the outline of our method is presented. Section 5.4 describes the algorithm to generate the cylindrical structures. In section 5.5, the algorithm that uses the cylinder structure to accelerate volume rendering is described. The results are presented in section 5.6.

5.2 Related Work

Space leaping avoids the processing of those voxels that are not contributing to the final image (see section 2.8). A good overview of these techniques can be found inŠrámek's PhD thesis [Sram96].

Zuiderveld et al. [Zuid92] presented a space leaping approach based on the calculation of a distance map. The distance map is usually a volume data of the same size as the original volume which saves for every background voxel the distance to the nearest object voxel. The distance map can also be computed to save for every object voxel the distance to its nearest background voxel. Wan et al. [Wan99b] presented a distance transform approach for virtual colonoscopy which accelerates ray casting, increasing the distance between samples along a ray when there is empty space.

Sobierajski and Avila [Sobi95] propose a polygon-assisted ray-casting for volume rendering. Their approach generates a bounding polyhedron that contains the external surface of the object. The polyhedron is rendered with Z-buffer graphics hardware. The Z-buffer is then used to quickly identify the first voxel along the ray which contributes to the result image. Similar space-leaping approaches where introduced by other authors [Wan98, Wan99a, You97]

In our approach, we use a combination of the approach proposed by Sobierajski and Avila and distance transform techniques. In virtual endoscopy, the camera is situated inside the organ. We generate a structure formed by cylinders which approximates the organ cavity. This structure is used to accelerate ray casting by identifying the ray parts that are traversing voxels inside the cavity of the organ, and therefore do not contribute to the final image.

Generating cylindrical structures from volume data has been described before by other authors. Puig et al. [Puig97] use cylindrical structures as symbolic trees of blood vessels. These structures give a sketch representation of the vessels, and they are used, for instance, to identify changes in the diameter of the vessels (i.e., to detect a stenosis).

5.3 Method Overview

The method presented in this chapter is proposed for virtual-endoscopy applications where the organ has a tubular shape and the physicians want to visualize the organ surface and its vicinity (i.e., the organ wall). This situation arises in many endoscopy applications, like colonoscopy or bronchoscopy.

Our algorithm is based on the definition of a safety region around the physical organ wall. This safety region will be called the wall interval of the organ. We define the wall interval of an organ as the approximated organ surface (obtained from a segmentation step) together with two tolerance values (see figure 5.1). The tolerance values are two distances from the approximated surface, one in the direction of the organ cavity and one pointing to the outside of the organ. The physical organ wall to be visualized

must be enclosed within the wall interval, since only what is within the wall interval will be rendered. The tolerance values of the wall interval are currently given by the user as parameters of the presented method.

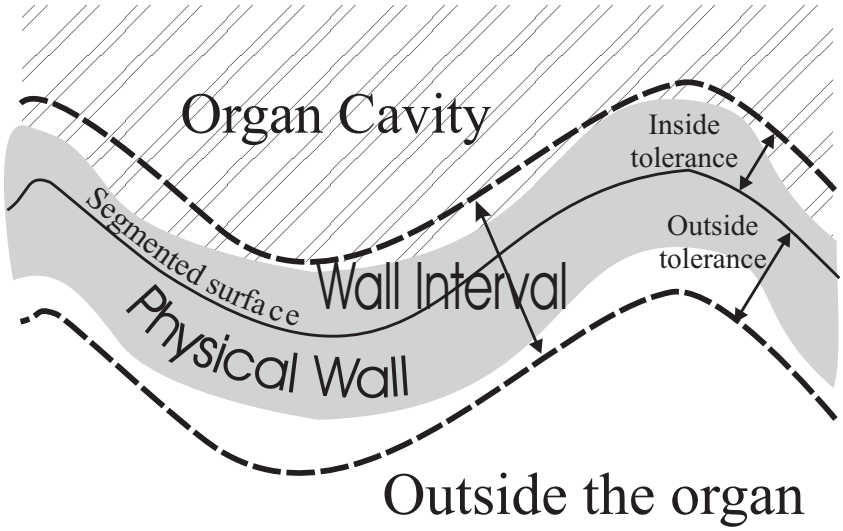


Figure 5.1: Sketch representation of a 2D cross-section of an organ. It shows the wall and the cavity of the organ together with a possible wall interval definition.

The algorithm can be divided into the following steps:

1. Segmentation of the organ to be inspected (e.g., colon, trachea).
2. Calculation of the distance map and the skeleton of the segmented organ.
3. Cylindrical approximation of the organ cavity.
4. Accelerated direct volume rendering using ray casting.

The segmentation is applied to the volume data to identify the organ and to obtain a binary volume. The segmentation does not have to be highly accurate. The algorithm will correct the segmentation error using the tolerance values of the wall interval (see figure 5.1). Therefore, the tolerance values must be an overestimation. They should include the wall thickness and the possible error generated in the segmentation step.

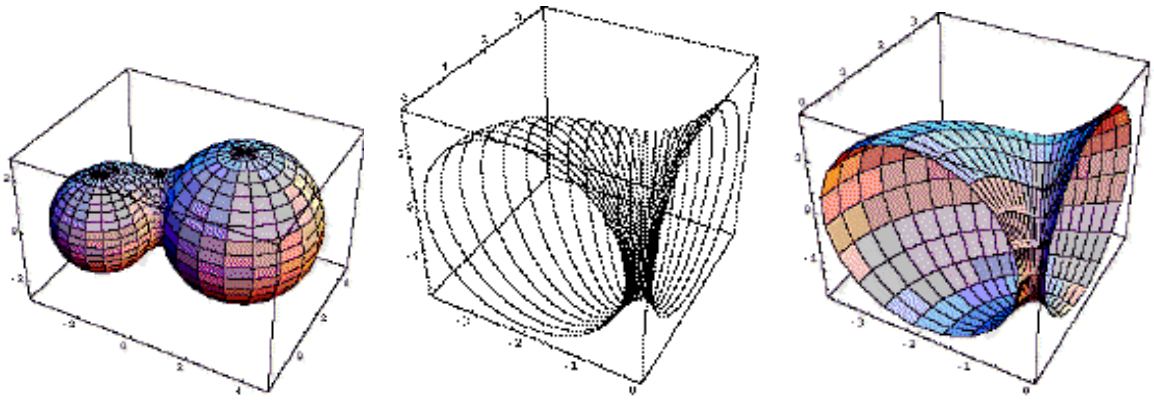
Using the segmented volume, the organ skeleton is computed. A skeleton is the locus of points that are geometrically centered with respect to the border. The skeleton is also used to find the optimal camera path which will be used for the camera-navigation metaphor and therefore for the 3D interaction.

The cylindrical approximation of the organ cavity and the accelerated direct volume rendering will be discussed in detail in the next sections.

5.4 Cylindrical Approximation

In this section, the algorithm to generate the cylindrical approximation of the organ cavity is described. The input of the algorithm are the skeleton and the distance map of the segmented object. The distance map contains for each object voxel the minimum distance to the background.

The main idea of using a cylindrical approximation of the organ started with canal surfaces [Palu97], which are commonly used in geometric modelling. A canal surface is the envelope of a 1-parameter family of spheres, whose radius and position change continuously. The envelope of the spheres is defined as the union of all circles of intersection of infinitesimally neighboring pair of spheres (see figure 5.2). The envelope represents the surface of the object.



a) b) c)

Figure 5.2: Canal Surfaces [Palu97]: a) 1-parameter family of spheres b) Circles of intersection c) Canal surface

Given the skeleton and the distance map, a piece-wise continuous canal surface can be defined. The family of spheres is defined with the centers on the skeleton and the radii equal to the minimal distance to the nearest background voxel.

Using ray casting as rendering technique, we would use the canal surface as a bounded structure to calculate the parts of the viewing rays that cross the cavity of the organ. Unfortunately, canal surfaces require an expensive ray-intersection calculation. Therefore, we approximate the canal surface by a set of primitives with a fast ray-intersection, i.e., cylinders. Apart from cylinders, it is also possible to use other approximations, such as generalized cylinders, but the overhead due to more costly intersection calculations would increase.

The cylindrical approximation to be used for space leaping should fulfill the following requirements.

1. The cylindrical structure is entirely enclosed in the approximated organ and does not intersect the wall interval of the organ.

2. It should have a limited number of cylinders. With a large number of cylinders the intersection procedure applied to the structure will be too expensive.

3. The cylinders should be as long as possible to enable long jumps, especially for central rays. This is quite advantageous in the typical case when the camera is looking straight into the organ cavity.

4. The cylinders should have large volumes to cover a considerable portion of empty space within the organ cavity.

5. The structure shall closely approximate the organ cavity. The intersection point of a viewing ray with the cylinder structure shall be close to the wall interval of the organ (i.e., close to where the sampling of the volume data must start).

The cylinders do not have to be connected and they can also overlap. However, overlapping reduces efficiency in the intersection calculation.

A cylinder is defined by a radius and an axis, given by position, direction and length. These parameters are assumed to be variables of an optimization problem to find the optimal cylindrical structure. This optimization problem should be solved using characteristics 1-5 as optimization criteria. However, the optimal solution is complex and time-consuming to achieve since there are too many variables. Therefore a heuristic approach has been used.

The goal is to have bounded cylinders that cover as much as possible of the organ cavity. We use the skeleton (see chapter 4) as a base to define the cylinder axes. Keeping the axes in the center as much as possible will give fewer and larger cylinders. Once the cylinders' axes have been defined, we define the radius of the cylinders using the distance transform.

The creation of the cylinder structure can be divided into two consecutive steps:

1. Cylinder-axis definition.

2. Cylinder-radius definition.

5.4.1 Cylinder-Axis Definition

The axes of the cylinders will be generated based on the skeleton. The skeleton obtained by either topological thinning or resulting from the distance transform is a set of 26-connected voxels. Based on the skeleton, a central path is defined. For simplicity, we deal with paths which do not have branches. An ordered sequence V_i of voxels is obtained. Two consecutive voxels are 26-connected and there is no cycle, so a voxel is represented at most once. The central path is then a polyline whose edges are the junction of the center points, P_i, of consecutive voxels.

This path is usually highly folded and noisy. In order to reduce the high frequencies and to smooth the path a low pass filter is applied. If m is the kernel dimension and n is the number of path points, with $0 \leq i \leq n-m$, then

$$P_i = \sum_{h=0}^{m-1} P_{i+h} * w_h \;\; where \;\; \sum_{h=0}^{m-1} w_h = 1$$

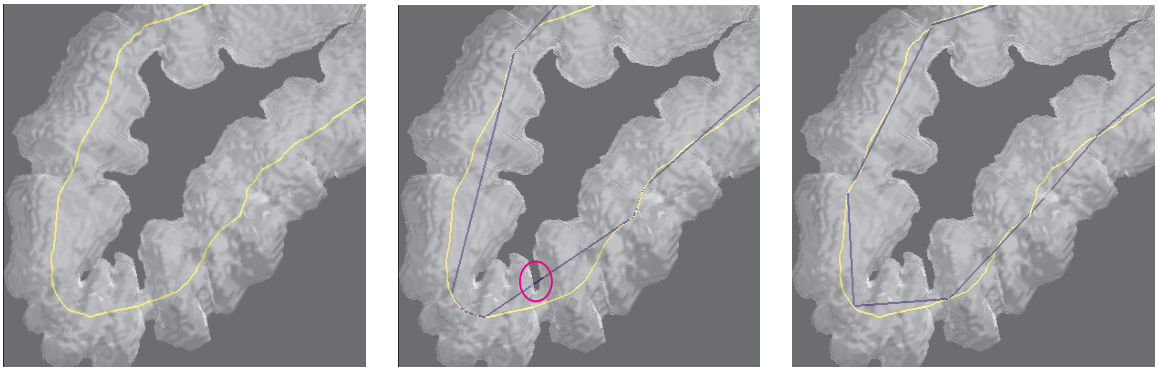
The weights w_h of the kernel are determined using a tent filter. Other low pass filters can also be used.

The maximum variation of the new path compared with the old one depends on the kernel size. If the kernel size increases, then the path is smoother but it also moves from the center of the hollow organ.

After filtering, we have a smooth path represented by a polyline with a large amount of small edges. The polyline is approximated by longer edges to be used as axes for the cylinders. Furthermore, the variation from the original path has to be controlled in order that the axes are not moving outside the organ.

An approach to create the cylinder axes is undersampling the central path by taking into account the curvature of the path. In the areas of the polyline with low curvature the axes can be long and in the areas with high curvature we need more but shorter axes. The number s of cylinders that the structure shall contain is defined by the user. The curvature in the different points of the central path is computed. Then the first and last point of the central path together with the $s-1$ points with the highest curvature are joint in the path order to create the axes.

This method has the problem that s is not easy to estimate. The axes are also not well distributed since they are concentrated in the areas where the curvature is high. Furthermore, there is no control of the deviation between the original path and the axes. Therefore, it can happen that the axes are partially outside the organ if s is not large enough (see figure 5.3b).



a) b) c)

Figure 5.3: Surface of the segmented organ together with the cylinder axis generation based on the central path. **a)** Central path. **b)** Central path, cylinder axes generated from curvature information. The circle shows an area where an axis is outside of the organ. **c)** Central path, cylinder axes generated according to distance errors.

We use another approach which prevents the axes to move outside of the organ. A distance error ε is specified as the maximum distance difference between the original path and the cylinder axes. The error ε is defined as a portion of the minimum distance from the central path to the wall interval of the organ. In this way, the axes are generated ensuring that they will stay within the organ cavity (see figure 5.3c).

The algorithm treats the central-path points sequentially (see figure 5.4). Sub-paths of the central path are consecutively used to generate the axes. It is ensured that the distance difference between a sub-path and the corresponding axis is not larger than the defined error ε.

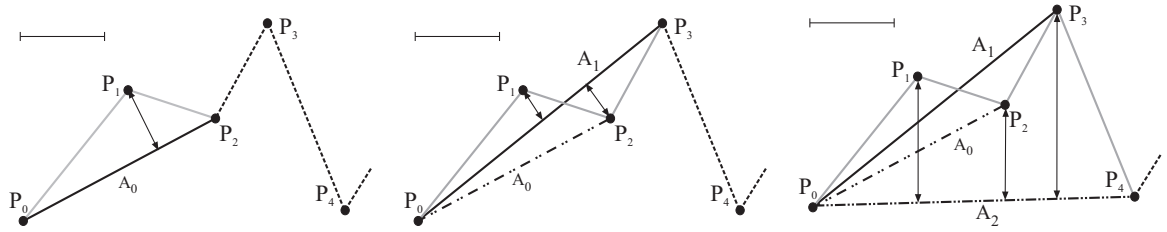


Figure 5.4: Illustration of the axes generation algorithm using the distance error information. A_j is the axis generated in iteration j; P_i denote the central-path points. The central path is drawn as a discontinuous line and the current sub-path is painted in grey. Three iterations of the algorithm are shown. In this example, the final valid axis would be A_1.

A sub-path is a sequence of consecutive points of the central path. An axis A_j is created joining the first and the last point of the current sub-path. $\varepsilon_{i,j}$ is defined as the distance of the point P_i to the axis A_j. If for each point P_i of the current sub-path the distance $\varepsilon_{i,j}$ to the axis A_j is smaller than ε, then the next point of the central path is added to the sub-path. This process is repeated until an iteration is reached, in which there exists a P_i whose $\varepsilon_{i,j}$ is greater than ε. Then, the axis generated in the previous iteration, A_{j-1}, is defined as a valid axis. The sub-path is erased. The next current sub-path is defined by the last point of the axis A_{j-1} and its next two consecutive points in the central path. Initially the current sub-path contains the first point of the central path together with the next two consecutive points. If in the first iteration $\varepsilon_{i,0}$ is already greater than ε, then the valid axis is created between the first and the second point of the sub-path.

In figure 5.3, we observe that with the method that uses error distance ε the resulting axes are better distributed along the path than with the curvature based method.

5.4.2 Cylinder-Radius Definition

Once the axes for the cylinders have been defined, the radii of the cylinders have to be specified. Every cylinder axis is sampled with a rate smaller than the voxels distance. For each sample point, the distance map is used to determine the distance to the nearest wall point. The radius of each cylinder is fixed as the minimum distance from the axis to the wall. If the radius of a cylinder is smaller than a size defined by the user (e.g., one voxel size), this cylinder is discarded.

5.5 Volume-Rendering Acceleration using the Cylindrical Structure

The cylindrical structure can be used to accelerate volume rendering using space leaping. In this section, we describe how this can be done and we discuss the use of the distance map for early ray-termination.

The cylindrical structure generated is used to calculate the ray segments which go through empty space. Therefore, for each ray the intersection points with the cylindrical structure are calculated. Besides, frustum culling is applied to avoid intersection tests with cylinders that are not visible from the current camera position.

If the camera is inside the cylinder, then the first intersection is taken as the first possible sampling point. Afterwards, the distance map is used to jump until the distance to the wall interval is smaller than the sample rate, and then the usual slow compositing is applied [Wan99b]. It may happen that the ray enters another cylinder again. In that case, the next intersection point in the list of intersections between the viewing ray and the cylinder approximation is used as a possible sampling point. If the camera is outside the cylinder, a usual distance jumping is applied until the ray enters a cylinder or is close to the interval wall. Figure 5.5 illustrates how the algorithm works for three sample rays.

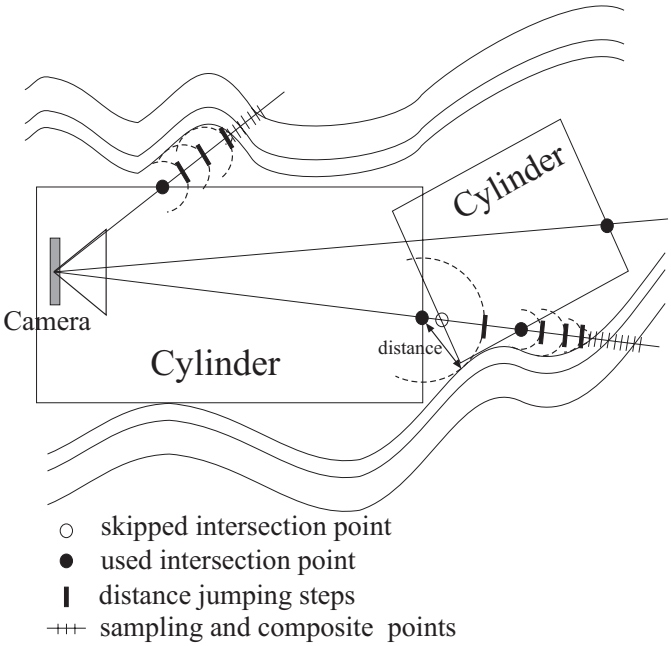


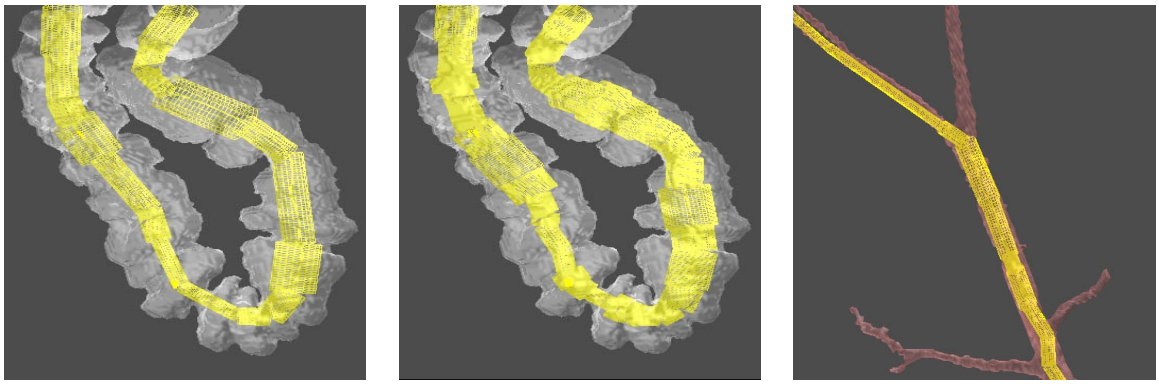
Figure 5.5: Illustration of the volume rendering algorithm using the cylindrical approximation structure.

The most common early ray-termination technique terminates the ray if the accumulated opacity approaches one. In our approach, another early ray-termination condition is added. Assuming that we want to sample points just within the wall interval of the organ, we do not want to go on sampling the rays in areas outside the interval wall. To determine when the ray is outside the wall interval, we calculate a distance map with the distance of the background voxels to the nearest object voxels. The algorithm consults the distance map to decide if the ray can be terminated at a given sample point. This early ray-termination approach cannot be used if the organ tissue is rendered transparently and the region outside of the organs shall be visualized.

5.6 Results

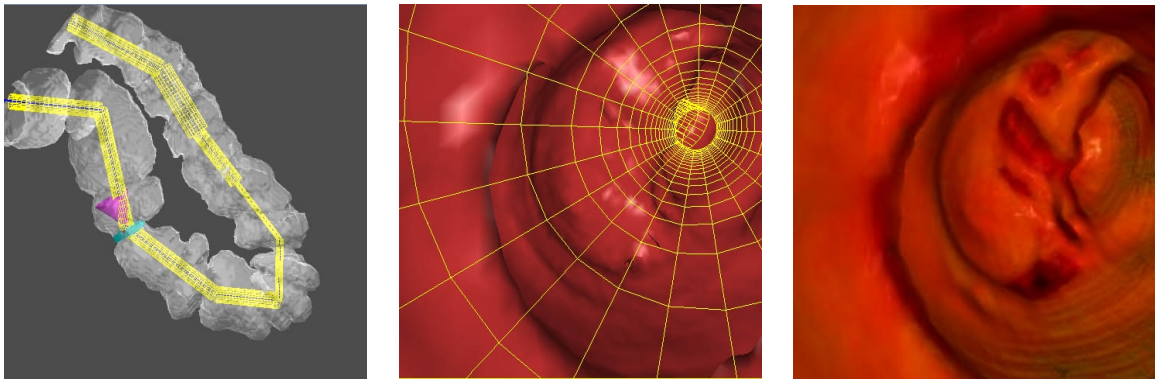
Figures 5.6a and 5.6b show the result of applying the cylinder-structure generation to a colon. Figure 5.6c shows the cylinder approximation of one central path in the aorta (the algorithm does not use branches). Figure 5.6a and 5.6b present the difference if the wall-interval tolerance-values are modified (Figure 5.6a has more inside tolerance of the wall interval than 5.6b). In figure 5.6b, the structure fits

better to the organ cavity but the number of cylinders generated also increases. Figure 5.7a illustrates the effect of decreasing the ε value for the same data as in figure 5.6a and 5.6b.



a) b) c)

Figure 5.6: Surface of the segmented organ together with the cylindrical approximation: a) CT colon data. b) The same CT colon data as in *a* but using a different wall thickness. c) CT kidney data set. The segmented organ is the aorta.

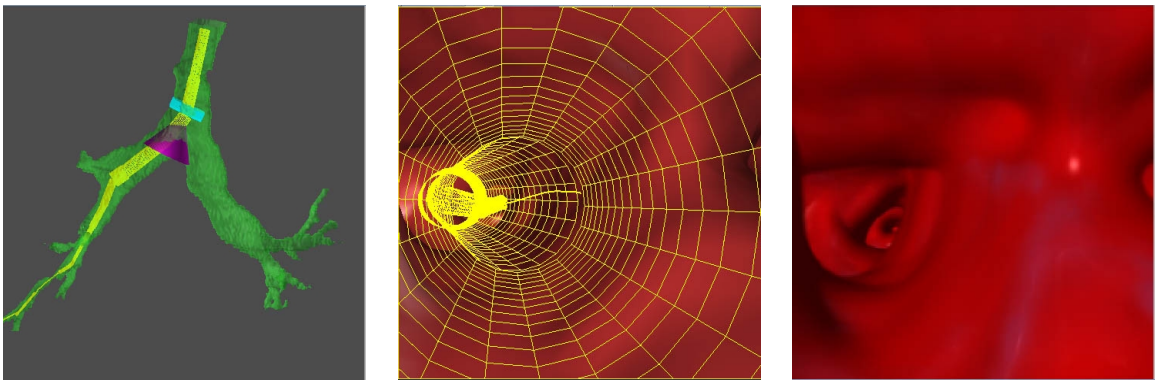


a) b) c)

Figure 5.7: Virtual endoscopy of a 200x115x300 CT data set of a colon : a) Outside view of the segmented object surface together with the camera and the cylindrical approximation. b) Endoscopic view of the cylindrical structure and a surface based rendering. The polygons have been generated using marching cubes. c) Endoscopic view using direct volume rendering and the cylindrical approximation. The accelerated rendering produces exactly the same result as a brute force approach.

In figure 5.7, a virtual endoscopy example of a 200x115x300 CT colon data set is shown. Figure 5.7a shows an outside view of the colon portion to be inspected and the camera position. The endoscopic view in 5.7b shows an extracted surface using marching cubes together with the cylinder structure which contains 9 cylinders. Figure 5.7c presents the result of applying direct volume rendering using the cylindrical approximation. The accelerated volume rendering produces exactly the same results as a brute

force approach. However, the rendering time of the 256x256 image using the cylindrical structure was four times faster than with the brute force approach.



a) b) c)

Figure 5.8: Virtual endoscopy of a 205x83x105 CT data set of a trachea: a) Outside view of the segmented object surface together with the camera and the cylindrical approximation. b) Endoscopic view of the cylindrical structure and a surface based rendering. The polygons have been generated using marching cubes. c) Endoscopic view using direct volume rendering and the cylindrical approximation.

In figure 5.8, a 205x83x105 CT dataset of a trachea is used. The rendering time was 3.5 times faster than the direct volume rendering using a brute force approach for a 256x256 image and using 14 cylinders.

The performance of the presented algorithm depends on several parameters, such as the transfer function, the organ shape, the number of cylinders generated and the characteristics of the cylinders. To obtain good results these parameters have to be tuned carefully.

If the algorithm presented by Wan [Wan99b] is extended to use the presented wall interval concept, and also to use the proposed early ray-termination, then in our implementation the performance is similar to the presented cylindrical approximation. The advantage of the cylindrical structure is that it has the potential of being used for other purposes, for example, being a sketch representation of the organ cavity.

The acceleration achieved by the presented method does not allow real-time direct volume rendering on a common PC (i.e., Pentium II 400MHz), but accelerates the generation of high quality precalculated movies.

One research direction to follow is the automatization of the parameter tuning. Other structures, like general cylinders, could be studied further to obtain a better approximation of the organ cavity, as well as new generation algorithms to obtain better approximations. The cylindrical structure might also be helpful for polygon assisted ray casting [Sobi95]. The cylindrical structure also has the potential of being used for image based rendering.

Chapter 6

Perspective Projection through Parallelly Projected Slabs

> If we would know what we are doing, it wouldn't be called research.
> *Albert Einstein (1879-1955)*

6.1 Introduction

This chapter presents a technique to accelerate perspective direct volume rendering for virtual endoscopy. Unlike chapter 5, we use hardware acceleration to achieve interactive frame rates for perspective direct volume rendering [Vila01b].

As described in chapter 2, 3D texture mapping [Cabr94, Cull93] is the most often used hardware acceleration technique. This method achieves interactive frame rates on an SGI Reality Engine, but it is difficult to incorporate this technique into a desktop machine like a PC. The basic method does not support the possibility to estimate gradients, which is required to employ lighting models like the Phong model with diffuse and specular lighting effects. Several approaches to overcome this problem have been proposed [West98].

The VolumePro board [Pfis99] is a hardware implementation of ray casting using shear-warp factorization [Lacr94]. It provides real time rendering with compositing, classification with density based transfer functions, and Phong shading.

One of the main drawbacks of this board, with regard to usage in virtual endoscopy, is that it does not produce perspective projection. For outside views, parallel projection gives good results but for inside views (e.g., endoscopic views), perspective projection is mandatory to provide a correct depth cue.

A method to approximate perspective projection from several parallel projected slabs, similar to slicing [Cull93] and to slab subdivision [Wan00] is presented in section 6.2. Section 6.3 describes the error estimation of the approximation. An improvement in the performance of the initial algorithm by using the error estimation is described in section 6.4. Some problems are investigated and improvements are

presented in section 6.5. Finally, a study using several clinical data sets and performance issues are discussed.

Actually, the presented algorithm is not only restricted to the VolumePro application. More generally, the concept can be applied wherever perspective projection is used.

6.2 Projected-Slabs Algorithm

The VolumePro system is able of producing high quality volume renderings of about 30 frames per second for a 256 cubic-size volume-data.

The VolumePro technology implements a shear-warp algorithm [Lacr94] in hardware. It renders the base-plane image and the 2D warp operation must be done by common graphics hardware.

The basic idea of the presented algorithm, called projected-slabs algorithm, is as follows: generate perspective rendering of the entire 3D data set, approximated by consecutive parallel projections of slabs of the volume data (see figure 6.1). A slab is part of the volume data in between two cutting planes which are orthogonal to the viewing direction.

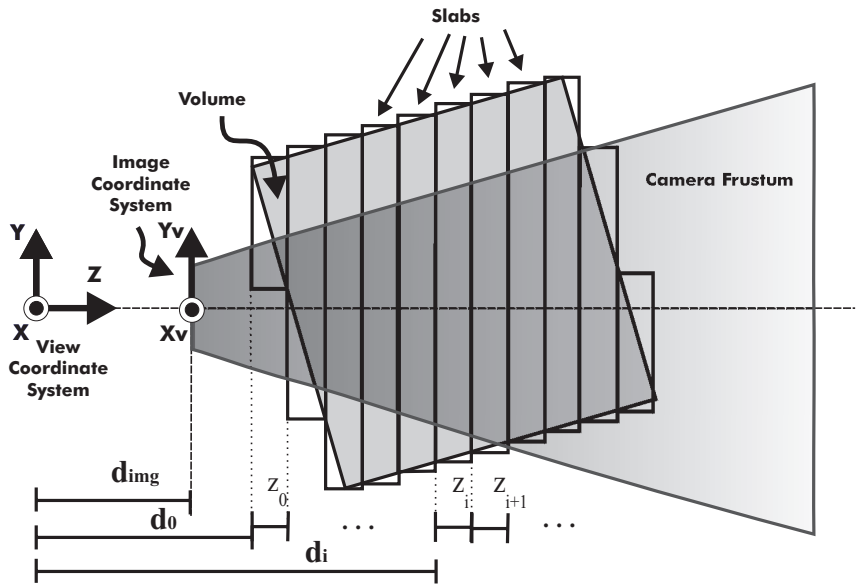


Figure 6.1: Illustration of the perspective approximation using the projected-slabs algorithm.

The thickness of a slab is selected such that the difference between a parallel projection and a perspective projection of the volume data contained within a slab is tolerable (i.e., below a certain error threshold). Using the cutting plane feature of the VolumePro system, each slab is rendered using parallel projection. The resulting base-plane image of an individual slab is then warped and transformed according to the perspective parameters of the defined camera.

All the images, one per slab, are finally blended to get the image of the entire data set. Figure 6.2 illustrates which data values are accumulated along a viewing ray with the projected-slabs method as compared to the correct perspective solution.

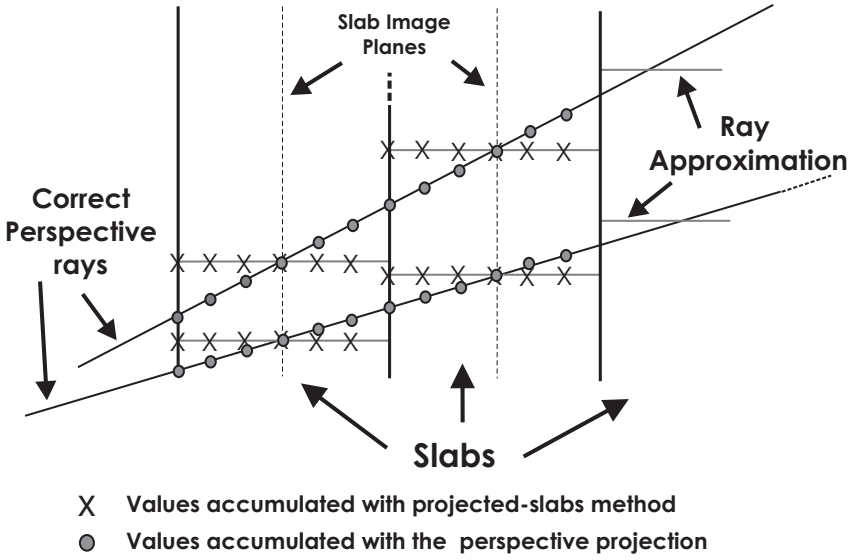


Figure 6.2: Accumulated values in a correct perspective projection as compared to the projected-slabs algorithm.

The algorithm uses an entire VolumePro rendering cycle for each rendered slab, and therefore the rendering frame rate is decreasing in the order of the number of slabs that are needed for the perspective approximation. Furthermore, there is an overhead due to the blending of the slabs.

Given the viewing position and the viewing direction, the slabs can be numbered using the distance to the viewpoint in the following way (see figure 6.1):

$$d_j = \quad d_0 + \sum_{i=0}^{j-1} \Delta z_i \tag{6.1}$$

where $j \geq 1$, d_0 is the distance from the viewpoint to the front plane of the first slab, and Δz_i is the thickness of the slab which starts at distance d_i.

If Δz_i is a constant value smaller than a voxel size for all slabs, it is intuitive to see that the result produces good quality perspective rendering. On the other hand, it also produces an intolerable high number of slabs. So the thickness of the slabs must be set to a value larger than one voxel size to get reasonable performance.

Since we are approximating perspective projection, it is important to be able to evaluate the error produced due to this approximation.

6.3 Error Estimation of the Projected-Slabs Algorithm

In this section, we study the error that results from the use of parallelly projected slabs to approximate the correct perspective view.

The basic error is that the sample points are projected to the wrong position in the image plane, and therefore they are accumulated into the wrong pixel. The error estimation is defined as the distance in the image plane between the correct perspectively projected point and the point produced by the projected-slabs algorithm.

In the rendering pipeline, the difference between parallel and perspective projection appears after the world coordinates have been transformed to view coordinates. To transform from view coordinates to image plane coordinates, the appropriate projection matrix is used. For simplicity and more intuitive explanation, we assume a left handed camera system (see figure 6.1) where the viewpoint is in the origin of the view coordinates. The image plane is orthogonal to the Z-direction and located at a distance d_{img} from the viewpoint.

We define a point $P_v = (X_v, Y_v, Z_v)$ as a point resulting from applying a view-coordinate transformation to an arbitrary point in world-coordinates.

The perspective projection matrix for a left handed camera system, supposing left accumulation matrix notation is:

$$M_{persp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where d is the distance from the viewpoint to an arbitrary projection plane. If $d = d_{img}$ then the projection plane is the image plane. The parallel transformation to image coordinates is simply the transformation of the Z-coordinate to the projection plane position d. It can be expressed by the matrix:

$$M_{paral} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d & 1 \end{bmatrix}$$

If these transformations are applied to P_v we obtain the following equalities (P^h are points expressed in homogeneous coordinates).

$$\begin{aligned} P^h_{persp} &= P^h_v * M_{persp} = \left[X_v, Y_v, Z_v, \frac{Z_v}{d} \right] \\ P^h_{paral} &= P^h_v * M_{paral} = [X_v, Y_v, d, 1] \end{aligned}$$

The points P^h_{persp} and P^h_{paral} are transformed to the 2D projection-plane coordinate-system. This coordinate system is defined with the same X and Y-direction as the view coordinate system. The projection plane's origin corresponds to $[0, 0, d]$ in view-coordinates. Then P^h_{persp} corresponds to P_{persp} and P^h_{paral} corresponds to P_{paral}, with

$$\begin{aligned} P_{persp} &= \frac{d}{Z_v} [X_v, Y_v] \\ P_{paral} &= [X_v, Y_v] \end{aligned}$$

Error e_p is defined as the distance between the perspective, P_{persp}, and parallel, P_{paral}, projection of a point P_v:

$$\begin{aligned} e_p &= \|P_{persp} - P_{paral}\| = \\ &= \left\| \left(\frac{d}{Z_v} - 1 \right) [X_v, Y_v] \right\| \end{aligned}$$

Z_v can be expressed by $Z_v = d + \Delta z_v$. Then we derive

$$\begin{aligned} e_p &= \left\| \left(\frac{d}{d + \Delta z_v} - 1 \right) [X_v, Y_v] \right\| = \\ &= \left| \left(\frac{\Delta z_v}{d + \Delta z_v} \right) \right| \| [X_v, Y_v] \| \end{aligned} \tag{6.2}$$

where $-\infty \leq \Delta z_v \leq \infty$.

Equation 6.2 represents the distance between a parallel projection and perspective projection of a point on a projection plane situated at a distance d from the viewpoint.

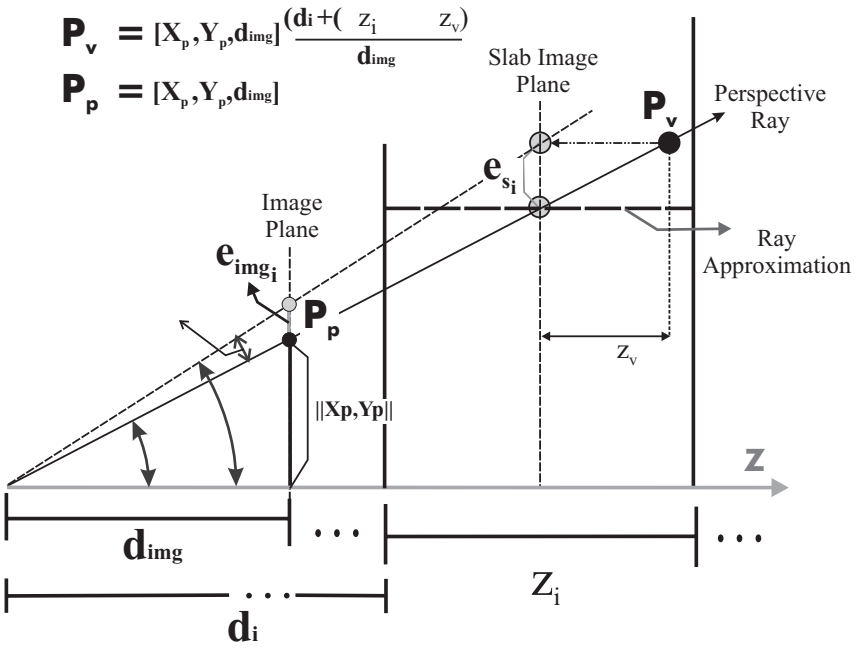


Figure 6.3: Illustration of the error estimation.

In the projected-slabs algorithm, for each slab i the points within the slab are parallelly projected to the slab image plane. The slab image plane is situated in the middle of the slab, at distance $d_i + \frac{\Delta z_i}{2}$ from the viewpoint (see figure 6.3).

We define e_{s_i} as the distance between the parallel projection and the perspective projection of a point on the slab image plane. Using equation 6.2 and $d = d_i + \frac{\Delta z_i}{2}$, it follows that

$$e_{s_i} = \left| \left(\frac{\Delta z_v}{d_i + \frac{\Delta z_i}{2} + \Delta z_v} \right) \right| \| [X_v, Y_v] \| \tag{6.3}$$

where $-\frac{\Delta z_i}{2} \leq \Delta z_v < \frac{\Delta z_i}{2}$.

Equation 6.3 gives a distance in the slab image plane. However, we are interested in this distance projected into the image plane. Therefore, we project e_{s_i} to the image plane and get e_{img_i}:

$$e_{img_i} = \left|\left(\frac{\Delta z_v}{d_i+\frac{\Delta z_i}{2}+\Delta z_v}\right)\right|\left(\frac{d_{img}}{d_i+\frac{\Delta z_i}{2}}\right)\|[X_v, Y_v]\| \tag{6.4}$$

where d_{img} represents the distance between the image plane and the viewpoint.

The point P_v can be expressed as follows

$$P_v = \frac{Z_v}{d_{img}} * [X_p, Y_p, d_{img}] \tag{6.5}$$

where $P_p = [X_p, Y_p, d_{img}]$ is the perspective projection of the point P_v on the image plane.

Combining equation 6.4 with 6.5, it follows that

$$e_{img_i} = \left|\left(\frac{\Delta z_v}{d_i+\frac{\Delta z_i}{2}+\Delta z_v}\right)\right|\left(\frac{d_{img}}{d_i+\frac{\Delta z_i}{2}}\right) \left|\left(\frac{d_i+\frac{\Delta z_i}{2}+\Delta z_v}{d_{img}}\right)\right|\|\left[X_p, Y_p\right]\|$$

where $-\frac{\Delta z_i}{2} \leq \Delta z_v < \frac{\Delta z_i}{2}$.

Simplifying the previous equation, we conclude

$$e_{img_i} = \left(\frac{\Delta z_v}{d_i+\frac{\Delta z_i}{2}}\right)\|\left[X_p, Y_p\right]\| \tag{6.6}$$

where $0 \leq \Delta z_v \leq \frac{\Delta z_i}{2}$.

From equation 6.6, we can deduce several properties:

1. The error e_{img_i} is proportional to $\|\left[X_p, Y_p\right]\|$, the distance of P_p to the image plane origin.
2. $\forall[X_p, Y_p]$: $\Delta z_v \to 0 \Rightarrow e_{img_i} \to 0$. This behavior has already been intuitively described in section 6.2. Furthermore, it means that for points on the slab image plane the error is 0.
3. The error value increases, if Δz_v increases.
4. For fixed $\|\left[X_p, Y_p\right]\|$, if d_i increases, then the variation in e_{img_i} due to the changes in Δz_v decreases.

The value of e_{img_i} represents the error or distance in the image plane between the projected-slabs algorithm and the correct perspective projection of a point P_v situated in the slab i. The point is situated at a distance Δz_v from the slab image plane and its perspective projection to the image plane gives $[X_p, Y_p]$.

We are interested in finding the maximum error produced in the image plane due to the approximation of the projected-slabs algorithm. This can be achieved by finding the upper bound for all values of e_{img_i}.

Due to property 3, it is clear that e_{img_i} is maximal when $\Delta z_v = \frac{\Delta z_i}{2}$, i.e., when the point P_v is situated on the front or back plane of the slab.

Due to property 1, it is clear that the upper bound value of e_{img_i} within slab i occurs when the highest value of $\| [X_p, Y_p] \|$ is reached. That implies that $[X_p, Y_p]$ can be fixed to the furthest point from the origin of the image plane that contributes to the final image. This value corresponds to the corners of the image quadrilateral defined by the intersection of the frustum and the image plane (i.e.,$[X_c, Y_c]$). Therefore we define e_i^{max} as the maximum error in the final image inferred by slab i.

$$e_i^{max} = \left(\frac{\frac{\Delta z_i}{2}}{d_i + \frac{\Delta z_i}{2}} \right) \| [X_c, Y_c] \| \tag{6.7}$$

The maximal error in the final image is the maximum value of e_i^{max} for any slab i.

$$e^{max} = max\{e_i^{max} | i \geq 0\} \tag{6.8}$$

Due to property 4 we see that if for all i, Δz_i is a constant, then every slab has a different e_i^{max} and its value decreases when d_i increases. So, the maximal error produced in the final blended image corresponds to the maximal error of the first slab, $e^{max} = e_0^{max}$.

In the next section, we use the previous observations to optimize the projected-slabs algorithm, without increasing the maximal error value e^{max} produced in the final image.

6.4 Error-Induced Variation of Slab Thickness

It has been observed that the maximal error for slab i due to the projected-slabs algorithm, e_i^{max}, depends on the distance of the slab to the viewpoint and on the slab thickness. From property 4 and equation 6.8, it can be deduced that slabs further away from the viewpoint may have a greater thickness than slabs closer to the viewpoint keeping the same maximal image error e^{max}.

In this section, we present the criterion for selecting the slab thickness dependent on the distance to the viewpoint, the camera characteristics and the maximal error. The rule is to have as few slabs as possible while keeping the error tolerance unchanged.

Using equation 6.7, Δz_i can be isolated in the following way:

$$\Delta z_i = 2 * \left(\frac{e_i^{max} * d_i}{\| [X_c, Y_c] \| - e_i^{max}} \right) \tag{6.9}$$

e_i^{max} is set to a constant value, $DistanceError$, for all the slabs. We define the incremental slab thickness algorithm using equations 6.9 and 6.1. The thickness of the slabs is defined in an iterative way, assuring that the error will be kept smaller than the defined value $DistanceError$. As mentioned in section 6.2, the projected-slabs algorithm decreases the frame rate if the number of slabs increases. Calculating the thickness using equation 6.9, the slab thickness will increase with the value of d and so, less slabs are needed and therefore the frame rate is higher (see figure 6.4).

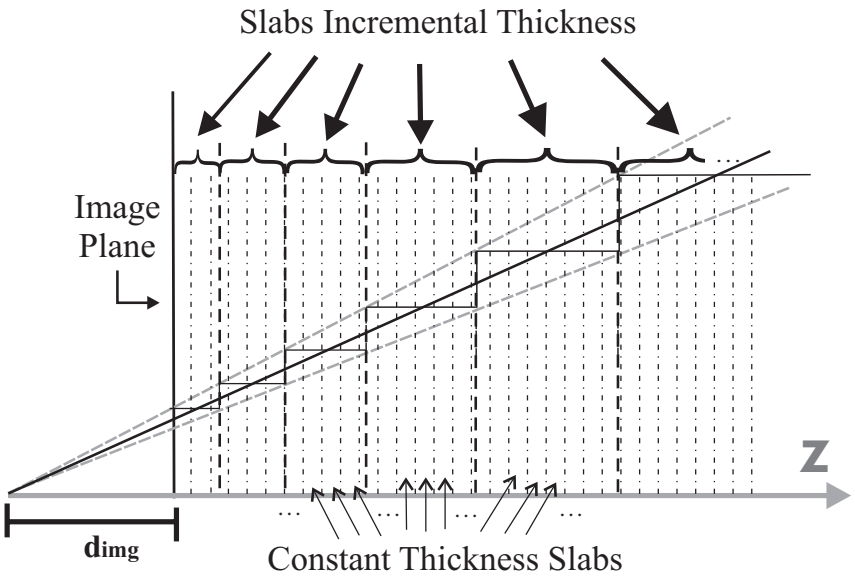


Figure 6.4: Comparison of the slabs using constant thickness and the incremental slab thickness calculation.

Apart from the $DistanceError$, equation 6.9 also depends on the camera characteristics. $[X_c, Y_c]$ is specified by the intersection of the frustum and the image plane. Δz_i just needs to be recomputed when the camera characteristics, the $DistanceError$ or the first slab distance d_0 are modified.

The presented result, obtained using mathematical derivation, illustrates the fact that in perspective projection objects far away are projected smaller to the image plane. This intuitive idea has been already used in other adaptive approaches [Brad98, Kree98, Wan00].

Using simple trigonometry and figure 6.3, Δz_i can be intuitively described as a function of the angles α and β:

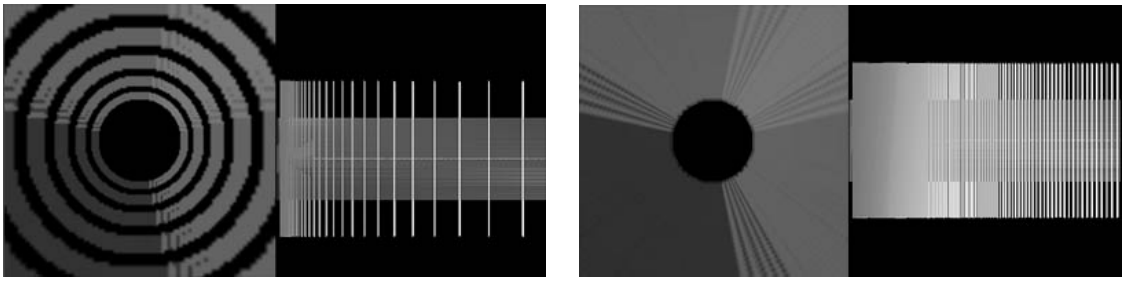
$$\tan\beta * \left(d_i + \frac{\Delta z_i}{2}\right) = \tan\alpha * \left(d_i + \frac{\Delta z_i}{2} + \Delta z_v\right)$$

where $-\frac{\Delta z_i}{2} \leq \Delta z_v < \frac{\Delta z_i}{2}$ and $\beta = \alpha + \Delta\alpha$. $\Delta\alpha$ corresponds to the angle defined between the ray where a point is accumulated using correct perspective projection and the ray that accumulates the point in the projected-slabs method. We can observe that the maximum value of $\Delta\alpha$ within a slab is reached when $\Delta z_v = \pm\frac{\Delta z_i}{2}$. For simplicity, we assume $\Delta z_v = -\frac{\Delta z_i}{2}$. It follows that

$$\Delta z_i = 2 * \left(\frac{\tan\alpha * d_i}{\tan\beta} - d_i\right) \tag{6.10}$$

Equation 6.10 gives the option to express the error tolerance as an angle, $AngleError$. It can be seen that α must be set to the maximum frustum angle of the camera to get the maximal value of $\Delta\alpha$. With some simple transformations, it can be proven that equation 6.9 and 6.10 are equivalent.

$DistanceError$ or $AngleError$ are parameters of the algorithm. The estimation of the error tolerance (i.e., $DistanceError$ or $AngleError$) depends on the volume data to visualize.



a) b)

Figure 6.5: An illustration of the error tolerance behavior. Two endoscopic views using the projected-slabs algorithm of a voxelized tube are shown together with the corresponding outside view with the slab image planes. The error tolerance is different for each endoscopic view: *a*) $DistanceError = 5\%$ of the image size, *b*) $DistanceError = 2\%$ of the image size.

To illustrate this, we study one of the worst cases for the projected-slabs algorithm. This case occurs when the camera is in the center of a straight tube and the camera is pointing in the direction of the tube axis.

In parallel projection, a ring with the tube thickness would be projected to the image plane. If the projected-slabs algorithm is used, for any slab a ring with the tube thickness will be projected. If the tube has a small thickness (see figure 6.5 a), it looks like several concentric rings, giving the impression of having different objects. This is because the specified error tolerance is larger than the projected thickness of the tube. Therefore, we see the tube as a discontinuous surface. If the error is decreased to a value that approximates the projection of the thickness in the image plane, a better result is obtained (see figure 6.5 b).

6.5 Performance Improvements

In this section, we describe several implementation strategies to improve the performance of the algorithm for virtual endoscopy.

Using the projected-slabs algorithm, the possibility to use perspective volume rendering with the VolumePro board is achieved. However, also several problems arise.

One of the problems is that the frame rate does not allow interactive use (i.e., less than 1 f.p.s.) when the number of slabs needed to cover the volume is bigger than 20 or 30. We implemented a progressive rendering algorithm. In this algorithm, the image shown to the user is updated after rendering each slab and not just after all the slabs have been processed. In this way the user can see the progressive blending of the slabs and get immediate feedback of the visualization.

There are two ways of compositing in volume rendering: front-to-back and back-to-front. These compositing techniques differ in the processing order of the slabs and the accumulation function applied. In OpenGL, back-to-front compositing is easy to implement, but it implies that the first slabs to be blended are the ones that are farthest from the viewpoint. Usually in virtual endoscopy, the region to explore is situated near the camera. In most data visualizations the slabs further from the viewpoint do not add

much information to the final image. This is because many rays have accumulated completely opaque values before reaching these slabs. Therefore front-to-back compositing has been implemented. The front-to-back compositing order using OpenGL implies a slow down of the algorithm, since each slab base-plane texture must be multiplied by its own opacity to be able to obtain the correct front-to-back compositing. It produces a reduction of speed of about 30%. We believe this loss is worthwhile since in the incremental front-to-back rendering the interesting regions are visualized first and therefore the user receives faster feedback of his modifications of the rendering parameters.

Another problem to deal with is the aliasing artifacts due to the undersampling of the base-plane image. One option to overcome this problem is to use supersampling. The VolumePro board provides several levels of supersampling. The disadvantage is that supersampling implies a corresponding slow down in rendering performance. The supersampling can be implemented in such a way that the slabs closer to the viewpoint are rendered with a higher supersampling level.

One of the drawbacks of the VolumePro is that only directional light sources can be defined. In virtual endoscopy it is quite convenient to use a head-mounted point light source. In the implementation we use a directional light pointing in the same direction as the camera and two additional directional lights rotated several degrees from the camera direction.

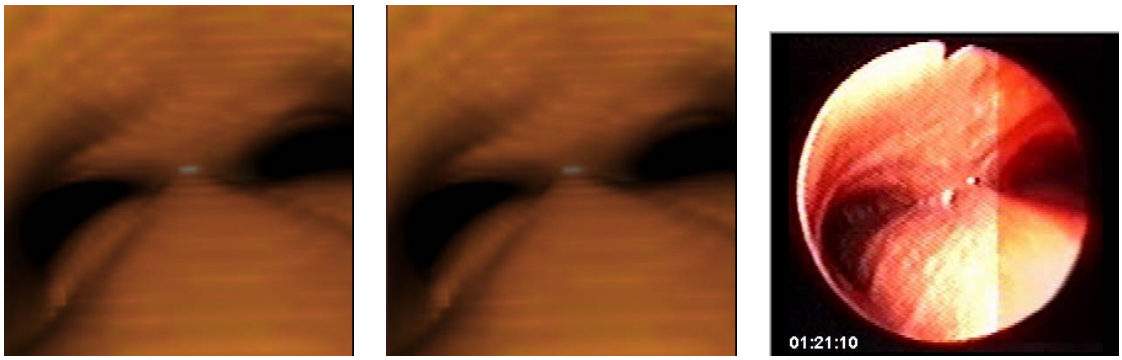
We observed that for wide tubular structures, the slabs close to the viewpoint do not contribute to the final image, since the frustum just intersects empty space. This depends, of course, on the data to be visualized. To attenuate this effect, front clipping has been implemented. The user defines the distance from the viewpoint to the first slab.

6.6 Results

We present some timings and the evaluation realized for three types of data sets. The calculation times were obtained by executing the algorithm on a Pentium II with 400MHz. The performance of the algorithm presented in this chapter basically depends on the time necessary for the VolumePro board to render a slab, and on the time for warping and blending. We provide timings relative to the times that can be achieved on a rendering cycle. A rendering cycle consists of the rendering of one slab using the VolumPro board and the warping of the resulting base-plane image. The speed and quality of the warping and blending steps depend on the graphics hardware used.

In tables 6.1, 6.2 and 6.3, the first column shows whether a constant or an incremental slab thickness algorithm was used, the error column denotes the maximal error expressed as a percentage of the image size, the third column gives the number of slabs used, the f.p.s. column represents the frame rate and the last column denotes the slow down factor compared with the time of one rendering cycle of the VolumePro.

The first data set is a CT scan of a trachea of size 292x136x114. The results of this visualization with different maximal errors are shown in figure 6.6. The CT was acquired from a corpse. After the scanning, a real bronchoscopy was performed. Figure 6.6 compares the results of the projected-slabs approach with the real endoscopic view from a similar camera position. It can be observed that the difference between incremental slab thickness (figure 6.6a) and the constant slab thickness (figure 6.6b) can be neglected. Numerical results are given in table 6.1.

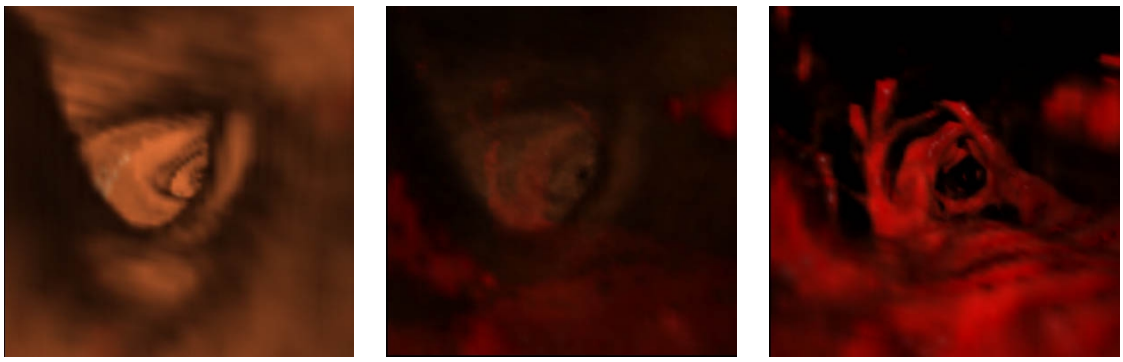


a) b) c)

Figure 6.6: Visualization of the CT trachea data set of a corpse compared to a real endoscopic view: **a)** projected-slabs algorithm with incremental slab thickness (37 slabs), **b)** projected-slabs algorithm with constant slab thickness equal to one voxel distance (162 slabs), **c)** real bronchoscopy snapshot.

slab thickness	error	# slabs	f.p.s.	slow down factor
constant	0%	162	0.093	173.40
incremental	2.5%	37	0.4	40.32

Table 6.1: Times for the CT trachea of the corpse, in back-to-front renderings. One rendering cycle takes 65 ms. The error in the second line is 0% since a constant slab thickness of one voxel size per slab has been defined.



a) b) c)

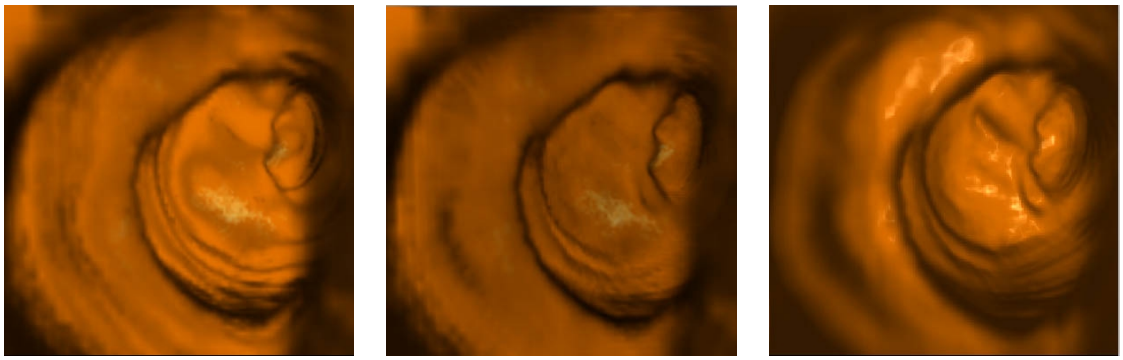
Figure 6.7: CT trachea data set rendered with different transfer functions. From left to right from opaque to transparent trachea walls for the same camera position.

The second data set is a CT volume data of a trachea with a resolution of 205x83x105 (see figure 6.7). Contrast medium was injected into the vessels. There is a common surgical procedure (i.e., transbronchial biopsy) where the endoscopist does a biopsy of a tumor near the outside walls of the trachea. Using an endoscope, the surgeon penetrates the walls of the trachea from inside at the appropriate position. It is important for the physician to be able to localize the pulmonary artery and the aorta together

with other vessels near the trachea walls. With our system, the endoscopist can navigate until the region of interest is reached. Thereafter, the vessels outside the trachea are inspected by rendering the trachea surface transparently. Figure 6.7 contains the images produced with different opacities for the walls of the trachea. The times are presented in table 6.2. It can be observed that it is necessary to change the opacity of the trachea walls interactively, since it is difficult to recognize the structure of the trachea when it is semitransparent.

slab thickness	error	# slabs	f.p.s.	slow down factor
incremental	4%	21	0.94	22.6

Table 6.2: CT of a trachea using transparency, in front-to-back rendering. One rendering cycle takes 47 ms (21.27 f.p.s.)



a) b) c)

Figure 6.8: Spiral CT colon data set visualization: **a)** projected-slabs algorithm with incremental slab thickness (28 slabs), **b)** projected-slabs algorithm with constant slab thickness and with the maximal error equal to **a** (179 slabs), **c)** brute force ray casting algorithm.

The third data set of size 198x115x100 is a portion of a spiral CT of a colon. In figure 6.8, a comparison between the projected-slabs technique and brute force volume rendering is presented. For the projected-slabs technique, results are presented for both a constant and an incremental slab thickness. Comparing constant slab thickness with incremental slab thickness, the number of slabs is smaller but the maximum error is the same. The times and frame rates are given in table 6.3.

slab thickness	error	# slabs	f.p.s.	slow down factor
constant	2.5%	179	0.13	163.61
incremental	2.5%	28	0.8	26.58

Table 6.3: Spiral CT colon data set: times for a complete back-to-front rendering with constant and incremental slab thickness. One rendering cycle in the VolumePro takes 47 ms (21.27 f.p.s.).

We have observed that structures which are wide cavities produce good results since the firstly projected

voxels cover a smaller image plane area and aliasing less affects the performance of the algorithm. We have also experienced that around 30 slabs are sufficient for most of the visualizations.

As has been mentioned in section 6.4, the projected-slabs algorithm is an approximation which depends on the characteristics of the structure to visualize. This implies that, depending of the structure and the visualization parameters, some artifacts appear. For instance the transition between planes can be observed in some cases where the visualization parameters are not adequate.

For some animations showing more results of the algorithm see
http://www.cg.tuwien.ac.at/research/vis/vismed/projected-slabs/animation.html.

The presented algorithm achieves reasonable frame rates for interaction. However it does not achieve real-time frame rates. As a further improvement, the algorithm could be sped up using active subvolumes. In the projected-slabs algorithm, the entire volume is rendered for every slab but just a small portion of it contributes to the final image. VolumePro allows the definition of an active subvolume with a size equal or less to the original volume size. Instead of the entire volume, the active subvolume is then rendered. The smaller the active subvolume is, the faster the rendering with the VolumePro is. The active subvolume can be defined by the cells of a regular grid that are within the camera frustum and contribute to the rendered slab.

An investigation of how to automate the estimation of the parameters of the algorithm (i.e., front clipping distance and the error tolerance) could be done. There are numerous parameters that affect the quality of the results. They are difficult to specify completely manually.

A study of other uses of the presented error estimation algorithm should be performed. More generally, the algorithm subdivides view space into slabs, within the slabs approximative but faster operations are possible. In our case, parallel projection is used to approximate the perspective view. The concept might also be applied to other algorithms where perspective projection is used (e.g., splatting and shear-warp rendering).

Part II

Virtual Colon Unfolding

Chapter 7

Introduction to Virtual Colon Unfolding

> Keep on the lookout for novel ideas that others have used successfully. Your idea has to be original only in its adaptation to the problem you're working on.
>
> *Thomas Edison (1847-1931)*

Most of the virtual endoscopy techniques presented in the last years concentrate on simulating the view of a real endoscope. This is the view that endoscopists are used to. It can be useful for certain applications, like in an intraoperative scenario, but it is not necessarily the best way to inspect the inner surface of an organ. Actually, a real endoscope and organ are subject to physical limitations that a virtual endoscope and organ do not have. We concentrate on virtual colonoscopy, which focuses on the examination of the colon. Physicians are mainly interested in visualizing the inner surface of the colon which is where polyps can be detected with endoscopy. It is important that the physician can estimate the size of polyps, since large polyps are more likely to develop into malignities. The usual endoscopic view visualizes just a small part of the surface. Furthermore, it is difficult to detect polyps that are situated behind the folds of the colon. An efficient way to inspect the inner surface would be to open and unfold the colon and then examine its internal surface. Unfortunately, this cannot be done in reality, if we want that the patient survives. On the other hand, there is no patient damage if this dissection of the organ can be achieved virtually with the medical data obtained by CT or MRI (i.e., the virtual organ).

Some authors propose a technique to straighten and unravel an organ virtually [Sora01, Wang95]. Their approach starts with defining a path which is placed as close to the center of the object as possible. Then, a sequence of frames is calculated. For each frame, a cross-section orthogonal to the path tangent is calculated. Then the central path is straightened and the cross-sections are piled to form a stack. As a last step, the straightened colon is unfolded obtaining a volume model of the unfolded colon. The model is displayed afterwards using standard volume rendering techniques. This method allows to visualize the complete surface at once. However, one of the main problems of this technique appears in high curvature areas of the central path, i.e., at path locations where the radius of curvature is bigger than the organ diameter. In such cases, orthogonal cross-sections intersect each other or are far apart in some other regions (see figure 7.1). As a consequence, a polyp can appear twice in the unfolded model or it can be missed completely.

Wang et al., in later work, [Wang99, Wang98] try to overcome this problem. The authors use electrical field lines generated by a locally charged path to govern curved cross-sections instead of the planar

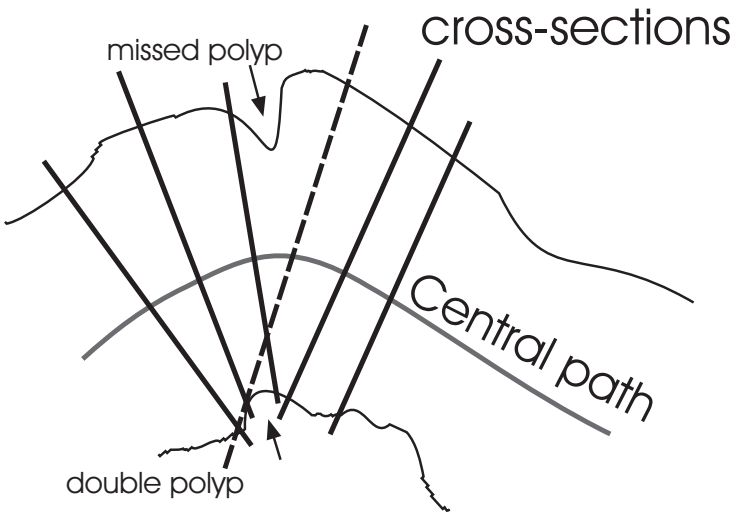


Figure 7.1: Illustration of the possible undersampling and double appearance of polyps due to intersections of the cross-sections in high curvature areas. The dashed cross-section line produces a double appearance of the polyp.

sections. The cross-sections tend to diverge avoiding conflicts. If the complete path is charged then the curved planes will not intersect. However, for each point of the field lines the contribution of each charge in the path must be calculated. This operation is computationally so expensive that the authors propose to just locally charge the path. A small segment of the path contains the charges for each cross-section. In this way, the method is feasible in practice, but it cannot ensure that the curved cross-sections will not intersect each other anymore.

Flattening a polygonal representation of a colon involves tasks that have already been used in texture-mapping techniques. A major step thereby is to come up with a suitable surface parameterization. For texture mapping, this parameterization is used to assign texture values to surface points. For surface flattening the parameterization allows to display surface values (e.g., color) in the 2D parameter space [Same86]. A huge amount of techniques are dealing with texture distortions which in many cases cannot be avoided entirely. The distortions depend on the chosen surface parameterization (e.g., length and area preserving [Benn91], or angle preserving [Hake00b] or a combination of both [Floa97, Lévy98]).

Haker et al. [Hake00a] use conformal (i.e., angle preserving) texture mapping to map the polygonal colon surface, colored according to its mean curvature, to a plane. One of the main problems of this method, or any other which directly uses texture-mapping techniques, is that a highly accurate segmentation is necessary to ensure good results for diagnosis. The entire polygonal surface is flat and the result is a triangulated plane where the polyps have also been flattened. The color-coded mean curvature of the extracted surface is the only information which helps the physicians in identifying polyps. Furthermore, the surface needs to be smoothed to achieve a good mean-curvature calculation.

Paik et al. [Paik00] propose other kinds of camera projections for virtual endoscopy. With a normal endoscopic view just 8% of the solid angle of the camera is seen in each frame. Paik et al. project the whole solid angle of the camera by map projection techniques used to map the world globe in charts.

They suggest the use of the Mercator projection to map the solid angle to the final image. This technique samples the solid angle of the camera, then the solid angle is mapped onto a cylinder which is mapped finally to the image. In this method, the physician does not have a complete view of the colon and has to inspect a complete video.

All of these methods introduce some kind of deformation since it is mathematically impossible to perform a mapping between two surfaces preserving length, angles and area at the same time if the two surfaces do not have the same gaussian curvature.

In the next two chapters we present two new methods which virtually unfold the colon and concentrate on avoiding the problems presented by previous methods, like double appearance of polyps or undersampled areas.

Chapter 8

Local Colon Unfolding

> The proof of the pudding is in the eating. By a small sample we may judge of the whole piece.
>
> *Miguel de Cervantes Saavedra, "Don Quijote" (1547-1616)*

8.1 Introduction

In this chapter, we propose a new method to unfold the colon using a new camera projection technique. This method [Vila01c] generates a video where each frame is a local unfolding of the organ.

In section 8.2, the local unfolding method is described. Section 8.3 presents different sampling options that cause different deformation problems. A minimization of the rotation for the camera movement is described in section 8.4. Section 8.5 describes a non-photorealistic technique that enhances the perception of the image. Then it is presented how an approximate but fast endoscopic view can be generated with the data calculated for the projection method. Finally, results and studies with real colon data are presented.

8.2 Method Overview

The method proposed by Wang et al. [Wang99] generates an unfolded model of the colon that later on will be carefully inspected by the physician. Our method will not generate an unfolded model of the whole colon, but allows to inspect locally unfolded regions such that double appearance of polyps does not occur.

The presented method involves moving a camera along the central path of the colon. The path is smoothed using the techniques described in chapter 5 and finally approximated by a B-spline curve. For each camera position a small cylinder tangent to the path is defined. The point in the middle of the cylinder axis corresponds to the camera position. Rays starting at the cylinder axis and being orthogonal to the cylinder surface are traced (see figure 8.1). For each ray, direct volume rendering is used to calculate the color which corresponds to the cylinder point where the ray was projected. Finally, the colored

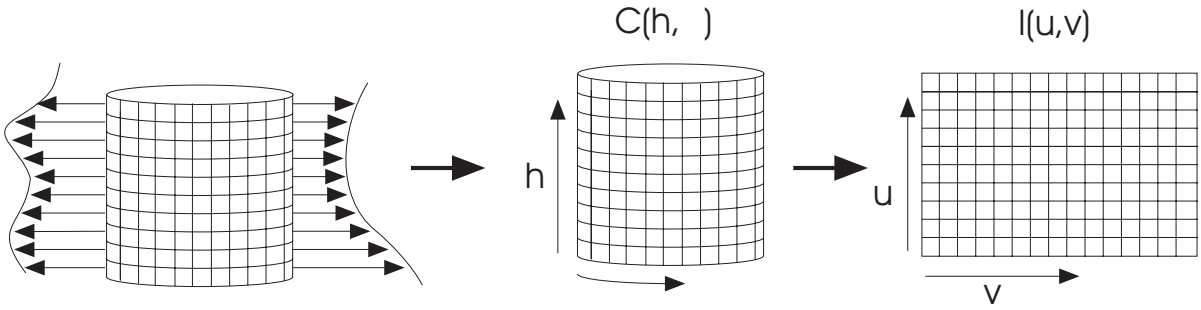


Figure 8.1: Illustration of the projection procedure. A region of the surface is projected onto the cylinder $C(h, \alpha)$. Then, the cylinder is mapped to the image $I(u, v)$

cylinder with the sampled rays is mapped to a 2D image. This is done by simply unfolding the cylinder. The cylinder axis must be short enough, so that the cylinder does not penetrate the surface of the colon. This can be done by taking into account the distance of the path to the organ surface.

The result is a video where each frame shows the projection of a small part of the inner surface of the organ onto the cylinder.

If the camera is moved slowly enough the coherence between frames will be high and the observer will be able to follow the movement of the surface.

In high curvature areas the intersection of cross-sections appears, too (see figure 7.1). However, possible double sampling of polyps emerges just between frames. Therefore, it does not cause a double counting of polyps since the human brain is able of tracking the polyp movement due to the coherence between frames. Moving along the central path in such a high curvature area, a polyp might move up and down (due to double sampling), but it is clearly identified as a single object.

8.3 Projection onto a Cylinder

The proposed projection is illustrated in figure 8.1. A cylinder $C(h, \alpha)$ is defined for each camera position. This cylinder is colored by tracing rays orthogonal to the cylinder surface (i.e., projecting a region of the organ surface onto the cylinder). Then the cylinder is mapped to the final image $I(u, v)$ by a simple mapping function $f : (h, \alpha) \rightarrow (u, v)$.

The sampling distance (i.e., the distance between two consecutive rays) in the h-direction is constant, and it must be at most half of the size of a voxel (see figure 8.1). In this way, correct sampling (Nyquist limit) in the h-direction is possible.

For each h-value, the rays are traced in radial directions with respect to the cylinder axis. The rays are diverging from each other, so the volume data is not sampled uniformly if the angle between rays is constant (see figure 8.2a). In the next section, two methods are described which project the organ surface onto the cylinder depending on the sampling of angle α, i.e., constant angle sampling and perimeter sampling.

8.3.1 Constant Angle Sampling

Constant angle sampling means that the angle between consecutive rays in the α direction is constant for rays with the same h-value. Figure 8.2a illustrates how this sampling is done. Using this method, the cylinder is sampled uniformly but the surface itself is not uniformly sampled.

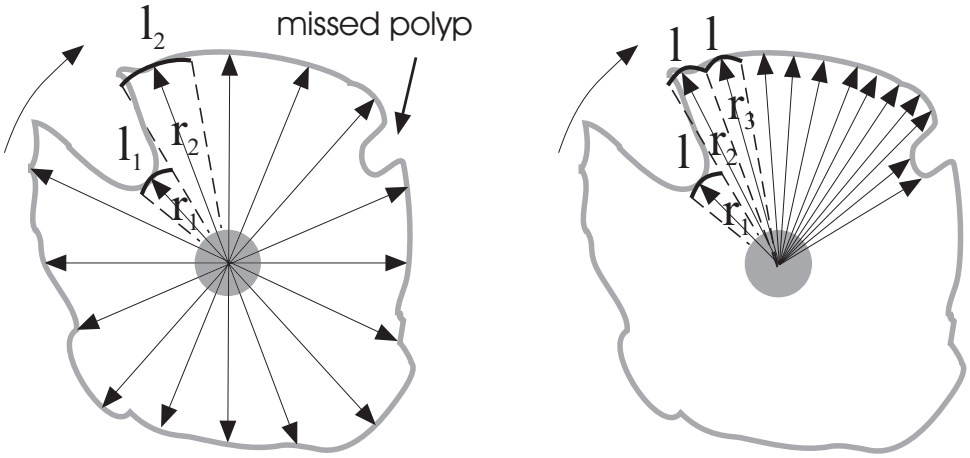


Figure 8.2: **a)** Constant angle sampling: it is shown that different surface lengths are represented by the same length in the cylinder. **b)** Perimeter sampling: same length but different angle.

The advantage of this method is that the relationship between both directions is locally preserved. Therefore, the angles are locally preserved too. An image generated by this method can be seen in figure 8.3a.

On the other hand, the area of the projected region is not preserved (see figure 8.2a). Therefore, the size of a projected polyp depends on the distance of the cylinder axis to the organ surface cavity. Consequently, the physicians cannot trust the sizes of the projected polyps.

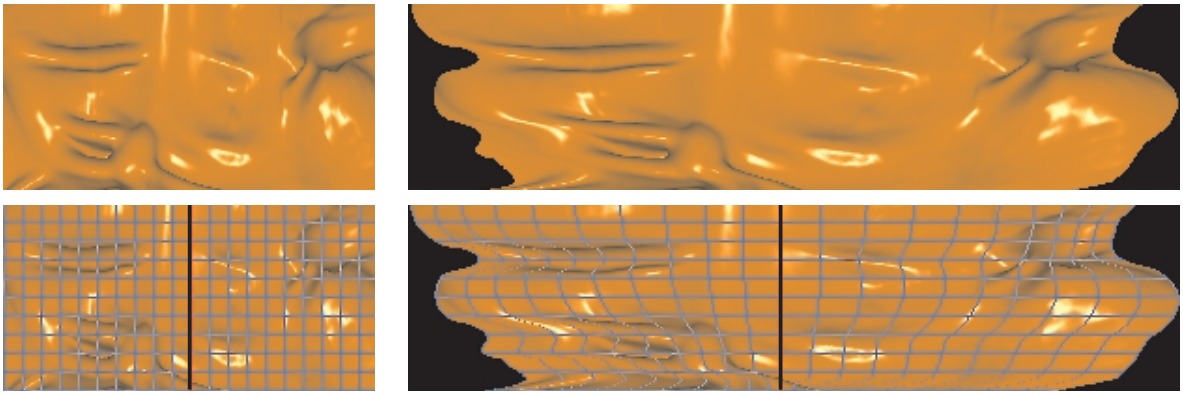
With constant sampling, Polyps can be missed if the angle increment is too large (see figure 8.2a). If the sampling distance is too small, rays are traced where it would not be necessary. This makes the method inefficient.

8.3.2 Perimeter Sampling

In this section, we propose a sampling strategy in which the rays are calculated so that the surface length that they represent is constant.

A constant sample length l is defined. l must be at most half the size of a voxel to keep the Nyquist frequency and therefore not to miss any important feature. l should have the same value as the sampling distance in the h-direction to preserve the ratio, or proportion, in the final mapping.

The algorithm incrementally calculates the ray directions which are in the plane defined by a certain value of h. The angle between the current ray and the next one is computed such that the length of the surface sample that the current ray represents is l in the α-direction (see figure 8.2b). The first ray is



a) b)

Figure 8.3: **a)** Resulting image of the projection technique using constant angle sampling. **b)** Same camera position as a) but with a perimeter sampling. The bottom images show a grid which was generated by fixing a constant angle value.

traced along an arbitrary angle α_0. α_0 must be the same for each value of h. r is defined as the distance from the cylinder axis to the surface point hit by the ray. The surface sample length in the α-direction that a ray represents is approximated by the arc with radius r. Therefore, the value of the angle increment for the next ray is estimated as $\frac{l}{r}$ radians. This projection method projects the organ surface to a generalized cylinder whose radii are not constant within the cylinder. In this case the mapping function f maps the contours and also the surface of the generalized cylinder uniformly. Moving along the central path, contours of varying length are represented by varying numbers of rays. In the mapping to the image plane, this results in the fact that in the v-direction (horizontal scanlines), typically only part of the pixels are covered by an unfolded contour. Therefore, the generalized cylinder is not mapped to the complete domain of the image (see figure 8.3b). The function f maps each sampled ray to a pixel in the image (i.e., each pixel corresponds to an area of size $l \times l$ on the surface). The projected points that correspond to the rays at angle α_0 are positioned on a vertical line in the center of the image. Then from left to right, the ray values are mapped onto the image until the perimeter length is reached.

This projection has the area preservation property, so the relative sizes of surface elements are preserved in the image plane and do not depend on the distance of the cylinder axis to the surface. On the other hand, a distortion is introduced with respect to the h and α-directions, so the angles are not preserved anymore. At the vertical center line of the image there is no distortion, but the distortion increases progressively when we move to the left or right. Figure 8.3b shows an image generated with perimeter sampling with a superimposed grid which would correspond to a regular grid in a constant angle sampling of the cylinder. In this way, it can be observed how the horizontal lines are varying in extent according to the varying length of the corresponding contour.

8.4 Minimally Rotating Frame

In the previous section, a technique has been presented to project the surface of the organ onto a cylinder and then to the image.

At each position of the camera along the central path, an orthogonal coordinate system is taken which specifies the location and orientation of the cylinder. One coordinate axis is given by the tangent vector of the central path. The other axes are in the plane orthogonal to the central path at the camera position. The Frenet frame is not a good choice for this coordinate system. Firstly, the Frenet frame is not defined in linear portions of the central path. Secondly, by moving along the path, the two vectors orthogonal to the tangent vector are rotating more than necessary, thus reducing coherence between adjacent frames. Therefore, we investigate a rotation-minimizing coordinate frame.

We have implemented the rotation-minimizing coordinate frame presented by Klok [Klok86]. The coordinate frame is obtained by solving the following differential equation:

$$\begin{aligned} z(s) &= \frac{c'(s)}{\|c'(s)\|} \\ x'(s) &= -(c''(s)\cdot x(s))\frac{c'(s)}{\|c'(s)\|} \\ y'(s) &= -(c''(s)\cdot y(s))\frac{c'(s)}{\|c'(s)\|} \end{aligned} \tag{8.1}$$

where $c(s)$ represents the parametric central path and $(x(s), y(s), z(s))$ is the coordinate frame we are looking for. An initial orthogonal frame (x_0, y_0, z_0) is defined. Then the differential equations are solved using a fourth order Runge-Kutta method. Theoretically, equations 8.1 produce orthogonal coordinate frames. To avoid accumulation of numerical errors (i.e., orthogonality is not ensured anymore), we take the following approach: $z(s)$ and $x'(s)$ are calculated according to the above formulas. Then $y(s)$ is taken as the cross-product of $z(s)$ and $x(s)$ ($y(s) = z(s) \times x(s)$). Finally we also correct $x(s)$ by taking it as the cross-product of $y(s)$ and $z(s)$ ($x(s) = y(s) \times z(s)$)

8.5 Level Lines Enhancement

Using the distance r of the hit surface-point to the cylinder axis, we can generate a depth image (see figure 8.4a). The depth image together with the shaded image represents a height field, similar to a landscape in topography. A good way to visualize landscapes in topographical maps is showing level lines, where each line corresponds to a level of depth. The level lines improve the perception of depth and surface changes of the map.

Many non-photorealistic techniques use the depth information of an image (i.e., for each pixel the distance of the visible surfaces to the view point) to enhance the image information and improve its perception. The enhancement we want to use is to draw level lines. These level lines help to understand the shape of the object, for example whether an element is a bump or a cavity or how much the surface is changing in an area.

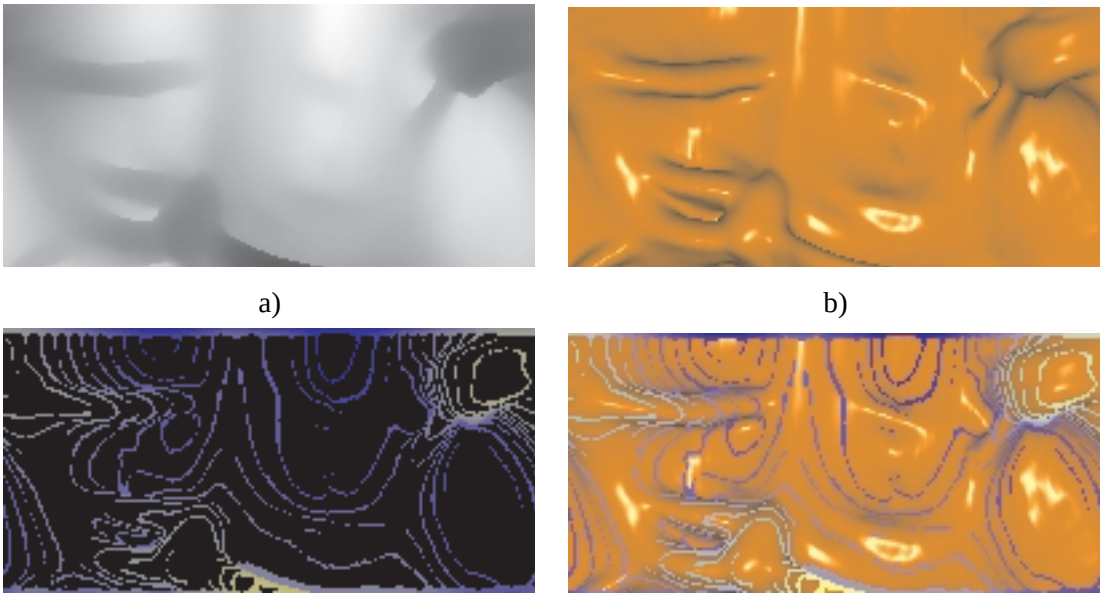
The level lines are generated from the depth image. Firstly, the gradient of the depth image is calculated using a first derivative of the Gauss filter. The level lines are drawn based on the technique described by Saito et al. [Sait90]. To draw a level line, we define a value p_i which corresponds to the depth of level i. p_i is defined discretely by $p_i = p_0 + iq$, where i is an integer and q is the depth interval between levels.

For each pixel of the output image its color value c is computed as follows

$$
\begin{aligned}
c &= c_{background} + f_1\left(\frac{|d - p_k|}{g}\right) * c_k \\
k &= \left\lfloor \frac{d - p_0}{q} + \frac{1}{2} \right\rfloor \\
f_1(t) &= \begin{cases} 0 & 0 \leq t \leq \frac{w}{2} \\ 1 & t \geq \frac{w}{2} \end{cases}
\end{aligned}
$$

where $c_{background}$ corresponds to the color of the image background and c_k (e.g., shaded image) is the color assigned to each level line. d is defined as the value of the depth image in the current pixel and g is its gradient magnitude. The level line width in pixels is described by w. If g is close to zero c is taken to be equal to $c_{background}$.

To improve perception, a hue shift is applied to the color of the level lines color. The c_k values are coded depending on the level of depth (see figure 8.4c). A hue shift has the advantage that it does not interfere with the highlights and dark areas of a shaded image.



a) b) c) d)

Figure 8.4: For the same camera position and using constant angle sampling: **a)** depth image, **b)** shaded image, **c)** level lines with hue shift color coded, and **d)** combination of the level lines and shaded image.

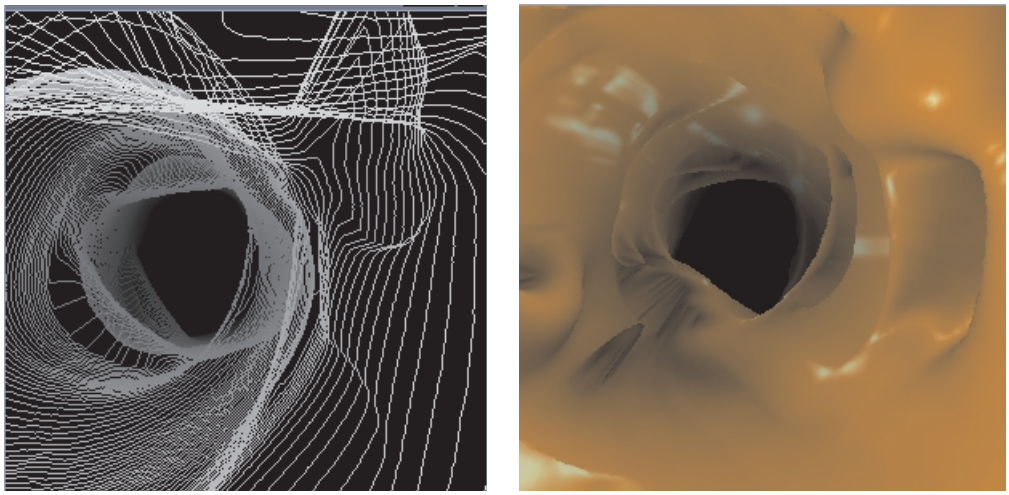
Technical-illustration artists commonly use the temperature of colors in their drawings. The temperature of a color is defined as warm (red, orange and yellow) and cool (blue, violet and green). The temperature also gives depth cue information since the perception of the cool colors recedes whereas the perception of the warm colors advances. The hue shift has been chosen in the range from yellow to blue since these colors have a large shift range, and red-green is undesirable due to color blindness. Yellow corresponds to closer level lines and blue to level lines farther away.

Once the level lines have been obtained, they are combined with the original shaded image (see figure 8.4d). The color of the shaded image should not be in the range between yellow and blue to achieve a good contrast. The contours provide information about the surface change and the distance of the surface to the cylinder axis.

8.6 Endoscopic View Generation

Once a polyp has been detected in the video of the unfolded colon, the physician would be interested in seeing its location with an endoscopic view. Using the already calculated shaded images and depth images for each frame, a fast fly-through can be generated efficiently.

To generate the interactive navigation, the center lines of the depth images of the movie of the unfolded colon are used. Knowing the camera location for each frame, the center lines can be backprojected to 3D space (see figure 8.5a). A polygonal surface can be easily generated using triangle stripes. Each stripe corresponds to the triangles generated between one center line of the depth image and the center line of the next frame. We obtain a fast rendering since we use stripes and we render just the surface in the neighborhood of the camera (see figure 8.5b). This can be achieved easily since the stripes are sorted by path position. With this method we achieve interactive frame rates around 30 f.p.s. with a Pentium II at 400 MHz with common OpenGL graphics hardware acceleration.



a) b)

Figure 8.5: **a)** Endoscopic view backprojecting the lines using the depth information and the camera frame. **b)** Endoscopic view using the backprojection of the generated shaded stripes.

Each triangle can be colored by OpenGL with a correct lighting. Another option is to assign to each triangle vertex its color value calculated in the corresponding shaded image of the video of the unfolded colon. The latter option produces incorrect lighting but it has a better matching with the unfolded colon images.

Obviously, the resulting images are approximations and some artifacts appear due to the cross-section problem (see figure 7.1). However, they give a good impression of the structure and they can be used by the physicians to position the camera to the area that they want to visualize with a better quality but slower rendering.

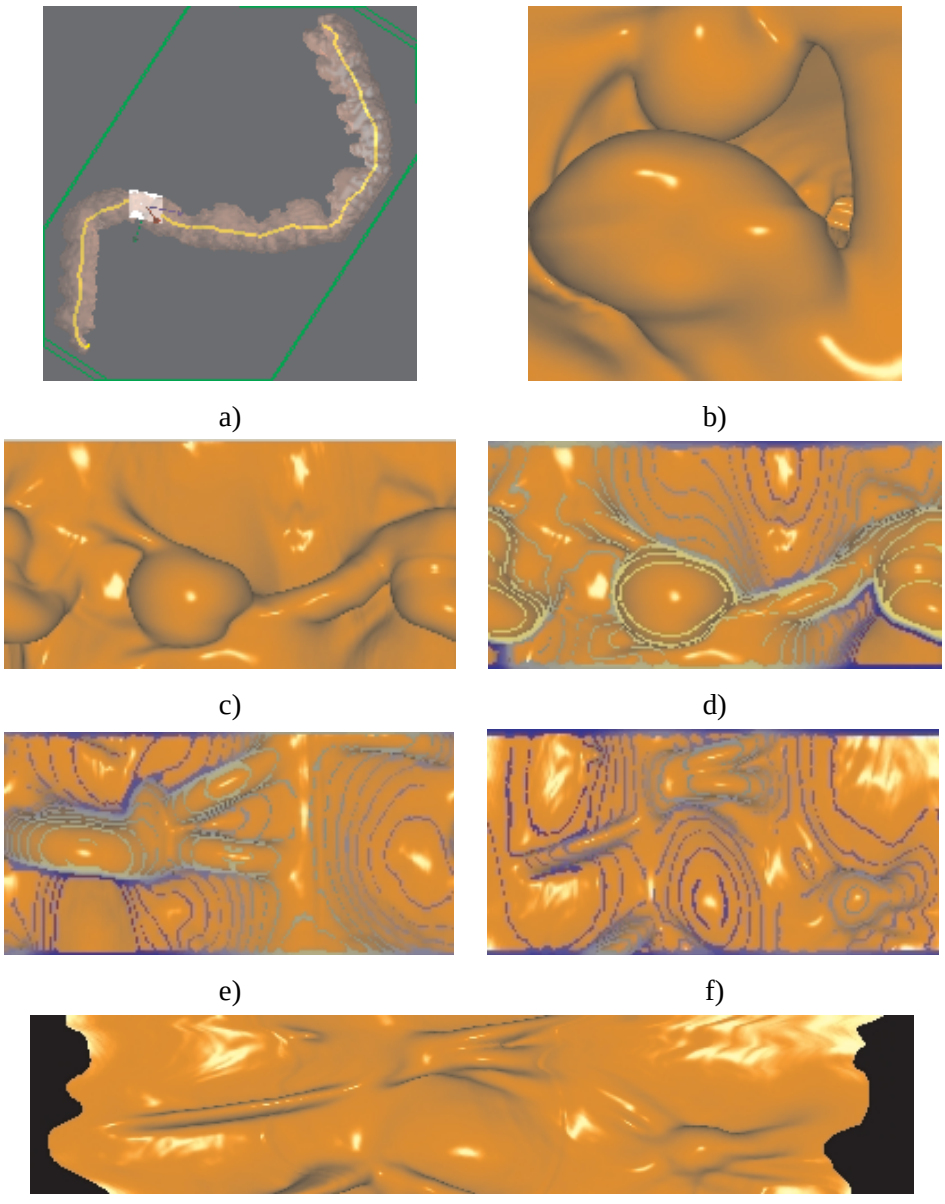


Figure 8.6: **Cadaveric colon CT data set 381x120x632**: **a)** Outside view and camera position for c) and d). **b)** Endoscopic view moving the camera in a) a bit backwards. **c)** Constant angle sampling showing two polyps **d)** The same as c) but with level-lines enhancement. **e)** and **f)** Constant angle sampling from other camera positions showing polyps. **g)** The same as f) but with perimeter sampling.

8.7 Results

The images presented in this paper correspond to a CT data set of a cadaveric colon with a resolution of 381x120x632. The colon is 50 cm long and 13 artificial polyps were physically created in the cadaveric colon. These polyps had a size between 3.5x2.5 mm and 11.8x9.0 mm. Figure 8.6a shows an outside view of the segmented cadaveric colon and its central path together with the camera. The camera position corresponds to the images in figure 8.6c and 8.6d. Figure 8.6b is an endoscopic view moving the camera a bit backwards to show the same region as in figure 8.6c and 8.6d . It can be observed that the shape of the polyps is much more clear in the unfolded images than in the endoscopic view in figure 8.6b (please, refer to http://www.cg.tuwien.ac.at/research/vis/vismed/ColonFlattening/ for the corresponding videos).

The physicians who collaborate in this project, could easily identify the polyps in the unfolded colon images. Figures 8.3, 8.6c, 8.6d, 8.6e, 8.6f and 8.6g show some of the polyps. Figures 8.6f and 8.6g were generated from the same camera position, but the projection was done with constant sampling and perimeter sampling respectively. Both sampling techniques are useful to the physician. Perimeter sampling preserves the area and allows the physician to evaluate the size of the polyps, while constant angle sampling preserves the angles and allows a better evaluation of the shape.

We also tested this method with real data sets without pathologies. Figure 8.7 shows the images generated from a 256x256x311 CT data set of a colon.

a) b)

Figure 8.7: **Colon CT data set 256x256x311**: **a)** Constant angle sampling. **b)** The same camera position as a) but with perimeter sampling.

An application to inspect the videos of the locally unfolded colon has been developed. Once the polyp has been detected, the physician is able to go back to the original data. This is easily done using the camera position of each frame of the video of the locally unfolded colon. The application allows the physician to easily associate the frames of the video with its corresponding region in the original volume data. It consists of different visualization techniques shown in separate windows whose interaction is connected to each other.

One window (see figure 8.8 B) shows consecutive local-unfolded colon images with constant angle sampling. The current camera position corresponds with the camera of the local-unfolded colon image shown at the moment. The user is able to perform a circular shift in the horizontal direction of the image, such that polyps, that appear at the borders, are moved to the center and can be inspected with less distortion. An overview image is generated using the horizontal center lines of each frame (see figure 8.8 A). In the overview image, the current camera position is indicated by highlighting its corresponding horizontal center line. The overview image includes artifacts as the cross-sections that correspond to each line may intersect each other, but the image gives enough context information to orient the user.

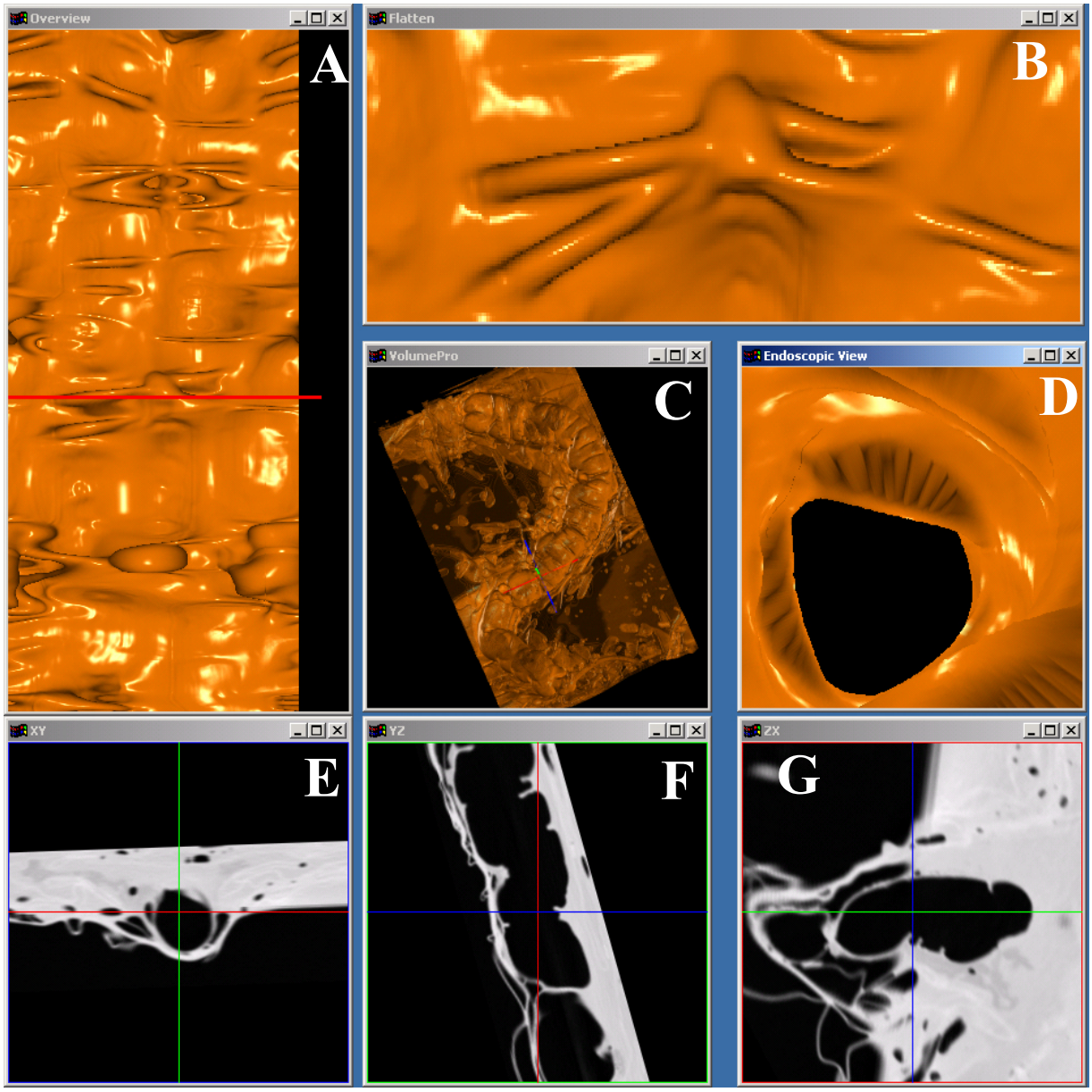


Figure 8.8: Snapshot of the application developed to inspect the video result of local unfolding of the organ.

Another window with an outside view showing the camera position is generated using the VolumePro board (see figure 8.8 C). Furthermore, there are three windows that show the three orthogonal planes corresponding to the three coordinate planes of the current camera frame (see figure 8.8 E,F and G). These planes are generated using multiplanar reconstruction.

A window showing a common perspective view of the data set (see figure 8.8 D) is also present. This window is updated with the current camera position. If the user clicks in an area of the unfolded colon

image (see figure 8.8 B), an update of the perspective view camera is done. The camera of the perspective view points to the same area that has been pointed to in the unfolded frame.

The relationship between windows and different visualization techniques provides a suitable tool for inspection of the data set.

It is important that the cylinder axes do not get outside the organ. The cylinder height could be optimized for each data set and even adaptively defined depending on the camera position.

The camera movement has to be specified in a way that we do not miss any surface region. In the current implementation, the camera steps are so small that they ensure this, but the method is also inefficient because unnecessary images might be calculated. Once a movement step of the camera has been specified, it can be calculated whether a visible surface area is missed by intersecting the plane of the top cover of the cylinder and the bottom cover of the next one. If they intersect in a position far away from the estimated colon radius, it can be ensured that no surface area visible from the central path will be missed. Otherwise, the movement step should be adapted. However, the surface behind folds with mushroom-like shapes will not be shown with our method. A solution to this problem should be investigated.

Chapter 9

Nonlinear Colon Unfolding

> I love fools' experiments; I am always making them.
>
> *Charles Darwin (1809-1882)*

9.1 Introduction

In chapter 8, a method for unfolding the colon was presented. This method results in a video with correct unfolded portions of the colon in each frame. It solves problems of previous techniques, like double polyp appearance (i.e., the same polyp can show up more than once) and undersampling. However, it has the drawback that the physician has to review a video and cannot visualize the complete surface at once.

In this chapter, we describe a new method [Vila01a] to obtain a single view of the complete unfolded colon. The physician can easily detect areas where polyps are located, and additionally get an idea of their shape and size. Afterwards, the physician can concentrate on an exhaustive inspection of the problematic areas. This method proposes solutions to the problems of double appearance and undersampling. In our approach, the colon unfolding does not just produce a surface but also a height field (distance of the colon surface to a central path). This avoids that the polyps are flattened as with the methods proposed by Haker et al. [Hake00a]. Furthermore, the height field gives a more natural visualization than a flattened surface with color-coded mean curvature.

In the next sections, this method will be discussed in detail. Section 9.2 gives an overview of the new approach. Sections 9.3 and 9.4 present in detail the main steps of the method. Results are presented in section 9.5.

9.2 Method Overview

Our method of unfolding the colon can be divided into two main steps: nonlinear ray casting, which solves the problem of double appearance of polyps, and nonlinear 2D scaling, which improves the nonuniform sampling and avoids to miss polyps.

The input data of the technique is a segmented volume data set of a colon. The segmentation is needed to find the central path. It is important that the segmentation is conservative. The segmentation mask does not contain any point outside of the colon. The central path is computed by thinning the segmented volume. The path is smoothed, but it is ensured that it does not cross the colon surface. We use the algorithm presented in chapter 5. A distance map is generated from the calculated central path [Lohm98]. The voxels of the distance map contain the distance to the nearest voxel on the central path.

A coordinate frame is moved along the path. For each position, rays are traced starting in the plane orthogonal to the path, and following radial directions (constant angle sampling). To avoid double polyp appearance in high curvature areas of the path, the rays follow the negative gradient direction of the precalculated distance map. The rays are not straight lines any more, and they do not cross each other, at most they meet (see figure 9.1). Nonlinear ray casting has already been investigated before [Gröl95]. Section 9.3 explains how these rays are traced.

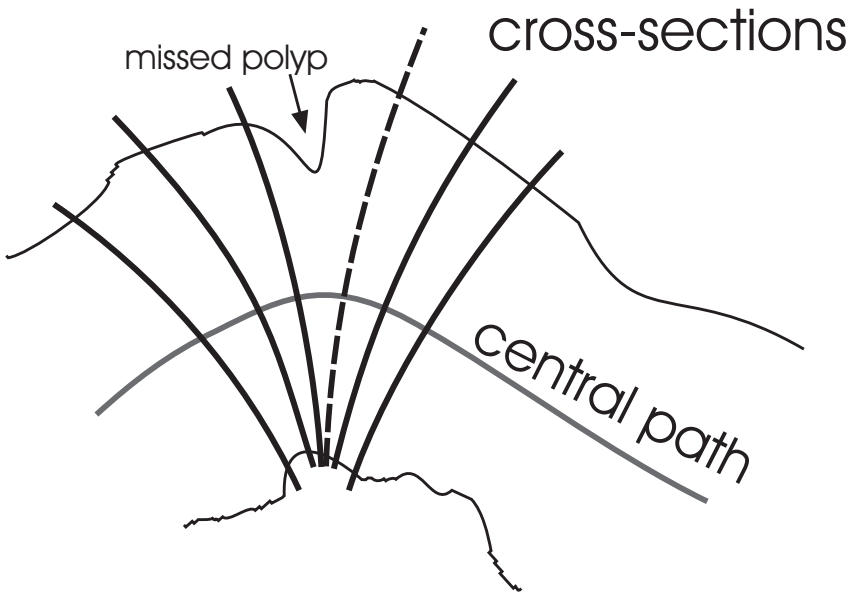


Figure 9.1: Elimination of double polyp appearance by nonlinear ray casting.

Along each curved ray, the volume is sampled in a uniform way and direct volume rendering is performed. The ray is terminated when it hits the surface of the colon. The result of the nonlinear ray casting can be interpreted as a 2D cylindrical parameterization of the inner colon surface. One parameter corresponds to the position along the path. The second parameter specifies the ray within the plane orthogonal to the current path position. The distances between ray origins on the central path and intersected points on the colon surface determine a height field. The height field is unfolded, and the result corresponds to a parallel projection of the unfolded height field.

Nonlinear ray casting samples the height field nonuniformly. A straightforward unfolding to a regular grid contains severe area distortions and is therefore not optimal. In the second step, an iterative scaling transforms the previously generated 2D parameter grid in order to compensate for these distortions. After the scaling, the ratios between the area that the samples represent and their area in the 2D grid are

approximately equal. The second step is based on nonlinear 2D scaling that is used in a similar way for magnification fields in information visualization [Keah97]. In section 9.4, the algorithm is described in detail. Next, the colon surface is resampled with an almost-uniform sampling rate using the transformed 2D grid.

In the next section, we describe the major steps of our algorithm in more detail.

9.3 Nonlinear Ray Casting

The curve $\mathbf{c}(v)$, which represents the central path of the colon, is calculated using thinning and the smoothing algorithm presented in our previous work [Vila00]. In order to trace the nonlinear rays, a distance map $Dist(\mathbf{p})$ of a point $\mathbf{p}$ to the central path $\mathbf{c}(v)$ is computed. $Dist(\mathbf{p})$ is calculated in discrete space. $Dist(\mathbf{p})$ is a volume data set that contains for each voxel the minimum distance to the path. A reconstruction filter is used to approximate the distance map in continuous space $dist(\mathbf{p})$.

For the calculation of $Dist(\mathbf{p})$, a minor modification of the Euclidean distance transformation presented by Lohmann [Lohm98] is used. The modified distance transformation saves for each voxel the vector $\mathbf{vDist}(\mathbf{p})$ from the voxel to the closest central path point, $Dist(\mathbf{p}) = \|\mathbf{vDist}(\mathbf{p})\|$. Using this information, the distance map $dist(\mathbf{p})$ in any point $\mathbf{p} \in \mathbb{R}^3$ is defined using the following reconstruction function.

$$dist(\mathbf{p}) = \{\, min(\|\mathbf{p_i} + \mathbf{vDist}(\mathbf{p_i}) - \mathbf{p}\|) \mid \forall i\ 0 \leq i \leq 7,\ \mathbf{p_i} \in \mathbb{N}^3 \,\}$$

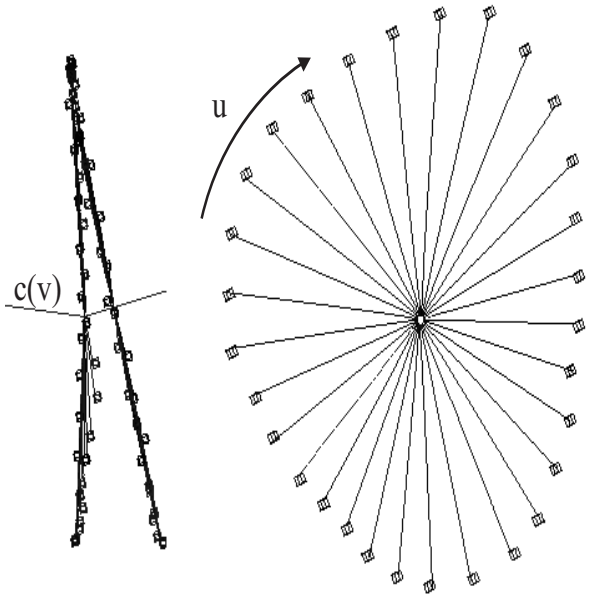
where $\mathbf{p_i}$ represent the eight vertices of the cell that contains $\mathbf{p}$. This reconstruction gives the exact value when one of the eight neighbors $\mathbf{p_i}$ has the same closest curve point as $\mathbf{p}$. Trilinear interpolation of the scalar values $Dist(\mathbf{p_i})$ gives a good performance when the distance map has a linear behavior, which happens just in some special cases.

The nonlinear rays are traced from the central path in uphill direction, i.e., along the negative gradient direction, $-\nabla\mathbf{dist}(\mathbf{p})$, of the distance map $dist(\mathbf{p})$. $dist(\mathbf{p})$ is continuous in the first derivative nearly everywhere, except for ridge and valley lines. The distance map $dist(\mathbf{p})$ induces a vectorfield $-\nabla\mathbf{dist}(\mathbf{p})$ which is defined by the gradient of the distance map. It is known that trajectories of such vectorfields will not cross each other and are unambiguous (see Abraham et al. [Abra92] for details). These trajectories will correspond to our nonlinear rays. Furthermore, in our situation, they will not produce cycles, since it is impossible to return to the same point if you always move uphill. In the worst case, the nonlinear rays will merge in ridge and valley lines, but they will not cross. With these curved rays the appearance of double polyps is avoided and an unambiguous and correct parameterization of the colon inner surface is obtained.

9.3.1 Casting of Nonlinear Rays

The first step to trace the nonlinear rays is to move a coordinate frame along the curve $\mathbf{c}(v)$. The frame is moved using a minimal-rotation frame (see section 8.4 for more details). For each position on the path, a constant number of rays is traced. The initial point of each ray is placed in the plane orthogonal to the path. Note that the gradient is not defined along the path $\mathbf{c}(v)$ since it is a valley line of the distance map $dist(\mathbf{p})$. Therefore, the initial points are located in a circular arrangement at a small distance from

the path position. Once the initial points have been determined (u parameterization), the rays are traced incrementally following the negative gradient of the distance map (see figure 9.2).



a) b)

Figure 9.2: Nonlinear rays traced from a specific position along the path $\mathbf{c}(v)$: a) the curved rays in areas of high curvature for two consecutive points along $\mathbf{c}(v)$, b) nonlinear rays traced in u direction.

The rays have the tendency to be perpendicular to the path $\mathbf{c}(v)$ since it is the direction of maximal change of $dist(\mathbf{p})$ in linear segments of the path. The rays become curved in areas where the curvature of the path increases (see figure 9.1 and 9.2).

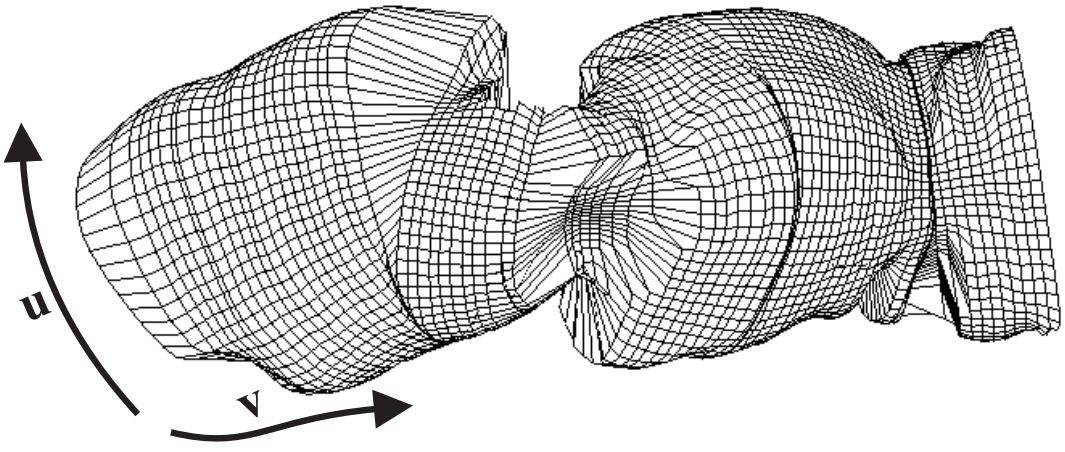


Figure 9.3: Surface obtained after nonlinear ray casting. The sampling of the surface is nonuniform.

9.3.2 Colon Surface Parameterization

In the previous section, nonlinear and especially non-crossing rays were traced from the central path $\mathbf{c}(v)$ towards the colon surface. While the rays are traced, direct volume rendering is performed. The ray terminates when the colon surface is hit. The result of the nonlinear ray casting is a sampling of the inner surface of the organ.

The tracing of the nonlinear rays defines an unambiguous parameterization of the inner colon surface $\mathbf{s}(u,v)$. Here, v is the parameter along the central path $\mathbf{c}(v)$, and u is the radial angle along which the nonlinear rays are started ($u \in [0, 2\pi]$).

Figure 9.3 shows $\mathbf{s}(u,v)$ which results from applying nonlinear ray casting to a piece of the colon. The lines correspond to the isolines of the parametric surface $\mathbf{s}(u,v)$. The parameter space is sampled uniformly in the u and v direction, but this does not correspond to a uniform sampling of $\mathbf{s}(u,v)$.

Unfolding of the $\mathbf{s}(u,v)$ surface can easily be done by mapping $\mathbf{s}(u,v)$ onto a regular grid in the 2D u,v-parameter space. In figure 9.4a a parameterization of the colon surface is done with straight rays (ambiguous, non-uniform sampling). In figure 9.4b, a parameterization of the colon surface is done with curved rays (unambiguous, but still non-uniform sampling).

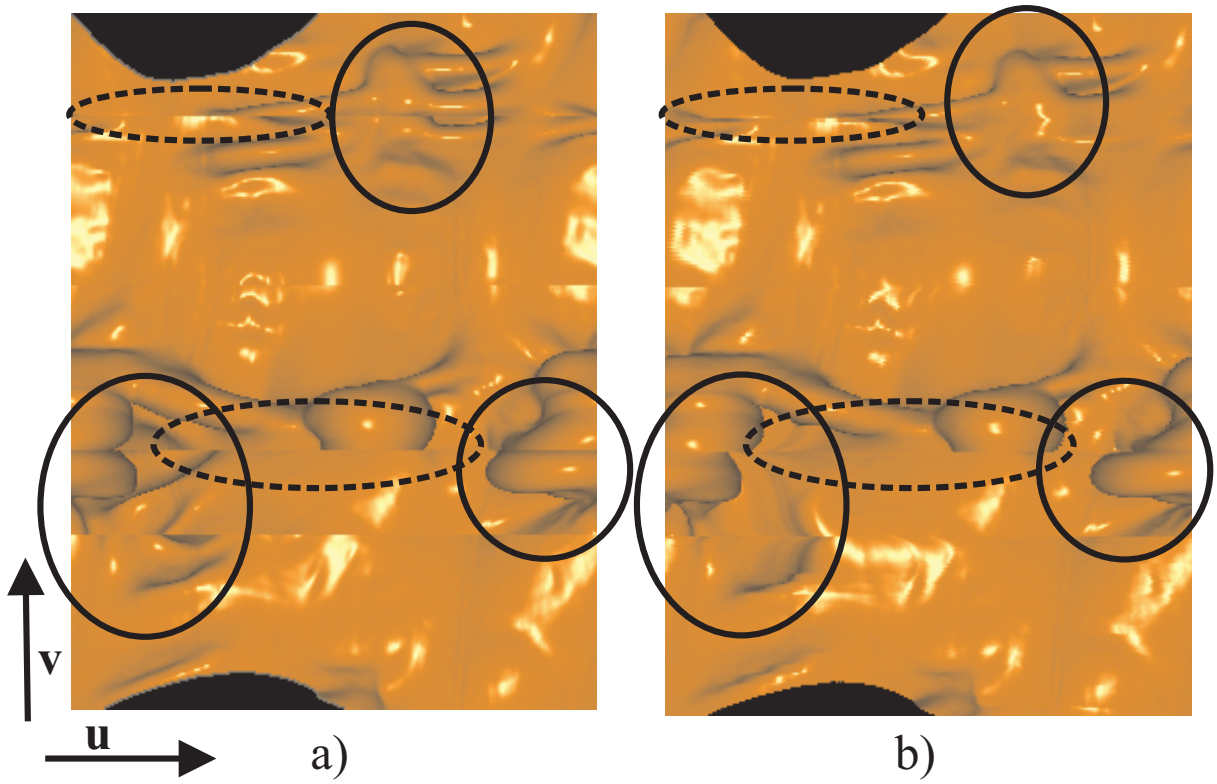


Figure 9.4: a) Unfolding of the colon surface of the data set presented in figure 9.3 using straight rays. Solid circles indicate double polyp appearance areas. Dashed circles indicate undersampled areas. b) unfolding of the parametric surface generated using nonlinear rays shown in figure 9.3, by mapping the parametric space to a regular grid. Double polyps disappear, but undersampled areas not.

Nonlinear ray casting avoids that features appear more than once, but on the other hand the sampling of the surface is far from being uniform. There are oversampled areas, which lead to geometric deformations, and also undersampled areas. In the latter case, deformations appear but also features of the surface can be missed.

In figure 9.4a, the solid circles indicate areas where features appear more than once. Using nonlinear ray casting, the double polyps disappear, and instead an enlargement of the feature appears (figure 9.4b). The areas encircled with dashed circles indicate undersampled areas and therefore areas where features are possibly missed. Note that the same undersampled areas are present in both figures.

In the next section, an algorithm is presented to obtain an unfolding of the parametric surface $\mathbf{s}(u, v)$ which avoids geometric deformations and to miss features.

9.4 Nonlinear 2D Scaling

In the previous section, an unambiguous parameterization of the inner colon surface projected to the central path has been introduced. The sampling of the surface $\mathbf{s}(u, v)$ defines a quadrilateral mesh on the colon surface (see figure 9.3). The curved rays generated by the nonlinear ray casting do not intersect, therefore, the resulting quadrilateral mesh is valid and also not self intersecting. Furthermore, the distance between the surface point $\mathbf{s}(u, v)$ and the corresponding path position $\mathbf{c}(v)$ defines a height field $r(u, v)$.

The goal of nonlinear 2D scaling is to achieve a 2D grid which approximates a parallel projection of the unfolded height field defined by the nonlinear ray casting. We desire that the unfolded height field preserves the length of the edges of $\mathbf{s}(u, v)$ in u and v direction and, therefore, we also preserve the area. In section 9.4.1, we define the length of the edges of the 2D grid to achieve the area preservation. Section 9.4.2 describes the iterative algorithm which yields a 2D grid with such edge lengths.

9.4.1 Height field unfolding

Figure 9.5a is an illustration of a cross-section of the height field for a given value of v. The unfolding of the height field $r(u, v)$ in figure 9.5a to a 2D regular grid is shown in figure 9.5b. The parallel projection of the unfolded height field corresponds to the mapping of the surface $\mathbf{s}(u, v)$ presented in section 9.3.2. The edges in the 2D grid have a constant length e. The unfolded object is locally scaled depending on the height-field value $r(u, v)$. If $r(u, v)$ has a high value (i.e., $\mathbf{s}(u, v)$ is far from the center) the object shrinks. If $r(u, v)$ has a small value (i.e., $\mathbf{s}(u, v)$ is close to the center), the object is enlarged. This is since the area that the samples represent depends on their distance to the central path and is not constant as is assumed by mapping them to a regular grid.

From the 3D quadrilateral mesh obtained in the nonlinear ray casting, we know the distances between adjacent quadrilateral vertices (i.e., the length of the edges of the quadrilateral). If these distances are preserved in the 2D grid, the sizes of the quadrilaterals will be preserved. Figure 9.5c shows the result of preserving the 3D edges of the quadrilateral mesh that corresponds to figure 9.5a. Note that geometric deformations also appear. In this method, we flatten the surface and the polyps. This is due to the fact that we do not take the height field into account. We do not want that the 2D grid has the same edges as

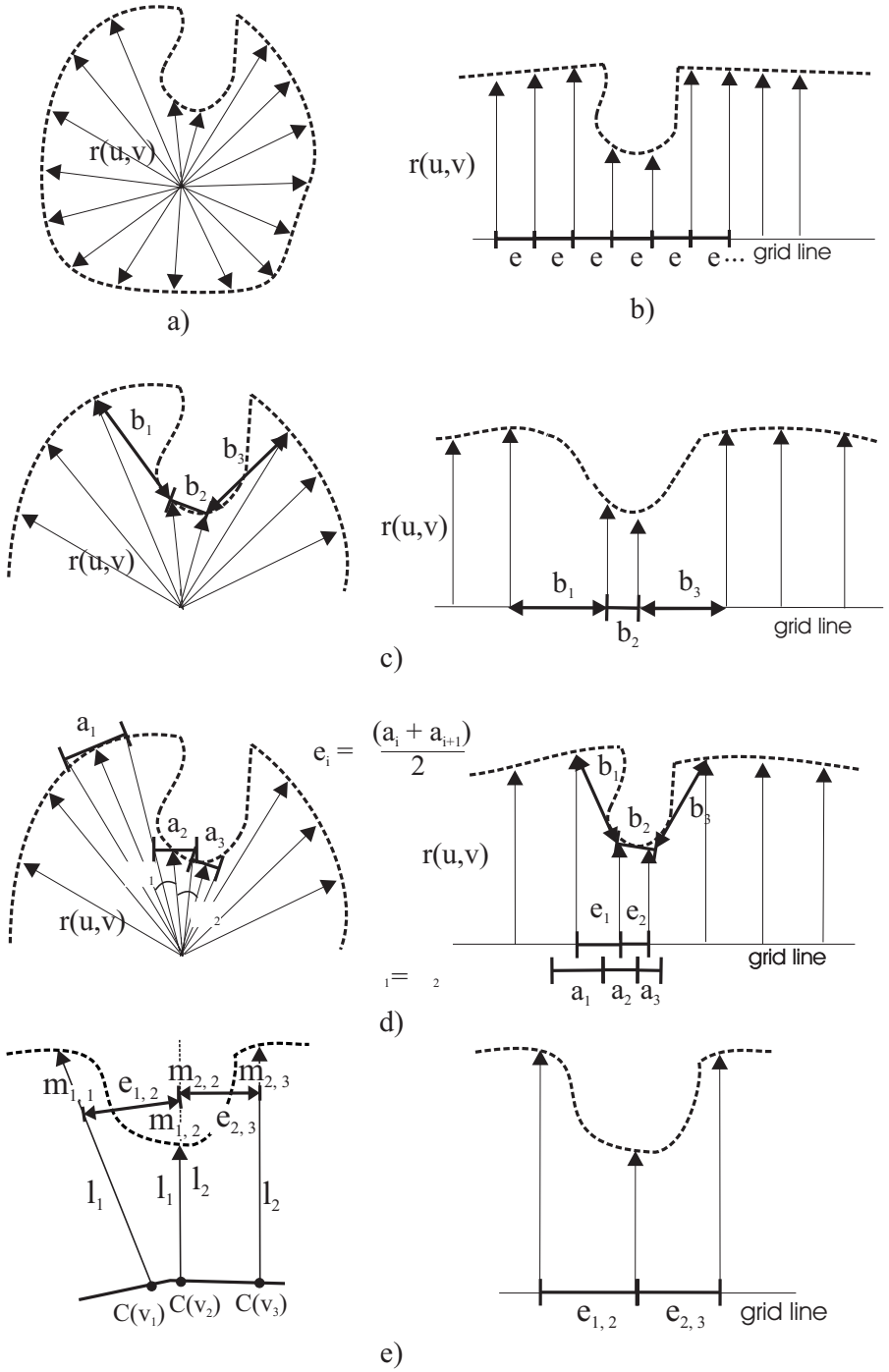


Figure 9.5: Illustration of height field unfolding in u direction: a) cross-section of $r(u, v)$ for a fixed value of v, b) unfolding to a regular grid, c) unfolding preserving the edge lengths b_i of the 3D quadrilateral mesh in the 2D grid, and d) unfolding preserving the edge lengths b_i of the 3D quadrilateral mesh in the height field. e) Unfolding in the v direction.

the 3D quadrilateral grid, but we want that the edges of the unfolded height field have the same length as the edges of the 3D quadrilateral mesh (see figure 9.5d). This implies that the distance between edges in the 2D grid should correspond to the length a that each sample represents in u and v directions.

To calculate the edge lengths, we use different approximations for the u and v direction. For simplicity, we define $\mathbf{w_{i,j}}$ as a point in parametric space (u_i, v_j) (u_i is the ith sampled parameter value in the u direction and v_j the jth in the v direction). In the u direction (see figure 9.5d), each sample represents a length approximated by

$$a(\mathbf{w_{i,j}}) = 2 * \tan\left(\frac{\alpha}{2}\right) * r(\mathbf{w_{i,j}})$$

where α is the angle between the straight rays from the path to $\mathbf{s}(\mathbf{w_{i,j}})$ and $\mathbf{s}(\mathbf{w_{i+1,j}})$. For simplicity, we illustrate in figure 9.5d the case where the angle α is the same between consecutive rays. In general, α is not constant due to the nonlinear rays. The edge distance between two consecutive samples in the u direction is approximated by

$$e(\mathbf{w_{i,j}}, \mathbf{w_{i+1,j}}) = \frac{a(\mathbf{w_i}) + a(\mathbf{w_{i+1,j}})}{2} \tag{9.1}$$

This equation cannot be used for the v direction since angle α is not defined. Therefore, in the v direction the following expression is used

$$\begin{aligned} l_j &= \frac{r(\mathbf{w_{i,j}}) + r(\mathbf{w_{i,j+1}})}{2} \\ \mathbf{m_{j,k}} &= \mathbf{c}(v_k) + \frac{\mathbf{s}(\mathbf{w_{i,k}}) - \mathbf{c}(v_k)}{r(\mathbf{w_{i,k}})} * l_j \\ e(\mathbf{w_{i,j}}, \mathbf{w_{i,j+1}}) &= \|\mathbf{m_{j,j}} - \mathbf{m_{j,j+1}}\| \end{aligned} \tag{9.2}$$

If the rays are parallel, equation 9.2 results in the projection of the edge between two samples to the central path (see figure 9.5e). Equation 9.2 approximates the distance between two samples subtracting the distance due to the difference in height of the two samples (i.e., the distance from the sampled point to the central path).

Using equations 9.1 and 9.2, we have defined the length of the edges of the 2D grid such that the unfolded height field preserves the length of the edges of $\mathbf{s}(u, v)$ in u and v direction. In the next section, an algorithm to obtain such a 2D grid is presented.

9.4.2 Nonlinear 2D scaling

The objective of the nonlinear 2D scaling algorithm is to generate a 2D grid whose edges preserve the length of the edges calculated by $e(\mathbf{w_{i,j}}, \mathbf{w_{k,l}})$ (see equation 9.1 and 9.2). To find an analytical solution to the problem is too complex, so a numerical solution is adopted. To generate such a grid we use an approach similar to the one presented by Keahey et al [Keah97]. The main difference is that the new algorithm not only preserves areas, but also the edge lengths.

A transformation of a 2D regular grid is defined by function $\mathbf{T}(i, j) : \mathbb{N}^2 \rightarrow \mathbb{R}^2$. $\mathbf{T}(i, j)$ has to be C^0-continuous and it should preserve the order (i.e., no edge or grid node flipping). For $\mathbf{T}(i, j) = (x, y)$ and $\mathbf{T}(k, l) = (x', y')$ the condition to preserve the order is defined by

$$i < k \Longleftrightarrow x \leq x' \quad j < l \Longleftrightarrow y \leq y' \tag{9.3}$$

We define a 2D scaling field S as a field of scalar values for each edge. Each scalar value indicates the scaling factor that a transformation $\mathbf{T}$ has applied to the edge. The 2D scaling field S for an edge defined between $\mathbf{T}(i,j)$ and $\mathbf{T}(k,l)$ is $S(i,j,k,l) := \|\mathbf{T}(i,j) - \mathbf{T}(k,l)\|$. A 2D scaling field S is defined for any transformation $\mathbf{T}$.

The goal of the nonlinear 2D scaling algorithm is to find a transformation $\mathbf{T_g}$ such that for all values of i and j the equation $e(\mathbf{w_{i,j}}, \mathbf{w_{k,l}}) = \|\mathbf{T_g}(i,j) - \mathbf{T_g}(k,l)\|$ holds, where $\mathbf{w_{k,l}}$ is a 4-connected neighbor of $\mathbf{w_{i,j}}$. In other words, we want to find a transformation $\mathbf{T_g}$ whose 2D scaling field is $S_g(i,j,k,l) = e(\mathbf{w_{i,j}}, \mathbf{w_{k,l}})$ (see equations 9.1 and 9.2) for each edge of the grid.

The major problem is to find the coordinates (x,y) of the transformation $\mathbf{T_g}$, given the scalar values of the 2D scaling field S_g. It is clear that for the same 2D scaling field several transformations are possible.

We have used an iterative method which provides a numerical solution. The goal of the algorithm is to find a transformation $\mathbf{T_a}$ that provides a good approximation of $\mathbf{T_g}$.

Given a transformation $\mathbf{T_a}$, the corresponding scaling field can be easily calculated by $S_a(i,j,k,l) = \|\mathbf{T_a}(i,j) - \mathbf{T_a}(k,l)\|$. A scaling field error can then be computed by $S_e = S_g - S_a$. S_e gives the difference between the computed scaling field S_a and the desired scaling field S_g.

The iterative algorithm starts with T_a as a regular grid. Then S_a and S_e are calculated. The algorithm iterates over each node of the grid. For each node, the value of S_e at each of the 4-connected neighbors is investigated. If $S_e > 0$ (i.e., the edge is not long enough) then the neighbor is moved away from the node . If $S_e < 0$ (i.e., the edge is too long) then the neighbor is pulled towards the node. The edge is modified by a length of $\frac{S_e(i,j,k,l) * C_r}{2}$ where $C_r \in [0,1]$ is a parameter of the algorithm. The division by 2 is necessary because each edge is treated twice, once for each limit node of the edge. Changing an edge is thus done by modifying each of its limit nodes. An important requirement of the algorithm is to preserve the order. So the neighbors are moved as far as S_e and C_r allow without violating equation 9.3.

The neighboring nodes are changed with coordinate-aligned movements. The movements correspond to the original orientation of the edges in the regular grid (i.e., horizontal and vertical). The movement is computed such that the resulting edge has the expected length determined by S_e and C_r. These movements have the tendency to preserve the rectangular appearance of the quadrilateral defined by $\{\mathbf{T_a}(i,j), \mathbf{T_a}(i+1,j), \mathbf{T_a}(i+1,j+1), \mathbf{T_a}(i,j+1)\}$, which, for example, does not degenerate to a triangle.

Once the iteration has run for all the nodes, the new $\mathbf{T_a}$ is generated. Then, a new S_a and a new S_e are calculated from the resulting $\mathbf{T_a}$. S_e is calculated just once per iteration.

The algorithm's convergence depends on the complexity of the scaling field S_g and the value of C_r. If $C_r = 0$, no movement will occur. If $C_r = 1$, the neighbors will be moved the maximum possible displacement. This can make the convergence faster, but can also lead the approximation to a state where the algorithm does not converge at all. A trade-off between speed and quality has to be made in defining the value of C_r.

The convergence factor is measured using the distance between the approximated scaling field S_a and

the desired 2D scaling field S_g. This is expressed as the root mean squared error σ of S_e. Given a grid of size $n \times m$ the convergence factor σ is defined as follows.

$$\sigma = \sqrt{\frac{\sum_{j=0}^{m-1}\sum_{i=0}^{n-2} S_e(i,j,i+1,j)^2 + \sum_{j=0}^{m-2}\sum_{i=0}^{n-1} S_e(i,j,i,j+1)^2}{2nm-n-m}}$$

The algorithm terminates, if σ becomes smaller than a defined constant or if after several iterations no considerable improvement occurs anymore.

The convergence of the algorithm can be improved by starting with a $\mathbf{T_a}$ which is a closer approximation of the desired result than a regular grid. The length of the edges within a horizontal line (i.e., the horizontal edges between nodes with the same j value) are set such that the line approximates the perimeter of the colon in the corresponding cross-section (i.e., $\frac{\sum_{i=0}^{n-2} S_g(i,j,i+1,j)}{n-1}$). The distance between two consecutive horizontal lines is set to the average of the vertical edge lengths in S_g which join the nodes between the two lines (i.e., $\frac{\sum_{i=0}^{n-1} S_g(i,j,i,j+1)}{n}$). Starting with this modified grid leads to a faster convergence by generating basically the same result as a regular grid would give.

The pseudocode to illustrate the presented algorithm is given below.

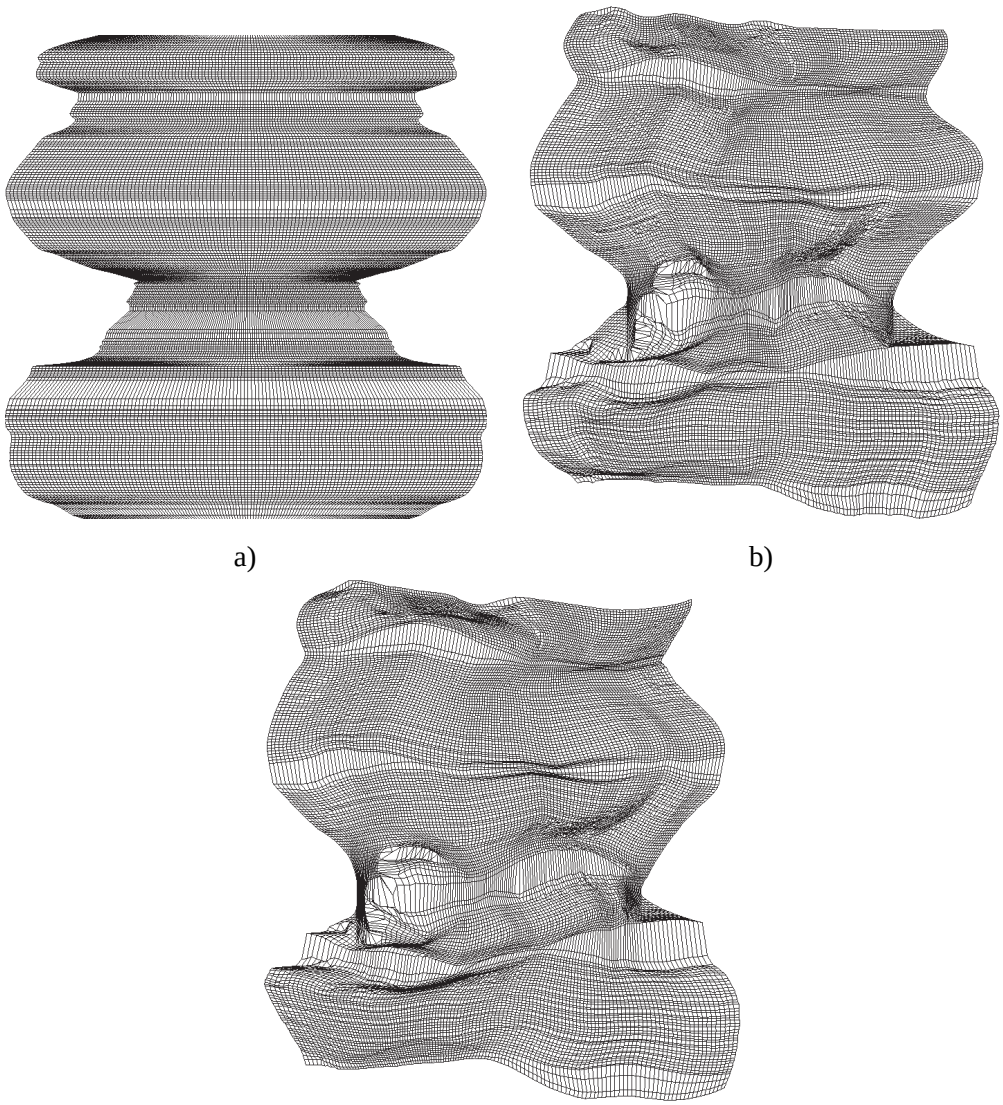
```
proc NonLinear2DScaling(
            in:    2DScalingField Sg, float Cr,
            out:   2DTransformation Ta)
{
  2DScalingField Sa;
  2DScalingField Se;
  boolean bNoEnd;

  Ta = IntialGrid(Sg);
  Sa = Calculate2DScalingField(Ta);
  Se = Sa − Sg;
  σ = CalculateRootMeanSquare(Se);
  bNoEnd = FinishApproximation(σ);

  while (bNoEnd) {
    forall Ta(i, j){ MoveNeighbors(Ta, i, j, Se, Cr);}
    Sa = Calculate2DScalingField(Ta);
    Se = Sa − Sg;
    σ = CalculateRootMeanSquare(Se);
    bNoEnd = FinishApproximation(σ);
  }
}
```

Figure 9.6a shows the initial grid $\mathbf{T_a}$ for the segment of the colon presented in figure 9.3. The resolution of the grid is 128x171 and the initial value of σ is 0.8008. After 960 iterations $\mathbf{T_a}$ has been evolved into

the grid in figure 9.6b. The value of σ is 0.2808. Figure 9.6c is the result of the algorithm after 1687 iterations. The grids of figures 9.6b and 9.6c are similar. In figure 9.6c, the value of σ is 0.2650.

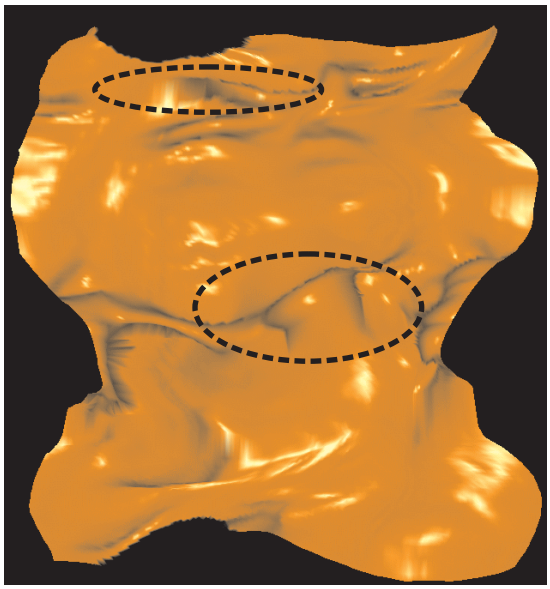


a) b)

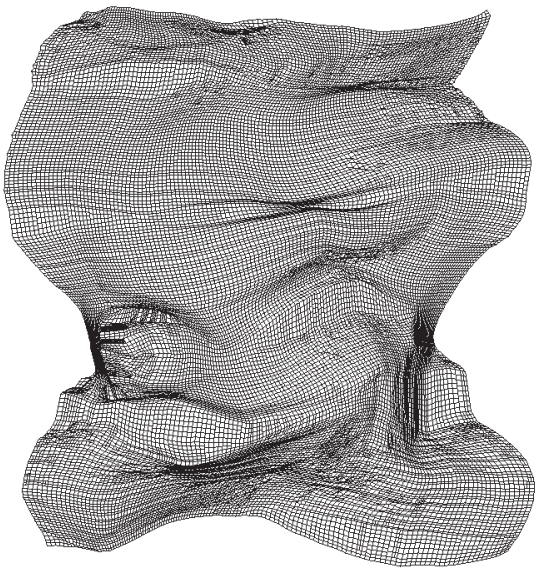
c)

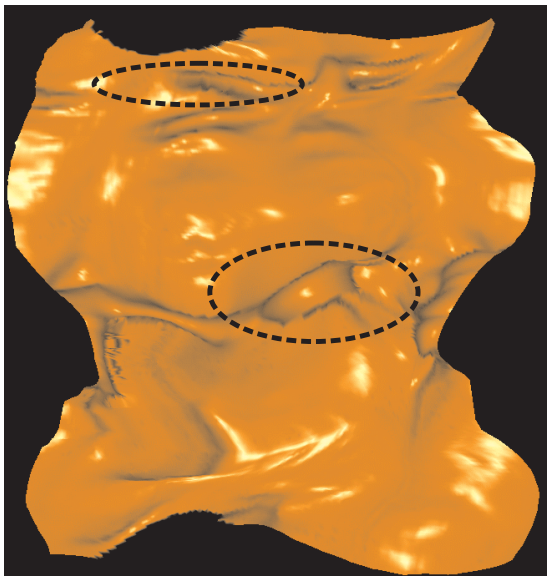
Figure 9.6: Illustration of the nonlinear 2D scaling algorithm using the same data set as in figure 9.3. a) Initial $\mathbf{T_a}$ corresponding to a 128x171 grid. b) $\mathbf{T_a}$ after 960 iterations of the algorithm. c) $\mathbf{T_a}$ after 1687 iterations.

Once the nonlinear 2D scaling has been performed, we obtain a parallel projection of the unfolded height field to a 2D grid preserving the edge lengths. This avoids deformation due to the nonuniform sampling. In the next section, a method to resample the undersampled areas and to detect possible missing features is presented.



a)





b) c)

Figure 9.7: Resampling after the nonlinear 2D scaling. a) 128x171 shaded grid using bilinear interpolation. b) the resulting grid of the nonlinear 2D scaling after resampling c) Shading of the resampled grid.

9.4.3 Resampling

The nonlinear 2D scaling provides a mapping between the 3D quadrilateral mesh and a 2D grid avoiding geometric deformations. The color of each ray obtained in the nonlinear ray casting step is assigned to its corresponding node in the 2D grid. Bilinear interpolation is used to fill the quadrilaterals of the grid. An

example can be seen in figure 9.7a. The areas encircled by dashed ellipses are the same as in figure 9.4. Some features are missing due to undersampling.

The undersampled areas are easily identifiable from the 2D grid. A minimum sample step h for the 2D grid is defined. The sample step corresponds directly to a sample step in the 3D space. A quadrilateral is subdivided if at least one of its edges is longer than h. For a quadrilateral, the subdivision consists in generating a subgrid whose edge lengths are smaller or equal to h. The result of subdividing the quadrilaterals of a grid can be seen in figure 9.7b.

The resulting subdivided grid has assigned color values just in the nodes of the original quadrilateral. Therefore a resampling of the colon surface has to be done for each of the newly generated nodes.

Each point in the grid can easily be identified with its corresponding point in 3D using linear interpolation. The 3D points do not correspond to surface points, but they are close to the colon surface. Therefore, curved rays are traced several steps backwards following the nonlinear ray casting algorithm. Then the rays are traced forward again to find the correct surface point. The color of the sampled point is determined by the same procedure as described in section 9.3.

The resulting color values are mapped directly to the corresponding point in the 2D grid. The results of the resampling procedure can be seen in figure 9.7c. The encircled areas show regions where features that were not present in figure 9.7a have been identified.

9.5 Results

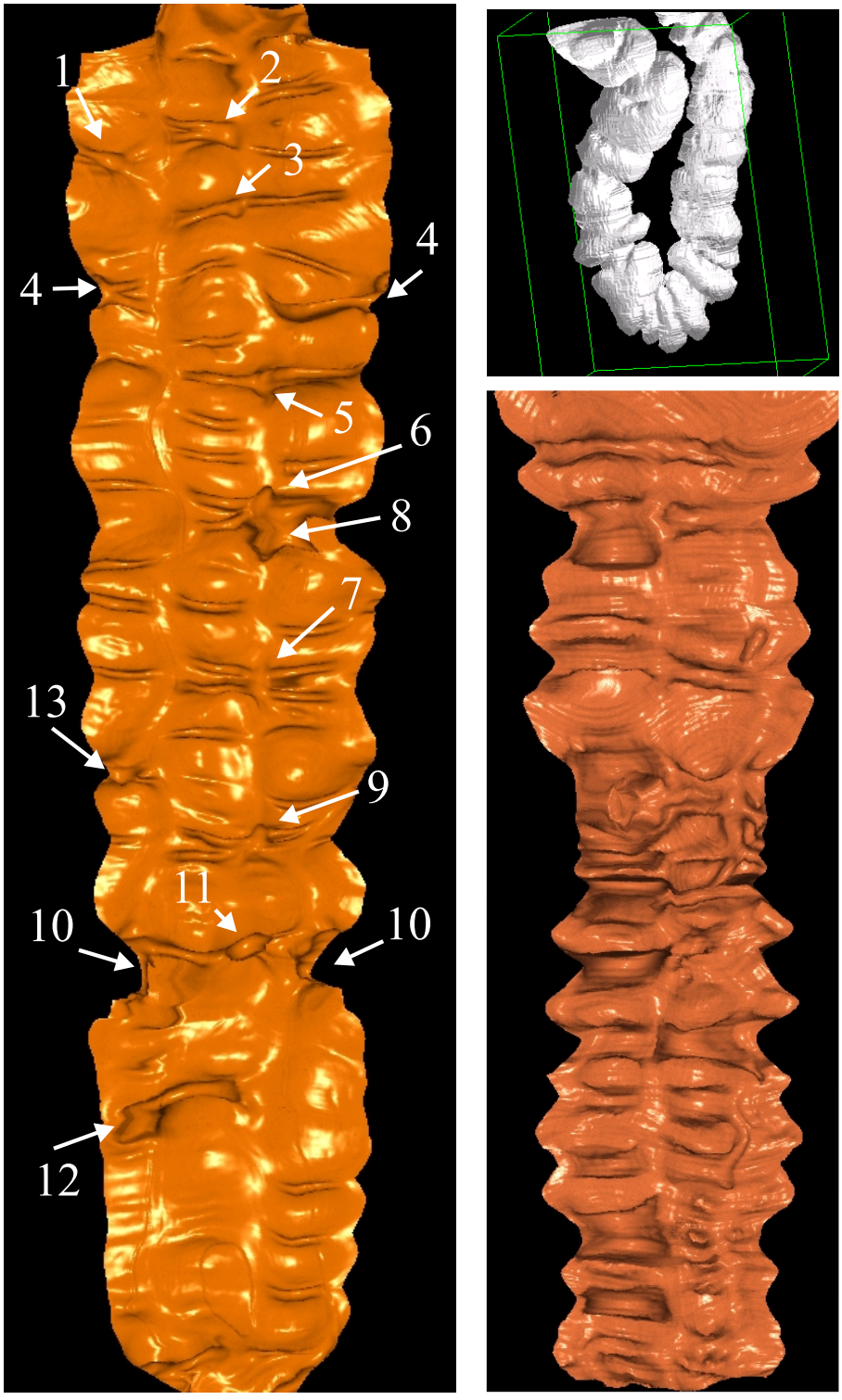
In this section, we describe the results of applying virtual colon unfolding to several data sets.

One data set is the CT volume data of an extracted colon with a resolution of 381x120x632 presented in chapter 8 (see figure 8.6a). The colon is 50 cm long and contains 13 polyps. The results of the unfolding procedure for this colon can be seen in figure 9.8a. All the polyps could be easily detected by inspection. The extracted colon was physically dissected and several pictures of the dissected colon were also taken. These pictures enable a qualitative comparison between the real data and the results of the presented algorithm (see figure 9.9).

A second data set is a segment of the extracted colon with a resolution of 190x120x150. This data set has been used throughout the chapter to illustrate the algorithm. The comparison of the virtually unfolded colon with the picture of the dissected colon can be seen in figure 9.9a.

The third data set is from a CT data set of a healthy colon segment with a resolution of 198x115x300. The unfolding of the data set and an outside view can be seen in figure 9.8b. This colon is healthy, so no polyps could be found after the unfolding. For videos and further images please refer to:
`www.cg.tuwien.ac.at/research/vis/vismed/ColonUnfolding/`.

Table 9.1 shows the calculation times of the presented method on a Pentium II (400MHz). The timings correspond to preprocessing time. Once the method has been executed the physician can inspect the resulting overview image interactively. The table is divided into three parts. The first part presents the timings obtained from generating the nonlinear rays and doing direct volume rendering. The timings depend on the resolution of the sampling, i.e., the number of samples taken in the u and v direction, and the data set resolution.



a) b)

Figure 9.8: **a)** Virtually unfolding of the extracted colon with the polyps numbered according to the real dissection. **b)** Outside view and virtual unfolding of a segment of a CT data set of a healthy colon with a resolution of 198x115x300.

Data Set	Nonlinear Ray Casting		Nonlinear 2D Scaling				Resampling	
	Grid	time(min)	C_r	iterations	σ	sec./iter.	h	time(min)
Extracted Colon	128x890	21.65	0.3	352	0.2797	0.797	0.5	83.80
			0.5	601	0.3165		1.0	26.50
Segment of the Extracted Colon	126x171	3.4	0.3	960	0.2808	0.141	0.5	15.46
			0.5	1687	0.3511		1.0	5.32
Healthy Colon	200x1178	13.2	0.3	70	0.2799	1.656	0.5	59.18
			0.5	58	0.2797		1.0	6.75

Table 9.1: Calculation times depending on the different parameters of the algorithm. The times are calculated using a Pentium II CPU (400MHz).

The middle part of the table shows the timings of the nonlinear 2D scaling algorithm. The convergence speed of the algorithm depends basically on the grid resolution, the value of C_r and σ (i.e., the precision achieved by the unfolding). In the first two data sets the increase of C_r produces a reduction of the convergence rate while for the healthy colon it improves. The value of C_r has to be tuned for each data set.

The last column in table 9.1 gives timings concerning the resampling process. The times depend basically on the sampling step of the grid and the difference between the original grid and the resampled grid (i.e., number of resampling points). We present the times with a resampling step h of 0.5 and 1.0 times the size of a voxel in 3D space.

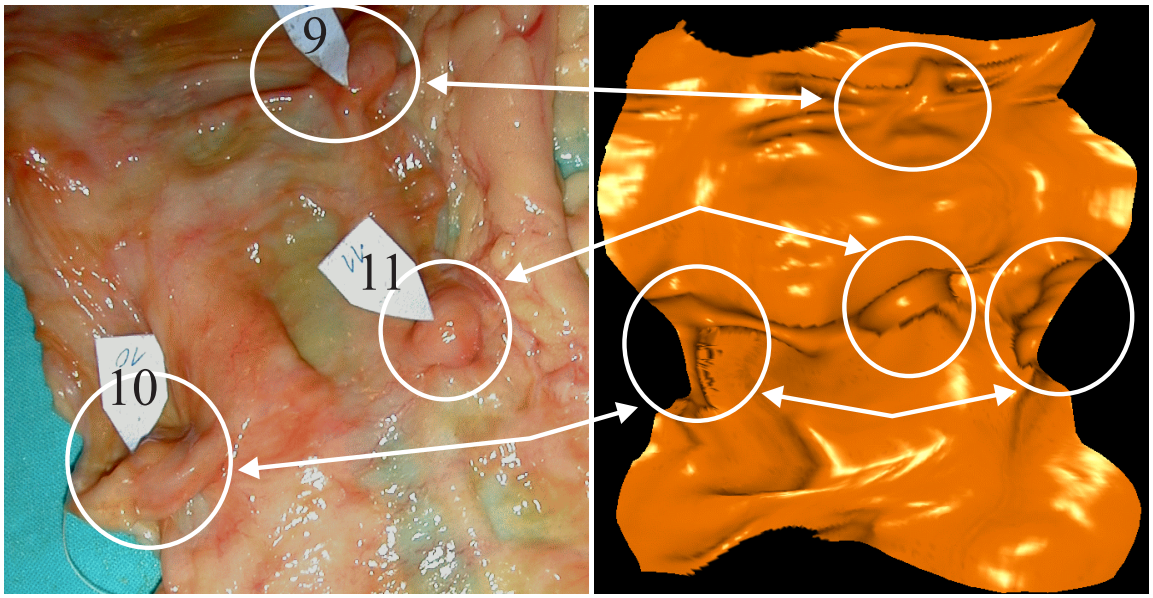
The time of the entire process depends on the data set and the precision of the result. In total, the time of the whole process, including the distance map calculation, is in the range of hours. This is, however, preprocessing time. Once the computation has been done, the resulting 2D grid can be inspected interactively by the physician. The result can be seen like an overview map. If suspicious areas are identified they can be inspected more carefully using other tools like cross-sections of the original data set or conventional rendering of the region in question.

Experiments have shown that the algorithm is quite sensitive to the smoothness of path $\mathbf{c}(v)$. For a good result, it is important that the path is smooth and has as many linear segments as possible.

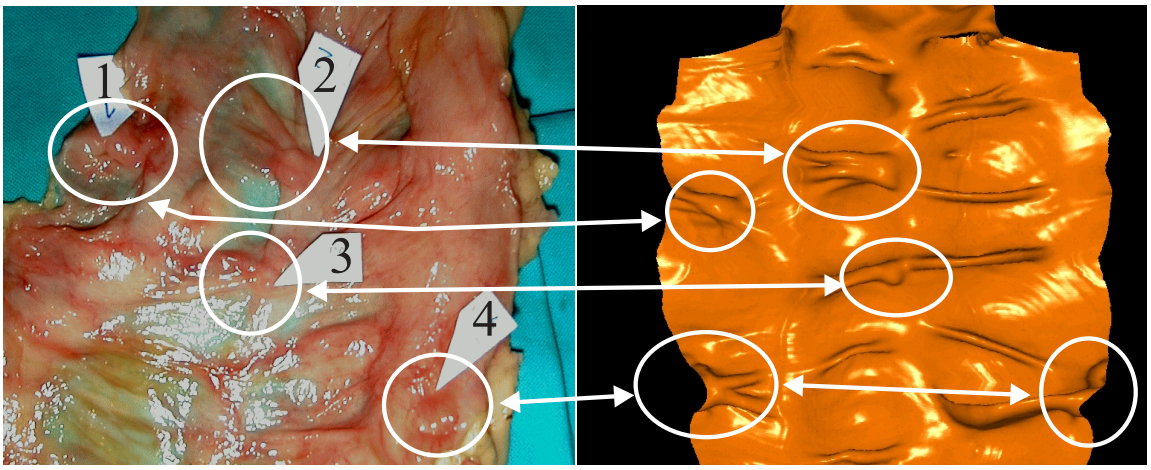
The nonlinear ray casting produces a problem that occurs when a ridge line of distance map $dist(\mathbf{p})$ is inside the colon. Then, there will be rays that never reach the surface of the colon. Although, this case never appeared in the tested data sets, a solution to this situation has to be studied. A possible solution would be to combine the central path distance map with the colon surface distance map.

The presented algorithm, as well as the algorithm presented in chapter 8, takes only into account the areas that are directly projected to the central path. Areas which are behind mushroom-like folds are not visualized. Using more than one surface point per ray, could solve the problem of occluded areas.

The current implementation of the algorithm has not been optimized so there is room for temporal improvement. Although the virtual unfolding is a preprocessing step, there is a need to accelerate it to facilitate parameter tuning.



a)



b)

Figure 9.9: Qualitative comparison of the virtually unfolded colon with pictures taken from the real dissection. In **a)**, the images correspond to the segment of the extracted colon data set which was used throughout this chapter as example. The orientation in which the pictures were taken does not correspond to the orientation of the virtually unfolded colon.

Once the physicians have detected a polyp, they are interested in going back to the original volume data and inspect it using other visualization techniques like perspective volume rendering or multiplanar reconstruction. An application allowing such an inspection similar to the one presented in chapter 8 must be developed using the relation between the points in the 2D unfolded map and 3D volume data.

The nonlinear 2D scaling step unfolds the colon with area preservation. This allows the physicians to estimate the sizes of the polyps. However, the shape is not preserved as the angles are not preserved.

Methods for preservation of the angles or a combination of angle and area preservation will be subject of future study.

The presented method has shown promising results, and it should be tested with more data sets of real pathological cases. The algorithm is not restricted to the colon. It has the potential of being used for other tubular organ as well.

Chapter 10

Summary and Conclusions

> Qui busca la veritat es mereix el càstig de trobar-la.
> Who looks for the truth, deserves the punishment of finding it.
> *Santiago Rusiñol i Prats (1861-1931)*

We developed and investigated different visualization techniques for virtual endoscopy. Virtual endoscopy has the potential to be used to substitute some real endoscopy procedures or to reduce their drawbacks. This thesis has been divided into two parts.

In the first part, we concentrated on techniques to improve virtual endoscopy techniques which simulate the behavior of a real endoscope. In chapter 3, we presented a general framework for virtual endoscopy systems. A prototype based on this framework was implemented. In this prototype, the camera motion follows the principles of guided navigation with free rotation movements. The position of the camera is restricted to the calculated optimal path. The thinning algorithm chosen to calculate the optimal path calculation was explained in chapter 4.

We focus on achieving high quality renderings, since we concentrate on developing a prototype to investigate visualization techniques for diagnosis. Therefore, direct volume rendering by ray casting has been used. The main drawback of direct volume rendering is that it has a high computational cost.

A new space leaping acceleration technique for ray casting was presented in chapter 5. The algorithm generates a cylindrical approximation of the cavity of tubular-like organs. This approximation is used as bounded cylinders to accelerate direct volume rendering for virtual endoscopy. An early termination criterion for virtual endoscopy is also presented. The achieved acceleration depends on the data set. On a common PC (i.e., Pentium II 400MHz), the presented algorithm is between three and four times faster than a common ray-casting algorithm. However, it does not allow direct real-time volume-rendering.

A new technique to obtain perspective high-quality volume-rendering using hardware acceleration (i.e., VolumePro) was presented in chapter 6. This approach produces perspective projection views using parallel projection techniques (i.e., projected-slabs algorithm). The algorithm uses consecutive parallelly projected slabs of the volume. The error produced due to the approximation of perspective projection is investigated. Besides, based on error studies, we introduce a criterion to vary the slab thickness and therefore to improve the performance of the algorithm.

Simulating an endoscopic view is not the most suitable visualization technique in many endoscopy procedures. The second part of the thesis presents techniques which virtually unfold the colon to inspect its surface and detect polyps. We concentrated on the colon although they could be used with other organs too.

Chapter 8 described a technique that locally unfolds the colon, and generates an animation sequence from consecutive unfolded regions. The images are generated with a projection technique that allows the physician to visualize most of the surface, and to easily recognize polyps that in an endoscopic view would be hidden by folds or would be hard to localize. The presented method avoids double counting of polyps. We presented two sampling strategies that respectively preserve the angle or area of the projected surface elements. We maximized the coherence between frames by minimizing camera rotation. The images are also enhanced by calculating level lines which represent the depth. Finally, we presented a technique to generate an endoscopic-view navigation in real-time by using the data of the video of the unfolded colon. An application has been developed which allows the physicians to inspect the original slices once a polyp has been detected.

The drawback of the previous method is that the physician has to inspect a video to be able to visualize the whole surface. In chapter 9, the goal is to enable the physician to inspect and get as much information as possible of the inner organ surface at a first glance. The problematic areas can be identified quickly and inspected later in more detail. This approach solves the problem of double appearance of polyps using nonlinear ray casting. Compensation of the distortions due to the unfolding of the colon is achieved using an iterative method called nonlinear 2D scaling which is similar to the nonlinear magnification fields used in information visualization. Finally, a method is presented to resample in areas where features can be missed due to undersampling.

The methods presented in chapters 8 and 9 have been tested with several data sets. One of them enabled a qualitative comparison of the resulting images with images of the corresponding real extracted colon.

In each of the corresponding chapters, the conclusions and future work for each of the methods have already been presented. From the work and the acquired experience during the development of this thesis some more general conclusions can be pointed out.

- To achieve meaningful visualizations, it is important to adapt the visualization method to the desired application. It is nearly impossible to develop a system which can be used in any of the manifold applications of virtual endoscopy. For example, there are applications like training, where the quality of the images is not as important as achieving real time frame rates. On the other hand, diagnosis procedures need high quality images. This quality should not be reduced to gain speed. Therefore, completely different requirements and methods must be used in each of the cases which makes a generic solution difficult.
- One of the main drawbacks of direct volume rendering is the definition of a correct transfer functions. Correct means that the desired objects to be visualized are actually visible. This is still an open problem, and the definition of transfer functions is a field of research by itself.
- The use of virtual objects allows visualization techniques to have much more freedom than restrict themselves to imitate well-known medical procedures. The second part of this thesis shows that not following the common approaches can lead to meaningful visualization techniques like virtual colon unfolding.

- For diagnosis applications, it is important that physicians can rely on the results that are obtained. Just one result image is not enough. Different visualization techniques for the same data and region linked together are necessary (e.g., use the original slices or multi-planar reconstruction planes). This gives the physician the possibility to verify the results.
- Medical visualization is an interdisciplinary field. Computer scientists know the techniques that can be used to visualize the data, while physicians know what should be seen. It is important that collaboration between both fields is achieved to obtain meaningful results. One of the most difficult tasks is to achieve a good communication channel between both parties.
- The new methods presented in this thesis are not yet able to be used in normal clinical routine. There is a still a long way to go. They need to be evaluated with a much larger number of data sets and reach a state of maturity, such that clinical staff is able to use them. We are working on that and there is good hope that some day these methods will get to such a status.

Bibliography

[Abra92] R. H. Abraham and C. D. Shaw. *Dynamics: The Geometry of Behavior*. Addison-Wesley, 1992. Cited on page 76.

[Adel94] S. Adelson and C. Hansen. Fast Stereoscopic Images with Ray-Traced Volume Rendering. In A. Kaufman and W. Krueger, editors, *1994 Symposium on Volume Visualization, Conference Proceedings*, pages 3–10, Washington, DC., October 1994. ACM SIGGRAPH. Cited on page 17.

[Artz81] E. Artzy, G. Frieder, and G.T. Herman. The Theory, Design, Implementation and Evaluationn of a Three-Dimensional Surface Detection Algorithm. *Computer Graphics and Image Processing*, 15:1–24, 1981. Cited on page 9.

[Bart99] D. Bartz and M. Skalej. VIVENDI-A Virtual Ventricle Endoscopy System for Virtual Medicine. In *VisSym'99 Joint Eurographics- IEEE TCVG Syposium on Visualization, Conference Proceedings*, pages 155–166, 1999. Cited on page 26.

[Benn91] C. Bennis, J.M. Vézien, and G. Iglésias. Piecewise Surface Flattening for Non-Distorted Texture Mapping. In *SIGGRAPH'91, Conference Proceedings*, pages 237–246, 1991. Cited on page 61.

[Bitt00] I. Bitter, M. Sato, M. Bender, K.T. McDonnell, A. Kaufman, and M. Wan. CEASAR: A Smooth, Accurate and Robust Centerline Extraction Algorithm. In *IEEE Visualization 2000, Conference Proceedings*, pages 45–52, 2000. Cited on page 31.

[Blin82] J. Blinn. Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. *Computer Graphics*, 16(3):21–29, 1982. Cited on page 11.

[Blin88] J. Blinn. Where am I? What am I looking at? *IEEE Computer Graphics and Applications*, 8(4):76–81, July 1988. Cited on page 24.

[Bloo88] J. Bloomenthal. Polygonization of Implicit Surfaces. *Computer Aided Geometric Design*, 5(4):341–345, 1988. Cited on page 9.

[Brad97] M. L. Brady, K. Jung, H. T. Nguyen, and T. Nguyen. Two-Phase Perspective Ray Casting for Interactive Volume Navigation. In *IEEE Visualization'97, Conference Proceedings*, pages 183–189, October 1997. Cited on pages 17 and 26.

[Brad98] M. L. Brady, K. K. Jung, H. T. Nguyen, and T. Nguyen. Interactive Volume Navigation. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):243–255, July – September 1998. Cited on pages 17, 26, and 53.

[Cabr94] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *1994 Symposium on Volume Visualization, Conference Proceedings*, pages 91–98. ACM SIGGRAPH, October 1994. Cited on pages 18, 26, and 46.

[Cast98] S. Castro, A. König, H. Löffelmann, and E. Gröller. Transfer Function Specification for the Visualization of Medical Data. Technical Report TR-186-2-98-12, Institute of Computer Graphics, Vienna University of Technology, 1998. http://www.cg.tuwien.ac.at/research/TR/. Cited on pages 12 and 29.

[Cséb01] B. Csébfalvi. *Interactive Volume-Rendering Techniques for Medical Data Visualization*. PhD thesis, Vienna University of Tehnology. Institute of Computer Graphics and Algorithms, 2001. Cited on page 16.

[Cull93] T.J. Cullip and U. Neumann. Accelerating Volume Reconstruction With 3D Texture Hardware. Technical Report TR93-027, Department of Computer Science at the University of North Carolina, Chapel Hill, 1993. Cited on page 46.

[Dach00] F. Dachille, K. Mueller, and A. Kaufman. Volumetric Backprojection. In *2000 Symposium on Volume Visualization and Graphics, Conference Proceedings*, pages 109–117, 2000. Cited on page 12.

[Dans92] J. Danskin and P. Hanrahan. Fast Algorithms for Volume Ray Tracing. In *1992 Workshop on Volume Visualization, Conference Proceedings*, pages 91–98, New York, October19–20 1992. ACM Press. Cited on page 16.

[Dara97] K. Darabi, K. D. M. Resch, J. Weinert, and U. Jendrysiak. Real and simulated endoscopy of neurosurgical approaches in an anatomical model. *Lecture Notes in Computer Science*, 1205:323–326, 1997. Cited on page 26.

[Druc92] S. M. Drucker, T. A. Galyean, and D. Zeltzer. CINEMA: A System for Procedural Camera Movements. In Marc Levoy and Edwin E. Catmull, editors, *ACM 1992 Symposium on Interactive 3D Graphics, Conference Proceedings*, pages 67–70, Cambridge, MA, March–April 1992. ACM Press. Cited on page 24.

[Dune90] S. Dune, S. Napel, and B. Rutt. Fast Reprojection of Volume Data. In *The First Conference on Visualization in Biomedical Computing, Conference Proceedings*, pages 11–18, 1990. Cited on page 15.

[Floa97] M.S. Floater. Parametrization and Smooth Approximation of Surface Triangulation. *Computer Aided Geometric Design*, 14:231–250, 1997. Cited on page 61.

[Fuch77] H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal Surface Reconstruction from Planar Contours. *Communications of the ACM*, 20(10):693–702, 1977. Cited on page 8.

[Gagv97] N. Gagvani. Skeletons and Volume Thinning In Visualization. Master's thesis, New Brunswick, Rutgers, The State University of New Jersey, 1997. Cited on pages 30 and 31.

[Galy95] T. A. Galyean. Guided Navigation of Virtual Enviroments. In *ACM 1995 Symposium on Interactive 3D Graphics, Conference Proceedings*, pages 103–104. ACM SIGGRAPH, April 1995. Cited on page 24.

[Geig95] B. Geiger and R. Kikinis. Simulation of Endoscopy. In Nicholas Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine, Conference Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, April 1995. Cited on page 26.

[Geld96] A. van Gelder and K. Kwansik. Direct Volume Rendering with Shading Via Three-Dimentional Textures. In *1996 Symposium on Volume Visualization, Conference Proceedings*, pages 23–30, 1996. Cited on page 18.

[Gibs98] S. F. F. Gibson. Using Distance Maps for Accurate Surface Representation in Sampled Volumes. In *1998 Symposium on Volume Visualization, Conference Proceedings*, pages 23–30, 1998. Cited on page 11.

[Gröl92] E. Gröller. *Coherence in Computer Graphics*. PhD thesis, Vienna University of Technology. Institute of Computer Graphics and Algorithms, 1992. Cited on page 16.

[Gröl95] E. Gröller. Nonlinear Ray Tracing: Visualizing Strange Worlds. *The Visual Computer*, 11:263–274, 1995. Cited on page 75.

[Gudm90] B. Gudmundsson and M. Randen. Incremental generation of projections of CT-volumes. In *The First Conference on Visualization in Biomedical Computing, Conference Proceedings*, Atlanta, 1990. Cited on page 17.

[Guen94] T. Guenther, C. Poliwoda, C. Reinhard, J. Hesser, R. Maenner, H.-P. Meinzer, and H.-J. Baur. VIRIM: A Massively Parallel Processor for Real-Time Volume Visualization in Medicine. In *EUROGRAPHICS Workshop on Graphics Hardware'94, Conference Proceedings*, pages 103–108, 1994. Cited on page 18.

[Hake00a] S. Haker, S. Angenent, A. Tannenbaum, and R. Kikinis. Nondistorting Flattening Maps and the 3D Visualization of Colon CT Images. *IEEE Transactions on Biomedical Engineering*, 19(7):665–671, July 2000. Cited on pages 61 and 74.

[Hake00b] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal Surface Parameterization for Texture Mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, April-June 2000. Cited on page 61.

[Hand97] C. Hand. Survey of 3-D Interaction Techniques. *Computer Graphics Forum*, 16(5):269–281, December 1997. Cited on pages 21, 24, and 25.

[He96] T. He and A. Kaufman. Fast Stereo Volume Rendering. In *IEEE Visualization'96, Conference Proceedings*, pages 49–56. IEEE, October 1996. Cited on page 17.

[Hiet00] R. Hietala and J. Oikarinen. A Visibility Determination Algorithm for Interactive Virtual Endoscopy. In *IEEE Visualization 2000, Conference Proceedings*, pages 29–36, 2000. Cited on page 27.

[Hinc94] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. A Survey of Design Issues in Spatial Input. In *ACM 1994 Symposium on User Interface Software and Technology, Conference Proceedings*, Two Hands and Three Dimensions, pages 213–222, 1994. Cited on page 24.

[Hong97] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual Voyage: Interactive Navigation in the Human Colon. In *SIGGRAPH'97, Conference Proceedings*, Annual Conference series, pages 27–34, 1997. Cited on page 25.

[Jefr98] R. B. Jefrey. Three-Dimensional Imaging and Virtual Reality Application of Spiral CT. In G.P. Krestin and G.M. Glazer, editors, *Advances in CT IV*, volume 4, Conference Book 18, pages 149–153. Springer, March 1998. Cited on page 22.

[Kaji84] J. Kajiya and B. von Herzen. Ray Tracing Volume Densities. *Computer Graphics*, 18(3):165–174, 1984. Cited on page 11.

[Kass87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. In *First International Conference on Computer Vision, Conference Proceedings*, pages 259–268. IEEE Computer Society Press, 1987. Cited on page 22.

[Keah97] T.A. Keahey and E.L. Robertson. Nonlinear Magnification Fields. In *IEEE Information Visualization'97, Conference Proceedings*, pages 51–58, 1997. Cited on pages 76 and 81.

[Kepp75] E. Keppel. Approximating Complex Surfaces by Triangulation of Contour Lines. *IBM Journal of Research and Development*, 19(1):2–11, 1975. Cited on page 8.

[Kind98] G. Kindlmann and J. W. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In *1998 Symposium on Volume Visualization, Conference Proceedings*, pages 79–86, 1998. Cited on page 13.

[Klok86] F. Klok. Two Moving Coordinate Frames for Sweeping Along a 3D Trajectory. *Computer Aided Geometry Design*, 3:217–229, 1986. Cited on page 67.

[Knit97] G. Knittel and W. Straßer. VIZARD: Visualization Accelerator for Realtime Display. In *SIGGRAPH /EUROGRAPHICS Workshop on Graphics Hardware, Conference Proceedings*, pages 139–147, 1997. Cited on page 18.

[Kong93] T.Y. Kong. One problem of determining whether a parallel reduction operator for n-dimensional binary images always preserves topology. In *SPIE, Conference Proceedings*, pages 69–77, 1993. Cited on page 31.

[Köni01] A. König and E. Gröller. Mastering Transfer Function Specification by Using VolumePro Technology. In *SCCG'01 Spring Conference on Computer Graphics, Conference Proceedings*, pages 279–286, 2001. Cited on page 12.

[Kree98] K. Kreeger, I. Bitter, F. Dachille, B. Chen, and A. Kaufman. Adaptive Perspective Ray Casting. In *1998 Symposium on Volume Visualization, Conference Proceedings*, pages 19–20, 1998. Cited on page 53.

[Lacr94] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear–Warp Factorization of the Viewing Transformation. In *SIGGRAPH'94, Conference Proceedings*, pages 451–458. ACM SIGGRAPH, ACM Press, July 1994. Cited on pages 15, 46, and 47.

[Laur91] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics*, 25(4):285–288, July 1991. Cited on page 16.

[Levo88] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(5):29–37, May 1988. Cited on page 14.

[Levo90a] M. Levoy. Efficient Ray Tracing of Volume Data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990. Cited on page 16.

[Levo90b] M. Levoy. Volume Rendering by Adaptive Refinement. *The Visual Computer*, 6(1):2–7, February 1990. Cited on page 16.

[Lévy98] B. Lévy and J.L. Mallet. Non-Distorted Texture Mapping For Sheared Triangulated Meshes. In *SIGGRAPH'98, Conference Proceedings*, pages 343–352, 1998. Cited on page 61.

[Lohm98] G. Lohmann. *Volumetric Image Analysis*. Chichester Wiley, 1998. Cited on pages 8, 16, 23, 30, 31, 35, 75, and 76.

[Lora01] T. Lorang. *Organisation and Visualisation of Medical Images in Radiotherapy*. PhD thesis, Vienna University of Technology. Faculty of Technical Sciences. Institute of Medical Computer Sciences., 2001. Cited on page 17.

[Lore87] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In Maureen C. Stone, editor, *SIGGRAPH'87, Conference Proceedings*, pages 163–169. ACM SIGGRAPH, Addison Wesley, July 1987. Cited on pages 9 and 28.

[Lore96] B. Lorensen, R. Kikinis, J. Flynn, A. Kaufman, and S. Napel. Surface Rendering versus Volume Rendering in Medical Imaging: Techniques and Applications (Panel). In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization'96, Conference Proceedings*, pages 439–440, October 27–November 1 1996. Cited on page 22.

[Lueb95] D. Luebke and C. Georges. Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics, Conference Proceedings*, pages 105–106. ACM SIGGRAPH, April 1995. Cited on page 25.

[Ma96] C. M. Ma and M. Sonka. A Fully Parallel 3D Thinning Algorithm and its Application. In *Computer Vision Graphics and Imaging Processing: Image Understanding, Conference Proceedings*, volume 64, pages 420–433, 1996. Cited on pages vi, 31, 32, 33, and 34.

[Mack90] J. D. Mackinlay, S. K. Card, and G. G. Robertson. Rapid Controlled Movement Through a Virtual 3D Workspace. In Forest Baskett, editor, *SIGGRAPH'90, Conference Proceedings*, Annual Conference Series, pages 171–176. ACM SIGGRAPH, Addison Wesley, August 1990. Cited on page 24.

[Malz93] T. Malzbender. Fourier Volume Rendering. *ACM Transactions on Graphics*, 12(3):233–250, 1993. Cited on page 15.

[Mars94] S. R. Marschner and R. J. Lobb. An Evaluation of Reconstruction Filters for Volume Rendering. In R. Daniel Bergeron and Arie E. Kaufman, editors, *IEEE Visualization'94, Conference Proceedings*, pages 100–107, Los Alamitos, CA, USA, October 1994. IEEE Computer Society Press. Cited on page 11.

[Max95] N. Max. Optical Models for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. Cited on pages 11, 12, and 29.

[Mitc88] D. P. Mitchell and A. N. Netravali. Reconstruction Filters in Computer Graphics. *Computer Graphics*, 22(4):221–228, August 1988. Cited on page 11.

[Möll97a] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A Comparison of Normal Estimation Schemes. In *IEEE Visualization'97, Conference Proceedings*, pages 19–26, 1997. Cited on page 11.

[Möll97b] T. Möller, R. Machiraju, K. Müller, and R. Yagel. Evaluation and Design of Filters Using a Taylor Series Expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, 1997. Cited on page 11.

[Mori96] K. Mori, J. Hasegawa, J. Toriwaki, H. Anno, and K. Katada. A Fast Rendering Method Using the Tree Structure of Objects in Virtualized Bronchus Endoscope System. *Lecture Notes in Computer Science*, 1131:33–42, 1996. Cited on page 26.

[Mort95] E. N. Mortensen and W. A. Barrett. Intelligent Scissors for Image Composition. In Robert Cook, editor, *SIGGRAPH'95, Conference Proceedings*, Annual Conference Series, pages 191–198. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995. Cited on page 22.

[Mroz01] L. Mroz. *Real-Time Volume Visualization on Low-End Hardware*. PhD thesis, Vienna University of Technology. Institute of Computer Graphics and Algorithms, 2001. Cited on page 16.

[Müll99] K. Müller, T. Möller, and R. Crawfis. Splatting Without the Blur. In *IEEE Visualization'99, Conference Proceedings*, pages 363–371, 1999. Cited on page 15.

[Neum00] L. Neumann, B. Csébfalvi, A. König, and E. Gröller:. Gradient Estimation in Volume Data Using 4D Linear Regression. In *EUROGRAPHICS 2000, Conference Proceedings*, pages 351–357, 2000. Cited on page 11.

[Niel91] G. M. Nielson and B. Hamann. The Asymptotic Decider: Resolving Ambiguity in Marching Cubes. In *IEEE Visualization'91, Conference Proceedings*, pages 83–91, 1991. Cited on page 9.

[Ning93] P. Ning and J. Bloomenthal. An Evaluation of Implicit Surface Tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993. Cited on page 9.

[Paik98] D. S. Paik, C. F. Beaulieu, R. B. Jeffrey, G. D. Rubin, and S. Napel. Automated Path Planning for Virtual Endoscopy. *Medical Physics*, 25(5):629–637, May 1998. Cited on page 26.

[Paik00] D.S. Paik, C.F. Beaulieu, R. B. Jeffrey, Jr. C.A. Karadi, and S. Napel. Visualization Modes for CT Colonography Using Cylindrical and Planar Map Projections. *Journal of Computer Tomography*, 24(2):179–188, 2000. Cited on page 61.

[Palu97] M. Paluszny and K. Buehler. Canal Surfaces and Inversive Geometry. In *Mathematical Methods for Curves and Surfaces II, Conference Proceedings*, pages 367–374. Vanderbilt Univ. Press, 1997. Cited on page 39.

[Pfis96] H. Pfister and A. Kaufman. Cube-4 - A Scalable Architecture for Real-Time Volume Rendering. In *1996 Symposium on Volume Visualization, Conference Proceedings*, pages 47–54. IEEE, 1996. Cited on page 18.

[Pfis99] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro Real-Time Ray-Casting System. In *SIGGRAPH'99, Conference Proceedings*, pages 251–260, 1999. Cited on pages 18 and 46.

[Pfis01] H. Pfister, B. Lorensen, C. Baja, G. Kinklmann, W. Schroeder, L. Sobierajski Avila, K. Martin, R. Machiraju, and J. Lee. Visualization Viewpoints: The Transfer Function Bake-Off. *IEEE Computer Graphics and Applications*, 21(3):16–23, 2001. Cited on page 12.

[Puig97] A. Puig, D. Tost, and I. Navazo. An Interactive Cerebral Blood Vessel Exploration System. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97, Conference Proceedings*, pages 443–446, 1997. Cited on page 37.

[Resz00] C. Reszk-Salama, K. Engel, T. Bauer, and T. Ertl. Interactive Volume Rendering on Standard Pc Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *EUROGRAPHICS Hardware Workshop 2000, Conference Proceedings*, pages 47–54, 2000. Cited on page 18.

[Rose94] L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann. *Scientific Visualization Advances and Challenges*. IEEE Computer Society Press, 1994. Cited on page 1.

[Sait90] T. Saito and T. Takahashi. Comprehensible Rendering of 3-D Shapes. In Forest Baskett, editor, *SIGGRAPH'90, Conference Proceedings*, Annual Conference Series, pages 197–206, August 1990. Cited on page 67.

[Same86] M. Samek, C. Slean, and H. Weghorst. Texture Mapping and Distortion in Digital Graphics. *The Visual Computer*, 2:313–320, 1986. Cited on page 61.

[Sata97] R.M. Satava and R.A. Robb. Virtual Endoscopy: Application of 3D Visualization to Medical Diagnosis. *Presence*, 6(2):179–197, April 1997. Cited on page 26.

[Shah96] R. Shahidi, V. Argiro, S. Napel, L. Gray, H P McAdams, G.D. Rubin, C F Beaulieu, R.B. Jeffrey, and A. Johnson. Assessment of Several Virtual Endoscopy Techniques Using Computed Tomography and Perspective Volume Rendering. *Lecture Notes in Computer Science*, 1131:521–526, 1996. Cited on page 26.

[Sobi94] L. M. Sobierajski. *Global Illumination Models for Volume Rendering*. PhD thesis, Stony Brook, NY, 1994. Cited on page 12.

[Sobi95] L. M. Sobierajski and R. S. Avila. A Hardware Acceleration Method for Volumetric Ray Tracing. In *IEEE Visualization'95, Conference Proceedings*, pages 27–34. IEEE, 1995. Cited on pages 17, 37, and 45.

[Sora01] E. Sorantin, E. Balogh, A. Vilanova Bartrolí, K. Palágyi, L. G. Nyúl, S. Loncaric, M. Subasic, and D. Kovaćević. *MEDICAL RADIOLOGY - Diagnostic Imaging*, chapter Technique of Virtual Dissection of the Colon based on Spiral CT data. Springer Verlag, 2001. Cited on pages 60 and XI.

[Sram96] M. Sramek. *Visualization of Volumetric Data by Ray Tracing*. PhD thesis, Vienna University of Tehnology. Institute of Computer Graphics and Algorithms, 1996. Cited on page 37.

[Tell91] S. J. Teller and C. H. Séquin. Visibility preprocessing for interactive walkthroughs. In Thomas W. Sederberg, editor, *SIGGRAPH'91, Conference Proceedings*, volume 25, pages 61–69, July 1991. Cited on page 25.

[Theu00] T. Theußl, H. Hauser, and E. Gröller. Mastering Windows: Improving Reconstruction. In *2000 Symposium on Volume Visualization and Graphics, Conference Proceedings*, pages 101–108, 2000. Cited on page 11.

[Tied98] U. Tiede, T. Schiemann, and K. Höhne. High quality rendering of attributed volume data. In *IEEE Visualization'98, Conference Proceedings*, pages 255–262, 1998. Cited on page 11.

[Tots93] T. Totsuka and M. Levoy. Frequency Domain Volume Rendering. In *SIGGRAPH'93, Conference Proceedings*, pages 271–278, 1993. Cited on page 15.

[Udup83] J.K. Udupa. Display of 3D Information in Discrete 3D Scenes Produced by Computerized Tomography. In *IEEE*, volume 71, pages 420–431, 1983. Cited on page 9.

[Vesa43] A. Vesalius. *De Humani Corporis Fabrica*. BASILE AE, EX OFFICI, 1543. Cited on pages vi, 1, and 2.

[Vila99] A. Vilanova Bartrolí, A. König, and E. Gröller. VirEn: A Virtual Endoscopy System. *Machine GRAPHICS & VISION*, 8(3):469–487, 1999. Cited on pages 20, 31, and XI.

[Vila00] A. Vilanova Bartrolí, A. König, and E. Gröller. Cylindrical Approximation of Tubular Organs for Virtual Endoscopy. In *Computer Graphics and Imaging 2000, Conference Proceedings*, pages 283–289. IASTED/ACTA Press, November 2000. Cited on pages 36, 76, and XI.

[Vila01a] A. Vilanova Bartrolí, R. Wegenkittl, A. König, and E. Gröller. Nonlinear Virtual Colon Unfolding, October 2001. Vienna University of Technology. The Institute of Computer Graphics and Algorithms. TR-186-2-01-08. *(to be published in IEEE Visualization 2001)*. Cited on pages 74 and XI.

[Vila01b] A. Vilanova Bartrolí, R. Wegenkittl, A. König, and E. Gröller. Perspective Projection Through Parallelly Projected Slabs for Virtual Endoscopy. In *SCCG'01-Spring Conference on Computer Graphics, Conference Proceedings*, pages 287–295, April 2001. (Also published in IEEE Computer Society, pp. 241- 248, ISBN 0-7695-1215-1). Cited on pages 46 and XI.

[Vila01c] A. Vilanova Bartrolí, R. Wegenkittl, A. König, E. Gröller, and E. Sorantin. Virtual Colon Flattening. In *VisSym '01 Joint Eurographics - IEEE TCVG Symposium on Visualization, Conference Proceedings*, pages 127–136, May 2001. Cited on pages 63 and XI.

[Vis86] The Visible Human Project, 1986. http://www.nlm.nih.gov/research/visible/visible_human.html. Cited on page 7.

[Wan98] M. Wan, S. Bryson, and A. Kaufman. Boundary Cell-Based Acceleration for Volume Ray Casting. *Computers and Graphics*, 22(6):715–722, December 1998. Cited on page 37.

[Wan99a] M. Wan, A. Kaufman, and S. Bryson. High Performance Presence-Accelerated Ray Casting. In *IEEE Visualization'99, Conference Proceedings*, pages 379–386. IEEE, nov 1999. Cited on page 37.

[Wan99b] M. Wan, Q. Tang, A. Kaufman, Z. Liang, and M. Wax. Volume Rendering Based Interactive Navigation within the Human Colon. In *IEEE Visualization '99, Conference Proceedings*, pages 397–400, 1999. Cited on pages 26, 37, 43, and 45.

[Wan00] M. Wan, W. Li, K. Kreeger, I. Bitter, A. Kaufman, Z. Liang, D. Chen, and M. Wax. 3D Virtual Colonoscopy with Real-Time Volume Rendering. In *SPIE's International Symposium on Medical Imaging 2000, Conference Proceedings*, 2000. Cited on pages 46 and 53.

[Wang95] G. Wang and M.W. Vannier. GI Tract Unraveling by Spiral CT. In *SPIE, Conference Proceedings*, volume 2434, pages 307–315, 1995. Cited on page 60.

[Wang98] G. Wang, E. G. McFarland, B. P. Brown, and M. W. Vannier. GI Tract Unraveling with Curved Cross Sections. *IEEE Transactions on Medical Imaging*, 17:318–322, 1998. Cited on page 60.

[Wang99] G. Wang, S.B. Dave, B.P. Brown, Z. Zhang, E.G. McFarland, J.W. Haller, and M.W. Vannier. Colon Unraveling Based on Electrical Field: Recent Progress and Further Work. In *SPIE, Conference Proceedings*, volume 3660, pages 125–132, May 1999. Cited on pages 60 and 63.

[Wege00] R. Wegenkittl, A. Vilanova Bartrolí, B. Hegedüs, D. Wagner, M. C. Freund, and E. Gröller. Mastering Interactive Virtual Bronchioscopy on a Low-End PC. In *IEEE Visualization 2000, Conference Proceedings*, pages 461–464, October 2000. Cited on pages 26 and XI.

[West90] L. Westover. Footprint Evaluation for Volume Rendering. In Forest Baskett, editor, *SIGGRAPH'90, Conference Proceedings*, pages 367–376. ACM SIGGRAPH, Addison Wesley, August 1990. Cited on pages 15 and 26.

[West98] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *SIGGRAPH'98, Conference Proceedings*, pages 169–177. ACM SIGGRAPH, 1998. Cited on pages 18 and 46.

[Yage92a] R. Yagel, D. Cohen, and A. Kaufman. Discrete Ray Tracing. *IEEE Computer Graphics and Applications*, 12(5):19–28, 1992. Cited on page 11.

[Yage92b] R. Yagel and A. Kaufman. Template-based volume viewing. *Computer Graphics Forum*, 11(3):C153–C167, 1992. Cited on pages 15 and 17.

[Yage93] R. Yagel and Z. Shi. Accelerating Volume Animation by Space-Leaping. In Gregory M. Nielson and Dan Bergeron, editors, *IEEE Visualization '93, Conference Proceedings*, pages 62–69, San Jose, CA, October 1993. IEEE Computer Society Press. Cited on page 17.

[Yage96a] R. Yagel and W. Ray. Visibility Computation for Efficient Walkthrough of Complex Environments. *PRESENCE*, 5(1):1–16, 1996. Cited on page 25.

[Yage96b] R. Yagel, D. Stredney, G. J. Wiet, P. Schmalbrock, L. Rosenberg, D. J. Sessanna, and Y. Kurzion. Building a Virtual Environment for Endoscopic Sinus Surgery Simulation. *Computers & Graphics*, 20(6):813–823, December 1996. Cited on page 26.

[You97] S. You, L. Hong, M. Wan, K. Junyaprasert, A. Kaufman, S. Muraki, Y. Zhou, M. Wax, and Z. Liang. Interactive Volume Rendering for Virtual Colonoscopy. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97, Conference Proceedings*, pages 433–346. IEEE, nov 1997. Cited on pages 25 and 37.

[Zhou98] Y. Zhou, A. Kaufman, and A. W. Toga. Three-Dimensional Skeleton and Centerline Generation Based on an Approximate Maximum Distance Field. *Visual Computer*, 14:303–314, 1998. Cited on page 31.

[Zuck76] S. W. Zucker. Region Growing: Childhood and Adolescence. *Computer Graphics and Image Processing*, 5(3):382–399, September 1976. Cited on page 22.

[Zuid92] K.J. Zuiderveld, A.H.J. Koning, and M.A. Viergever. Accelaration of Ray-Casting Using 3D Distance Transform. In *Visualization in Biomedical Computing II, Conference Proceedings*, pages 324–335, 1992. Cited on pages 16 and 37.

Related publications

- A. Vilanova Bartrolí, A. König, and E. Gröller: **VirEn: Virtual Endoscopy System**. Published in Machine GRAPHICS & VISION, vol. 8(3) pp. 469-487, 1999 [Vila99].
- A. Vilanova Bartrolí, A. König, and E. Gröller: **Cylindrical Approximation of Tubular Organs for Virtual Endoscopy**. Published in Proceedings of Computer Graphics and Imaging 2000, IAESTED/ACTA Press, pp. 283-289, 2000 [Vila00].
- A. Vilanova Bartrolí, R. Wegenkittl, A. König, and E. Gröller: **Perspective Projection through parallelly projected slabs for virtual endoscopy**. Published in Proceedings SCCG 2001 - Spring Conference on Computer Graphics, pp. 287-295, 2001 (also published in IEEE Computer Society, ISBN 0-7695-1215-1) [Vila01b].
- A. Vilanova Bartrolí, R. Wegenkittl, A. König, E. Gröller and E. Sorantin: **Virtual Colon Flattening**. Published in VisSym'01 Joint Eurographics - IEEE TCVG Symposium on Visualization, pp. 127-136 2001 [Vila01c].
- A. Vilanova Bartrolí, R. Wegenkittl, A. König, and E. Gröller: **Nonlinear Virtual Colon Unfolding**. To be published in IEEE Visualization 2001 Technical Report TR-186-2-01-08. Institute of Computer Graphics and Algorithms. Vienna University of Technology [Vila01a].
- R. Wegenkittl, A. Vilanova Bartrolí, B. Hegedüs, D. Wagner, M.C. Freund, and E. Gröller: **Mastering Interactive Virtual Bronchioscopy on Low-End PC**. Published in Proceedings IEEE Visualization'00, pp. 461-464, 2000 [Wege00].
- E. Sorantin, E. Balogh, A. Vilanova Bartrolí, K. Palgyi, L. G. Nyl, S. Loncaric, M. Subasic, D. Kovaćević: **MEDICAL RADIOLOGY - Diagnostic Imaging**. Chapter, Technique of Virtual Dissection of the Colon based on Spiral CT data. Springer Verlag Press, 2001 [Sora01].