Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Ray-Based Image Generation For Advanced Medical Applications

Bernhard Kainz

Graz, Austria, April 2011

Thesis supervisors

Prof. Dr. Dieter Schmalstieg

Prof. Dr. Markus Hadwiger

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort

Datum

Unterschrift

To Pia and my parents

One can only display complex information in the mind. Like seeing, movement or flow or alteration of view is more important than the static picture, no matter how lovely.

Alan J. Perlis

Abstract

This thesis discusses real-time issues of ray-based image generation beyond known Graphics Processing Unit (GPU) algorithms in a medical context. Medical diagnostics highly depends on volumetric images methods. Image processing and representation of this data must become more accurate and comprehensible and image investigation must be as fast as possible; ultimately not only because of cost reasons. Therefore, the overall benefit of 3D image synthesis for medicine along with related research is discussed in the first part of this thesis.

During the last years the author and his colleagues have developed algorithms for 3D ray-based image synthesis which overcome the problems caused by the restrictions of shaders in a fixed function graphics pipeline. Ray-based volume rendering algorithms have besides their high computation complexity often an inherent algorithmic problem. Intersecting two or more datasets requires ray setup algorithms which do not scale efficiently enough with an increasing number of objects. Volumetric and hybrid volumetric-geometric visualization of medical multi-modal conditions is therefore sparsely covered by related work from computer science and not present in clinical practice without extensive preprocessing. To overcome these problems we have developed a completely flexible rendering engine, fully exploiting the possibilities of modern GPUs and discuss that system in the second part of this work. With different novel algorithms we show how to implement basically a software rendering engine, executed in the GPU. This approach is comparable to shader based approaches for simple scenes but much more efficient for highly complex scenes which may contain dozens editable and intersecting volumetric and translucent geometric objects. Further on we discuss options for applications with hard real-time constraints where pure algorithmic optimization is not sufficient anymore. Inter-operative applications must not hinder smooth operation of a surgeon with low frame rates and the resulting juddering. To overcome these problems most applications use either very basic visualization or sacrifice indiscriminately image quality for rendering speed. Therefore, we show a way to utilize the abilities of the human visual system to perceive incomplete images by introducing well known 3D object features as paradigm for ray importance sorting. As final contribution we show also how conventional ray-based approaches can be improved in terms of calculation accuracy and speed for special use-cases by incorporating automatic code differentiation, which is useful as input for special registration tasks.

The last part of this thesis shows how our novel algorithms can be applied to practical applications. Therefore we present multi-volume rendering for medical tumor accessibility planning, ray-casting kernel specialization for a real 2D-3D image registration problem during prostate surgery and automatic transfer-function generation for unskilled users. All parts of this thesis have been fully evaluated by experts from the medical domain and computer science. These parts show furthermore strong evidence, that our hypothesis in this thesis are not only valid but also directly applicable in many real-world medical rendering environments.

Kurzfassung

Diese Dissertation befasst sich mit Problemen der volumetrischen dreidimensionalen Bildsynthese jenseits bekannter Algorithmen in einem medizinischen Kontext. Volumetrische Bildsynthesealgorithmen werden üblicherweise direkt auf der Grafikkarte (GPU) ausgeführt und zeigen oft schwerwiegende Limitierungen wenn mehrere Volumen gleichzeitig zu verarbeiten sind. Mehrere Volumen treten häufig bei modernen bildgebenden Diagnoseverfahren auf, wenn mehrere Bildmodalitäten für denselben Patienten verwendet werden. Diese Daten müssen innerhalb einer Szene mit höchstmöglicher Genauigkeit und maximaler Interaktivität dargestellt werden. Da dies mit aktuellen Systemen nicht möglich ist wird häufig auf reine zweidimensionale Untersuchung der vorhandenen Volumsdaten zurückgegriffen. Daher untersucht die vorliegende Arbeit im ersten Teil mittels einer Umfrage unter mehr als zwei Dutzend Medizinern, ob dreidimensionale Bildsynthese in der klinischen Praxis überhaupt erforderlich ist. Da dies unter bestimmten Voraussetzungen positiv beantwortet werden kann haben der Autor dieser Arbeit und dessen Kollegen während der letzten Jahre Algorithmen entwickelt, die die Nachteile aktueller volumetrischer dreidimensionaler Bildsynthese größtenteils beseitigen. Diese Algorithmen basieren auf dem Prinzip, dass virtuelle Strahlen in einer dreidimensionalen Szene verfolgt werden. Durch die Ausnutzung moderner GPU Speicherarchitekturen ist es dem Autor gelungen diese Strahlenverfolgung so effizient zu gestalten, dass die Darstellung mehrerer beliebig verschnittener Volumen in Echtzeit möglich wurde. Da für manche Applikationen hohe Bilderzeugungsraten alleine nicht ausreichend sein können wird der oben genannte Ansatz durch Ausnutzung des menschlichen visuellen Systems erweitert, um nicht nur hohe, sondern auch Bilderzeugungsraten mit einer garantierten Frequenz zu ermöglichen. Solche garantierten Bilderzeugungsraten werden bei interoperativen Applikationen benötigt, bei denen unter keinen Umständen Bildruckeln ein kontinuierliches Arbeiten des durchführenden Arztes behindern darf. Im Gegensatz zu dem üblichen Verfahren, die Bilderzeugungsrate während der Szeneninteraktion zu Lasten der Qualität zu erhöhen, erhält dieser Ansatz die wichtigen Details der Szene und verhindert so eventuell lebensbedrohliche Fehlentscheidungen.

In einem weiteren Teil wird gezeigt, wie konventionelle volumetrische Bildsyntheseverfahren so weit verändert werden können, dass maximale Genauigkeit und Berechnungsgeschwindigkeit für die Registrierung verschiedener Datensätze resultiert. Die verwendeten automatischen analytischen Codeableitungsverfahren werden dafür im Detail erörtert und evaluiert.

Der letzte Teil dieser Arbeit befasst sich mit möglichen Anwendungsszenarien der vorgestellten Algorithmen. Zuerst wird ein System zur Visualisierung der sicheren Erreichbarkeit von Tumoren im menschlichen Körper vorgestellt. Dieses Problem kann durch volumetrische Methoden berechnet werden, konnte bisher aber durch die mangelnde Fähigkeit von volumetrischer Bildsynthese, mehrere Volumen gleichzeitig darzustellen, nicht zufriedenstellend dargestellt werden. Mit den Algorithmen aus dieser Arbeit wurde dieses Problem gelöst und es können vielversprechende Resultate mittels eines Prototypen gezeigt werden. Des weiteren zeigt dieser Abschnitt, wie automatische analytische Verfahren die fundamentalen Teile volumetrischer Bildsynthese so verändern können, dass sie erfolgreich auf ein Positionsbestimmungsproblem bei Prostatabiopsie und Prostatakrebsbehandlung eingesetzt werden können. Alle Teile dieser Arbeit wurden vollständig von Experten aus den Bereichen Medizin und Informatik evaluiert. Die Evaluierungsergebnisse bescheinigen den vorgestellten Methoden nicht nur deren Gültigkeit, sondern auch deren direkte Anwendbarkeit auf vielfältige Szenarien aus der klinischen Praxis.

Acknowledgments

I would like to thank everybody without whom this dissertation would not have been possible: First and foremost, I would like to thank my supervisor Prof. Dieter Schmalstieg for given me the opportunity of joining his research group and working with many different people. Dieter's drive and innovative ideas have effected me deeply. During my time with him at the and Graz University of Technology I learned a lot about research, leadership and team work. I want to thank him especially for all inspirational discussions and giving me the freedom to follow my own ideas. Special thanks goes to my co-examiner Prof. Markus Hadwiger for his very helpful suggestions, comments, and discussions.

Many thanks to my colleagues Markus Grabner, Stefan Hauswiesner, Markus Steinberger, Alexander Bornik, Rostislav Khlebnikov, Denis Kalkofen, Martin Urschler and my students Philip Voglreiter, Felix Nairz, Bernhard Roth and Markus Muchitsch for parts of the implementation and stimulating discussions. Furthermore, I want to thank Friedmann Roessler (University of Stuttgart), Ralph Brecheisen (Eindhoven University of Technology), and Stefan Bruckner (Vienna University of Technology) for providing their rendering systems for benchmarking and comparison, and Joseph Newman for many useful remarks.

Many fellow PhD students and friends have also helped and inspired me along the way, including Manuela Waldner, Marc Streit, Denis Kalkofen, Alexander Lex, Stefanie Zollmann, Tobias Langlotz, Eduardo Veas, Eric Mendez, Gerhard Schall, Lukas Gruber, Markus Tatzgern, and Christian Pirchheim. Also many thanks to all the professional expert participants of my online surveys related to this thesis's introductory section and application parts.

This work was funded by the European Union in FP7 VPH initiative under contract number 223877 (Project "Imppact"). Judith Muehl and colleagues wrote the associated project proposal.

I am grateful to my parents, Anna and Karl Kainz, who always believed in my scientific career and who supported me in my studies. They gave me the most valuable assets in life: love and encouragement. Dear Hanna, thank you for the all the encouragement, your time and the beer and wine. You provided me the necessary critical view on my work-life balance. Finally, I want to dedicate this work to my fiancée, Pia, in appreciation of her love and understanding. Without her patience and support I could not have finished this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

PART I - OVERVIEW

Chapter 1

Introduction

During nearly 25 years of volume graphics research, numerous ground-breaking visualization algorithms have been obtained for meaningful 3D image synthesis. Researchers still invent new image synthesis algorithms and make a substantial effort to accelerate these algorithms to accomplish interactive and real-time performance.

Volumetric data is common in medicine, geology and engineering, but the $O(n^3)$ complexity in data and algorithms has prevented the widespread use of volume graphics. Recently, however, 3D image processing and visualization algorithms have been parallelized and ported to graphics processing units (GPUs). This thesis discusses real-time issues of ray-based image synthesis beyond known GPU algorithms in a medical context. Today, medical diagnostics highly depends on volumetric imaging methods. Image processing and representation of this data has to be more accurate and comprehensible, and image investigation must be as fast as possible; ultimately not only because of cost reasons.

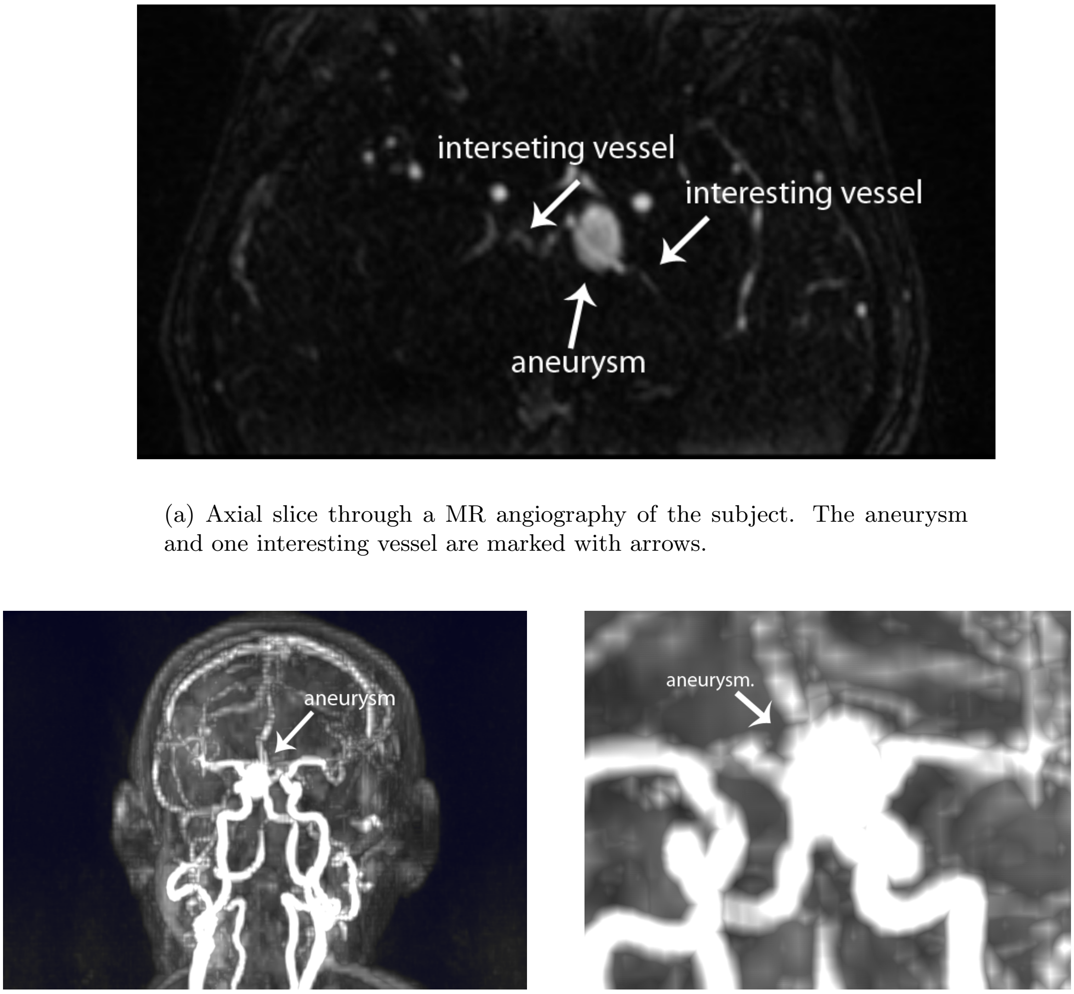
1.1 Motivation of this thesis

The first and very basic question in a medical context is if real-time 3D image synthesis is *necessary* at all. 2D slice section assessment is currently the most established diagnostic source and displaying 2D images from 3D data is trivial, fast and cheap. Furthermore many live 3D processing algorithms (e.g. 3D neonatal ultrasound) have only been developed because of the *pretty* resulting pictures but with no further diagnostic value. Of course, these methods might have financial and prestigious value for a clinic providing them. But they are not *necessary* for saving lives. We will show in the following sections, that there exist also several *necessary* and very important 3D image synthesis algorithms for diagnostics and we will investigate why those methods are not yet commonly used and expose their problems in the current clinical routine. We substantiate several observations in the upcoming analysis with a survey amongst 24 medical doctors (54% radiologists, 29% surgeons, 8% internists and 4% radiology physisits and 4% medical students) from all over Europe, of whom 50% have more than 5 years of professional experience. These observations are considered as application-oriented hypotheses (*AOH*) for this thesis, which are discussed and supported in this chapter. These application-oriented hypotheses form the base for our later discussed techniqual hypotheses.

3D image synthesis in medicine The main source for volumetric data in medicine is radiology. Therefore, it would be obvious that radiologists are the target users for volumetric visualization. However, radiologists are trained to gather information mostly from 2D image slices, originating from medical scanners. 3D image synthesis is rarely integrated into a radiologist's work flow. Furthermore, radiologists often do not trust direct 3D image synthesis methods. The general opinion of all radiologists we have been working with during the past seven years, is that 2D slices do not *pretend* information about the shape and appearance of a structure and thus 2D representation are the preferred diagnostic source for imaging sequences where a direct 2D slice-based assessment is possible. Angiography forms a suitable example to discuss this problem in detail:

Angiography is a medical imaging technique used to visualize the lumen (the inside) of blood vessels and organs. This is usually done by injecting a radio-opaque or magnetic contrast agent into the vessel of interest or into the whole vascular system by intravenous application. X-Ray-based techniques such as fluoroscopy and Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) are subsequently used as image modality. The resulting image shows all vessel lumens (which are filled with contrast agent) with a high and distinct intensity value. Figure 1.1 (a) shows one selected slice from a contrast enhanced image sequence for a human head compared to one of the most common 3D representations in clinical practice: Maximum Intensity Projection (MIP) [168] with the subtracted non-contrast enhanced native sequence, Figure 1.1 (b). From image acquisition on, the further investigation of the dataset highly depends on the necessary treatment and therefore on the necessary medical personnel. Diagnosis – which is in this case usually done by a radiologist – is based on 2D slice analysis. The main reason for that is that if a vessel is not filled with contrast agent and therefore not projected with high intensity it is not displayed with a MIP or other 3D image synthesis algorithms (illustrated in Figure 1.1). However, a missing vessel in 3D does not mean, that the vessel does not exist. A vessel can be stenosed or thrombosed, resulting in a decreased blood flow or it can simply be hidden by a structure with a higher intensity. A trained radiologist is still able to perceive at least the remaining perfusion in the area of that vessel which depicts as non-linear small contrast changes in usually 12-bit encoded medical images. All standard volume visualization methods or vessel segmentation methods are not able to reflect these subtle image variations. This fact makes 2D slice-based volumetric dataset investigation still the method of choice for a radiologist in this example.

An interventional radiologist might not be able to perform the necessary treatment. In this case – e.g. extensive surgery – clinicians are consulted. Clinicians, often surgeons, are trained to navigate inside the human body, obviously in 3D. For the angiography example, a 3D representation from medical scans (mostly MIP) is essential to provide a link



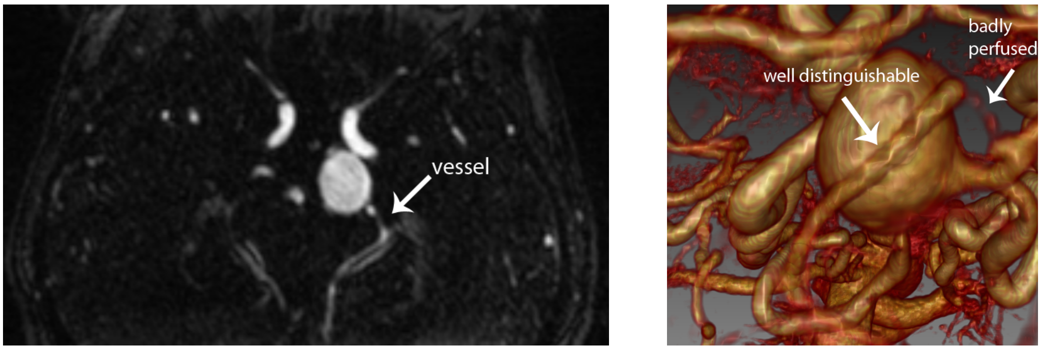
(a) Axial slice through a MR angiography of the subject. The aneurysm and one interesting vessel are marked with arrows.

(b) Maximum Intensity Projection (MIP) of the MR angiography sequence from (a). The vessels are clearly visible. The arrow is indicating the cerebral aneurysm of this subject.

(c) Closeup of the aneurysm in (b).

Figure 1.1: A human subject suffering from a cerebral aneurysm. (a) shows an axial slice through a MR angiography of contrast enhanced vessels with a subtracted previous native scan of the brain to get rid of low perfused image parts. This view would be used by radiologists for diagnosis. The reason is obvious when (a) is compared to (b) and its closeup in (c). Note that the marked vessel in (a) is hidden by the aneurysm in the MIP (b) and (c). However, the MIP can be freely rotated and gives a better impression for e.g. possible surgery access paths for clinicians.

between radiologists and clinicians and to illustrate pathological findings. Investigating an interesting vessel part from all sides without surrounding tissue is a vital procedure for e.g. vascular surgery planning in the current clinical practice. A MIP is simple and fast to compute but it has the severe drawback that structures of a lower intensity can be hidden by structures with a higher intensity as shown in Figure 1.1. To overcome this problem, direct volume rendering (DVR) – e.g. ray casting [60, 126] – can be used. Unfortunately, DVR algorithms show a tremendous algorithmic complexity and requires expert knowledge input for feasible diagnostic images (transfer function design problem, see Section 8). This is currently one of the main reasons, why more sophisticated direct 3D image synthesis algorithms find their way only slowly into the clinical practice. As recently as during the past years, hardware providing enough computational power got available also to normal clinical workstations. Besides that, imaging protocols and image synthesis have to be extensively tested and approved by health organizations as for example by the US Food and Drug Administration (FDA) to avoid wrong treatment decisions based on algorithmic shortcomings. This means for angiography that medical imaging devices supporting MIP representations have already been approved by the FDA in the early 1990th, whereas real-time DVR methods got first approved around the year 2000. However, the number of standard treatment procedures where DVR is used for intra-disciplinary communication rises. MR-based angiography is one of them, not only because of the hidden structures problem but also because of the better image quality, as shown in Figure 1.2 (b). Figure 1.2 shows that a well utilized DVR shows also structures which are hidden in a MIP (Figure 1.2) but that it still cannot display stenosed or thrombosed vessels which are not well filled with contrast agent.



(a) Axial slice through a MR angiography of the subject. A badly with contrast agent filled vessel is marked with an arrow.

(b) DVR of the same scene as shown in Figure 1.1 (b).

Figure 1.2: This figure shows the same subject as in Figure 1.1. The MIP is replaced by a DVR method in (b). Structures of similar intensity are now distinguishable but badly perfused vessels are still only identifiable on 2D slice representations of the original data.

The example of angiography and our overall experience with other clinical procedures lead to:

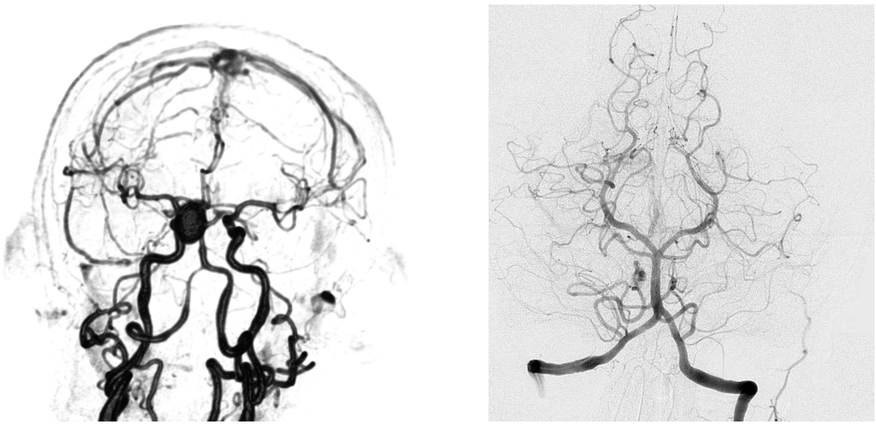
AOH1: *Medical 3D image synthesis algorithms must speed up the information finding process to be accepted by medical doctors. For standard diagnostic procedures, 3D representations do not provide additional information to radiologists but they are useful to illustrate pathological findings to other medical specialists who use that information for opinion making and intervention planning.*

Following the DVR visualization attempt of the angiography example leads to another observation. The basic idea of DVR is to accumulate intensity values along viewing rays through a volumetric dataset. Each intensity value is looked up in a discrete transfer function, containing color and opacity values. All intensity values of one ray are subsequently accumulated in the resulting image pixel. Considering e.g. a linear ramp as transfer function, which maps low intensity values to transparent image regions and high intensity samples to opaque regions, makes DVR of an angiography highly comparable to a well established interventional imaging technique: C-Arm fluoroscopy. A C-Arm is a relatively simple X-Ray-based device which can directly display X-Ray attenuation at a certain view port. The image contrast can be enhanced by injecting contrast agent during imaging. Figure 1.3 compares a linear ramp DVR to a fluoroscopy image during catheter-based minimal intervention. The similarity of the underlying principles (X-Ray attenuation vs. opacity accumulation) makes the DVR algorithm acceptable for clinicians and leads to:

AOH2: *If a 3D image synthesis algorithm is comprehensible and if it is related to a familiar physical principle, 3D image synthesis is accepted as diagnostic valuable tool and integrated into the clinical workflow.*

Besides 3D assisted angiography, which was first proposed by Napel and colleques [168], there are also further examples showing the same evidence for the need of 3D image synthesis in clinical practice. One of the most common needs is 3D image segmentation. Even though segmentation results are not often used for diagnosis, they are essential for Computer Assisted Intervention (CAI) and CAI-planning. Clear 3D boundary representations are essential for nearly all intervention planning systems and patient studies where derived organ specific measurements are required. Furthermore, most state-of-the-art advanced visualization algorithms cannot process volumetric data directly but have to use a polygonal representation of clearly defined image regions and hence anatomical structures.

So far, in this section, no clear evidence has been shown, that 3D image synthesis is *essential* for disease diagnosis in clinical practice. As long as the used image modalities are simple and individual, most radiologists prefer a straight forward 2D slice-by-slice investigation given that *AOH1* and *AOH2* are not fulfilled. However, many modern diagnostic procedures require either more than one image modality or an imaging mechanism whose result is too complex for 2D images or even both. Many MRI sequences for example do not only produce greylevel intensity images. They produce high dimensional matrix results for each sample, encoding different physiological conditions. A popular example is Diffusion Tensor Imaging (DTI) [122] which results the diffusion movement of water molecules



(a) Contrast enhanced MR angiography of the brain, displayed using DVR with a linear opacity transfer function.

(b) Contrast enhanced fluoroscopy of the brain, arteries

Figure 1.3: Figure (a) shows the same subject as in Figure 1.1. The DVR in (a) is used with a linear ramp opacity transfer function to underline the similarity of the algorithmic principle to the physical principle in (b). (b) shows a fluoroscopy image made with a C-Arm during catheter-based intervention. Contrast agent is injected into the vessel of interest.

within a test body. Because water tends to move along nerve fiber bundles, this sequence allows to draw conclusions about the spacial distribution of nerve fiber bundles, for example inside the living human brain. Figure 1.4 shows an example for the resulting spacial structures which were derived from several dozens MRI sequences of the same subject. A direct 2D investigation of these sequences is not possible anymore.

Also our further previous MRI work has shown clear evidence, that high dimensional MRI sequences cannot be interpreted without 3D image synthesis. In [194] we have shown that certain blood flow patterns (which can be measured by MRI as four dimensional dataset) allow a so far impossible non-invasive diagnosis of pulmonary hypertension. These patterns get only visible, if the measured blood-flow is visualized in 3D (with various flow-visualization techniques).

All these examples lead to:

AOH3: *3D image synthesis gets crucial if the data input dimensionality exceeds normal human experience.*

AOH1, *AOH2*, and *AOH3* lead to the conclusion that 3D image synthesis is necessary and appropriate for clinical diagnostic and interventional practice. However, which shortcomings prevent modern image synthesis algorithms like direct volume rendering (DVR) [126] and highly complex (e.g. photorealistic) representations, from becoming an integral part of daily hospital procedures? This question can be reformulated to:

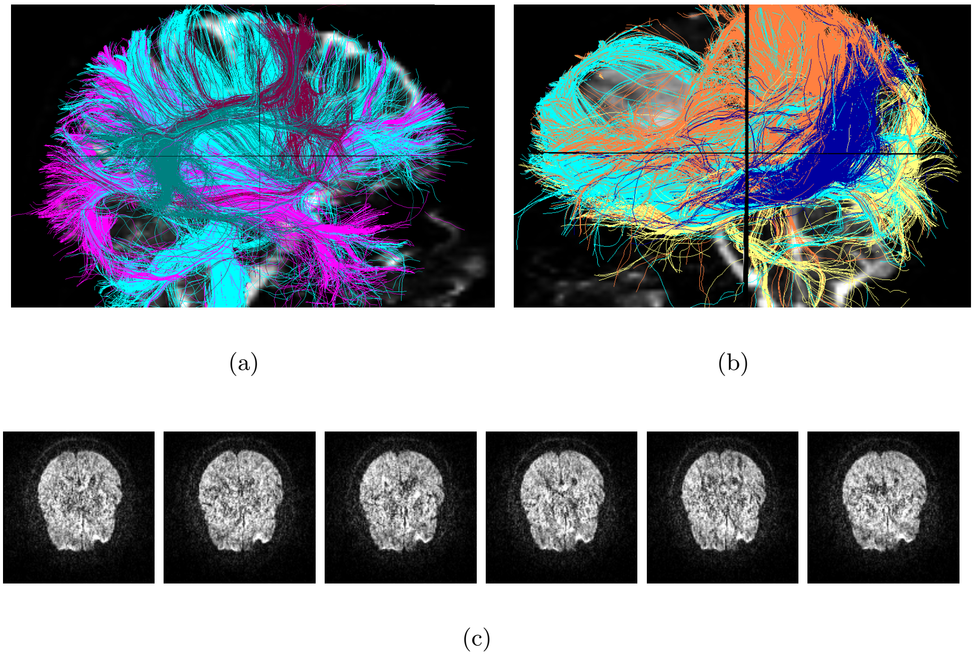


Figure 1.4: These images show a 3D overlay over a 2D image of certain fiber bundles which cross a certain area around a tumor in (a) and (b). The Diffusion Tensor Imaging (DTI) is therefore able to identify vulnerable structures which must not hurt during an intervention. Deriving this information from the raw image data is impossible. In this special case the raw data consists of 31 gradient direction DTI volumetric image sequences which allow no direct conclusion about the fiber direction. The bottom row (c) show some selected slice images from the raw data. The dataset is courtesy of Prof. B. Terwey, Klinikum Mitte, Bremen, Germany. Images taken from our IEEE Visualization Contest 2010 contribution [106].

Is 3D image quality and interactivity already feasible for clinical applications?
To answer this question, 3D image synthesis algorithms have to be classified more closely. For our context they can be roughly dived into

- rendering of polygonal surface representations with in hardware implemented rasterization units,
- Direct Volume Rendering (DVR),
- Non-Photorealistic Rendering (NPR) stylization,
- Photorealistic Rendering,
- algorithms where the rendering output is used for further operations, and
- hybrid approaches.

low quality – acceptable rendering speed – huge input space: *Rendering of polygonal surface representations* is the oldest and most common technique. A surface is generated by e.g. segmentation or iso-surface extraction and the resulting polygonal surface mesh is sent to the graphics processing unit (GPU) for rasterization. For simple and opaque models this approach can be sufficient. However, high polygon count surfaces as they result e.g. from vessel segmentation [5, 164] or many intersecting translucent objects are not feasible to be rendered at high resolutions with an interactive frame rate. Furthermore, surfaces tend to appear artificial because of very simple approximations of the surface illumination. Hence they are hard to integrate seamlessly into the real world experience of a surgeon or into Augmented Reality (AR) environments as they are sometimes used for inter-operative assistance. By using existing mesh simplification and smoothing algorithms, this approach can be considered as the most common for 3D image synthesis.

high quality – slow rendering speed – very limited input space: *Direct Volume Rendering (DVR)* [126] is not very common as diagnostic tool although the technique is nearly 25 years old. The main reason for that is the high complexity of the algorithm and therefore low frame rates during image synthesis without image quality reduction. Furthermore a higher input space compared to surface rendering is necessary to gain useful images. Whereas surface rendering requires only the definition of the camera and the lighting conditions for the simplest case, DVR requires additionally the generation of a full n-dimensional voxel-to-color-and-opacity transfer function. In recent years the DVR algorithm has been ported to parallel GPU programming languages which has mitigated at least the frame rate problem for interactive applications [65]. However, because of inherent limitations of the DVR algorithm, the vast majority of DVR rendering systems is not able to display more than one volume at a time or to intersect different datasets and volumes with geometry correctly. Note that this feature would be crucial for all state-of-the-art diagnostic methods which use multiple modalities. We have solved this problem and present the first system which is able bypass the normal problems with DVR at interactive frame rates in Chapter 3.

reduced quality – high rendering speed – limited input space: *Non-Photorealistic Rendering (NPR)* stylization techniques are very popular for clinical augmented reality applications. Firstly, most techniques reduce highly complex scenes to comprehensible image augmentation information to avoid visual clutter and to communicate the most important information in the simplest possible way [91]. Secondly, the majority of NPR techniques use very basic graphics operations (e.g. lines, strokes, and edges) which can usually be rendered at very high frame rates. However, a major problem of these attempts is the reduced or lost depth perception and the difficult estimation of *important* structures. We investigate that problem more closely in Chapter 4 and present an algorithm for high-quality interactive ray-based rendering which makes use of NPR techniques to boost image synthesis speed in Chapter 4.

very high quality - very slow rendering speed – limited input space: *Photorealistic rendering* of organic structures is common for endoscopic training simulators [111] and virtual colonoscopy [148]. Although these systems are well accepted by surgeons, the photorealism is restricted to *textured rendering of polygonal surface representations* illuminated with a high specular composed to simulate tissue moisture. “Real” photorealistic rendering algorithms, as for example ray-tracing [74], radiosity [45] etc., are not used at all in the clinical practice. Their extreme computational complexity and restriction to geometric objects has not allowed an interactive use so far. We present an algorithm in Chapter 4 which might be able to change that fact in future.

very high quality - very slow rendering speed – limited input space: *Algorithms where the rendering output is used for further operations* are often used for inter-patient and intra-patient registration. For these applications, real-time performance of the rendering system is not essential, since no user interaction is involved into the image generation process and the result are often derived values only. Most algorithms which use 3D-image synthesis as intermediate input often try to align a projected image of an object to an image modality whose output is already a projection of the scene. However, these applications are also required to deliver a result within reasonable time, to be accepted in the clinical practice. Therefore, the image generation part should not form the bottleneck of the application. We present such an algorithm in Chapter 5, where we solved the problem of aligning a patient’s fluoroscopy images to a previously taken CT-Scan.

very high quality - slow rendering speed – large input space: *Hybrid approaches* are sometimes used to provide different communication channels for multiple sources of information. Bruckner for example makes extensive use of NPR and DVR techniques to provide interactive illustrative volume visualization [26] for the effective communication of complex subjects and to provide a solution for the Focus and Context(F&C) problem [29]. Their problems are obviously formed by a linear combination of the shortcomings mentioned above. Our approach [105] is able to combine e.g. rendering results of polygonal surface representations with DVR.

The break-down of 3D image synthesis algorithms and their outlined limitations as given above leads to:

AOH4: *State-of-the-art 3D image synthesis algorithms are either not able to provide the necessary image quality or the necessary rendering speed, or they are restricted by the amount of input data. This prevents a common use of these techniques in the clinical practice and for clinical augmented reality applications or for applications where the rendering result is used as intermediate result and where the overall result must be available within reasonable time.*

1.2 Application-oriented hypotheses evaluation

To underline our general application-oriented hypotheses from Section 1.1 we have performed a survey with n=24 independent radiologists and clinicians (54% radiologists, 29% surgeons, 8% internists and 4% radiology physicists and 4% medical students), of whom 25% have more than ten, 25% have between five and ten, 33% have between two and five, and 17% have less than two years of professional experience. The specific knowledge of certain imaging modalities of the attendees is shown in Table 1.1. The full questionnaire can be found in Appendix A.

Modality	**Experience**
CT	87.5%
MRI (standard, e.g. T1, T2...)	83.3%
X-ray (C-Arm, film...)	75.0%
Ultrasound	70.8%
MRI (advanced sequences, e.g. fMRI, DTI, 4D...)	62.5%
Scintigraphy	58.3%
PET	45.8%
SPECT	37.5%
4D-CT	25.0%
Thermography (e.g. mamma)	8.3%
Other	0.0%
Total	100%

Table 1.1: This table shows the experience (permanent use) with certain modalities of the attendees of our survey.

We used the opportunity to evaluate also personal opinions of the survey participants about 3D assisted procedures vs. 2D slice investigation and those situations when the attendees are used to use 3D images instead of 2D slice views. The personal opinions on the importance for certain tasks of 3D image synthesis algorithms are summarized in Table 1.2. The general preference of 2D or 3D clearly indicates that the required representation strongly depends on the kind of application (63% answered “it depends on the application”, whereas 21% prefer always scrolling through slices.). The common use of 3D image synthesis versus 2D slice representation for certain examples is compared in Table 1.3.

Furthermore we have evaluated the shortcomings of the currently used 3D image synthesis algorithms and their enhancement requirements. The user’s satisfaction level for

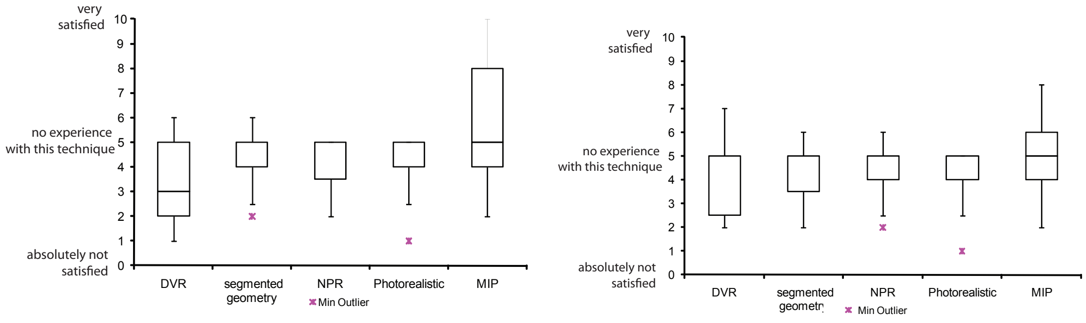
		Frequency	Percent	Valid Percent
Valid	Radiology and radiologic treatment planning	7	29.2%	31.8%
	Surgery and intervention planning	7	29.2%	31.8%
	Interdisciplinary communication and interdisciplinary intervention planning	5	20.8%	22.7%
	General practitioners and patient communication/explanation	2	8.3%	9.1%
	Internal medicine	1	4.2%	4.5%
	Total	22	91.7%	100.0%
Missing		2	8.3%	

Table 1.2: Personal opinions on when 3D image synthesis from medical volumetric data is most important.

Application	2D preference	3D preference
general diagnosis	33.3%	12.5%
millimeter accurate planning of radiological interventions	33.3%	0%
to get an overview over the whole scan	12.5%	33.3%
to communicate certain conditions to colleques from the same field of expertise	16.7%	20.8%
to communicate certain conditions to colleques from other fields of expertise	12.5%	16.7%
planning of surgery interventions	8.3%	8.3%
investigation of functional imaging (e.g. DTI, PC-MRI..)	0%	12.5%
surgery navigation	4.2%	4.2%
minimal invasive surgery navigation	8.3%	8.3%
angiography analysis	12.5%	16.7%
Other	4.2%	0%

Table 1.3: This table shows the common use of 2D versus 3D for certain example applications. Multiple answers have been possible.

image quality is outlined in Figure 1.5(a) and for rendering speed in Figure 1.5(b). Except for MIP, the results show clear evidence that both need further improvements in future. Personal interviews have shown that the results on MIP can be considered as outliers, because many clinicians often misuse the term MIP for all direct volumetric 3D rendering techniques (including DVR) and for some it is still the only 3D image representation they know.



(a) Image quality satisfaction.

(b) Image generation speed respectively interactivity satisfaction.

Figure 1.5: This figure shows the general satisfaction level in medicine for the image quality (a) and rendering speed (b), as given by the noted algorithms. The ends of the whisker are set at 1.5*IQR above the third quartile (Q3) and 1.5*IQR below the first quartile (Q1). If the minimum or maximum values are outside this range, then they are shown as outliers. The normal convention for box plots is to show all the outliers, but to simplify this figure, only the min and max outliers are shown.

Finally we measured the overall agreement with *AOH1-4* from Section 1.1. The results are shown in Figure 1.6. To verify these application-oriented hypothesis more closely, we have also tested the opposite statements (falsification [193]) which are listed below with their survey results. We consider for the falsification case only a clear and comprehensible *"Yes"* as falsified. Both results show a strong evidence, that our hypotheses (*cf.*, *AOH1-4*) are overall correct.

- **Plausibility check of *AOH1***: There exists a 3D image synthesis algorithms, which does not speed up clinical communication or the information finding process, but which is overall accepted by medical doctors.

  55% of the survey participants anserwed *"No"*, 36% answered *"I definitely don't know"* and one participant declared MIP as such a technique, which can be disproved with the angiography example from Section 1.1 and the further answers for the above questions.

- **Plausibility check of *AOH2***: There exists non-comprehensible 3D image synthesis

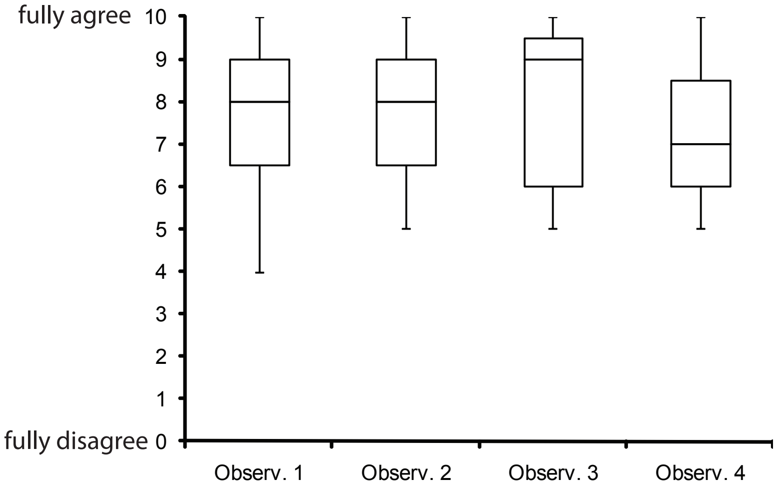


Figure 1.6: This plot shows the overall agreement of the survey participants with observation 1 - 4.

algorithm which has been integrated into a clinical workflow.

36% of the surevey participants anserwed *"No"* and 64% answered *"I definitely don't know"*.

- **Plausibility check of *AOH3***: There exists a human being who is able to interpret more than three dimensional raw data directly (e.g. DTI or PC-MRI data).

  55% of the surevey participants anserwed *"No"* and 45% answered *"I definitely don't know"*.

- **Plausibility check of *AOH4***: There exists a 3D image synthesis system which is able to process dozens of input data sets, to render the result at interactive frame rates at high quality and which is already standard in clinical visualization systems or which is used for intermediate data generation in other algorithms.

  37% of the surevey participants anserwed *"No"* and 55% answered *"I definitely don't know"*. One attendee mentioned DTI as such a technique. This is true if one considers the whole processing pipeline, which might be several hours for DTI preprocessing. A clearer falsifiaction statement in this case would have been: "*There exists a 3D image synthesis system which is able to process dozens of input data sets **online (no preprocessing)**, [...]*".

1.3 Organization and scope of this thesis

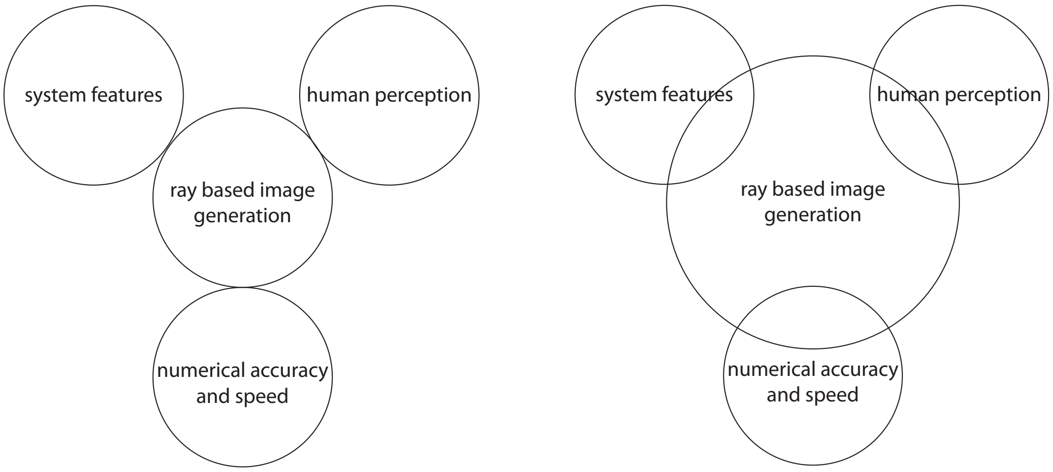
Section 1.1 motivated the need for 3D image synthesis in clinical practice. However, Section 1.1 diminished the upcoming enthusiasm with the observation that modern high-quality visualization algorithms are not yet feasible enough to get the professional acceptance they would need for a permanent application. This work solves several technical aspects when it comes to highly complex volumetric data processing and visualization methods as their basics are outlined in Section 2.2. We consider guaranteed interactivity of ray-based image synthesis as the base for feasibility and acceptance in the clinic. Human-Computer-Interaction (HCI) problems in medical data interaction, which build upon feasible rendering systems, are beyond the scope of this work and are therefore not covered. Figure 1.7 outlines the goal of this thesis, namely to maximize the optimization potential of classical ray-based image synthesis in three ways. We introduce the term *ray-based image synthesis* in this thesis and break it down into its sub-algorithms and applications in detail in Section 2.2. In contrast to general 3D image synthesis, which describes all kinds of object projections, ray-based image synthesis can be seen as an umbrella term for all image synthesis algorithms which use the metaphor of rays which are *shot* through each pixel in the image plane. To improve such systems we aim for a better utilization of modern host system features (*e.g.*, novel GPU features) in a first step. In the following, we exploit the peculiarities of the human visual system to obtain a fast and accurate ray-based image synthesis system, and third we go for more numerical accuracy and speed, which is usually restricted by the bit-depth of the host system, by applying automatic differentiation (AD) methods on ray-based image generation algorithms. All these methods have been successfully applied to real-world applications.

1.3.1 Technical hypotheses

Further on, this thesis discusses the technical suitability of ray-based image synthesis on the GPU for practical clinical applications. Therefore, we also pose three technical hypotheses (*TH*) additionally to the in Section 1.1 stated user-based observations and application-oriented hypotheses. The application-oriented hypotheses have already been discussed in detail in Section 1.1 and it has been shown that *AOH1-3* motivate the overall need for 3D image synthesis in the clinical practice. However, *AOH4* which states that state-of-the-art rendering system are not able to provide the necessary image quality or the necessary rendering speed, or they are restricted by the amount of input data, leads to the following technical hypotheses statements which are examined throughout the remainder of this document:

TH1	*Ray-based image synthesis can be extended to support multiple input datasets without a severe loss of interactivity.*
TH2	*Ray-based image synthesis can make use of perceptually prominent object features to boost or guarantee frame-rates.*
TH3	*Ray-based image synthesis rendering cores can be analytically derived and therefore be used for highly accurate and fast 2D/3D registration tasks.*

Table 1.4: Hypotheses of this thesis.


system features
human perception
ray based image generation
numerical accuracy and speed
system features
human perception
ray based image generation
numerical accuracy and speed

(a) The current optimization of ray-based image synthesis kernels is outlined in this figure. General purpose attempts reach the border of the shown optimization alternatives but do not go beyond.

(b) This thesis goes beyond the borders of the current state of algorithmic optimizations.

Figure 1.7: In this thesis we identified three main targets with optimization potential. First, specializing ray-based image synthesis to certain hardware features gives novel options in Chapter 3. Second, exploiting the peculiarities of the human visual system leads finally to a fast and accurate ray-based image synthesis system in Chapter 4. Third, numerical accuracy and gaining accurate results with reasonable speed is still restricted by the host system, but analytical improvements are possible as shown in Chapter 5.

To substantiated these technical hypothesis this thesis is organized in three main parts.

Part I encloses the current chapters and gives an introduction to this work, an overview over this thesis, the state-of-the-art and recent achievements for GPU-based rendering systems and their use in the clinical practice. Furthermore, this part outlines individual publications of the author which have been published previously about this thesis and the collaboration with other colleagues.

Part II describes our own achievements and discusses each of the above posed hypothesis in detail. Therefore, we first show in Chapter 3 that current ray-based image generation systems are not sufficiently optimized and feasible enough for many clinical applications which require multiple volumes in the same scene. State-of-the-art multi-volume rendering ray setup steps are simply too slow to solve this problem efficiently. Therefore we present a new GPU-based rendering system for *polyhedral ray-casting* to substantiate *TH1*. Our approach supports a large number of volumes, complex translucent and concave polyhedral objects as well as CSG intersections of volumes and geometry in any combination. The system (including the rasterization stage) is implemented entirely in CUDA, which allows full control of the memory hierarchy, in particular access to high bandwidth and low latency shared memory. High depth complexity, which is problematic for conventional approaches based on depth peeling, can be handled successfully. As far as we know, our approach is the first framework for multi volume rendering which provides interactive frame rates when concurrently rendering more than 50 arbitrarily overlapping volumes on current graphics hardware. The flexibility of this approach is also demonstrated by replacing the normally used Phong lighting model with a single scattering and shadowing approximation attempt in Section 3.4.

To provide the necessary accuracy and interactivity for clinical applications, which we have identified as major concerns of potential expert users in our survey in Section 1.2, and to substantiate *TH2*, we present in Chapter 4 a new method to control scene sampling in complex ray-based rendering environments. We propose to constrain image sampling density with object features, which are known to support the comprehension of three-dimensional shape. The method uses Non-Photorealistic Rendering techniques and image-based saliency computations to extract visual important features. In order to map different feature types to sampling densities, we also present an evaluation of the features' impact on the resulting image quality. In addition, we present a discussion of three different strategies to reconstruct the image from sparse sampling data. Furthermore, this chapter presents an algorithm which uses our method in order to *guarantee* a desired minimal frame rate. Our scheduling algorithm maximizes the utilization of each given time slice by rendering features in the order of their corresponding importance values until a time constraint is reached. We demonstrate how our method can be used to boost or stabilize the rendering time in complex ray-based environments consisting of geometric as well as volumetric data.

During the final chapter of **Part II**, we investigate novel GPU-based accuracy and rendering speed performance improvements for single volumetric scans to substantiate *TH3*. We have demonstrated the applicability of image-based similarity measures to pose optimization problems, occurring e.g. with C-Arm monitored interventions, performed on fast parallel stream processors utilizing an analytically computed gradient. This is achieved by automatic differentiation (AD) of the basic ray-casting kernel and used with an iterative optimizer. This results in a significant speedup and accuracy improvement compared to numerical gradient approximations (such as central differences). The use of AD techniques in Chapter 5 is in fact not limited to registration input, but opens a new dimension to accelerate any registration task where an objective function has to be optimized with respect to the transformation parameters.

Part III demonstrates the applicability of our methods on various concrete applications. First we exploit and map a natural phenomena to the problem of tumor accessibility planning. When the sun shines through clouds or trees into dusty or humid air which contains enough scattering particles, an observer can see several ray bundles shaped by obstacles. We think of a tumor as the light source and the important brain structures as the obstacles. The resulting scatter volume is rendered together with anatomical data by our polyhedral multi-volume DVR system. Furthermore, we provide constant frame rate for applications with hard real time requirements like interventional augmented reality as shown in Section 6.

The applicability of AD methods is shown for a concrete 2D/3D registration problem which occurs during prostate biopsy and surgery. A solution is the registration of a set of X-ray images taken during an intervention to a CT data set of the same patient taken before, which is also known as 2D/3D registration. To achieve the necessary accuracy and image generation speed, we use a specialized X-Ray target, which allows a registration initialization from a single image within one second with high accuracy. The core 2D/3D registration approach is based on an iterative nonlinear optimizer, where the gradient of the underlying similarity measure is computed analytically, thus reducing the bandwidth and computational requirements of the solver. The analytic gradient computation code is derived from the similarity measure code by means of automatic differentiation as presented in **Part II**, Chapter 5.

In another application we investigate the tedious task of transfer function design for multi volume rendering systems. Since DVR is still difficult to use for non-expert users, we show in Section 8 an application using our polyhedral multi-volume DVR system for the automatic suggestion of useful transfer functions of a certain dataset or multiple depending and intersecting patient scans.

Finally, we discuss the overall achievements of this thesis and parts which could have been solved differently along with further directions for future work in a concluding **Part IV** and Chapter 9.

This thesis is also a good starting point for anyone who is interested in the state-of-the-art of direct volume visualization methods and ray-based rendering algorithms in the clinical practice. However, considering certain results from our surveys, this thesis can only be a further step towards a common use of ray-based image synthesis in medicine.

1.3.2 Contributions of the thesis at a glance

This thesis describes several attempts of the author, which have been made towards highly interactive high-quality medical volume visualization and intermediate use of ray-based image synthesis and to substantiate *AOH1-4* and *TH1-3.* The following list gives an overview over the achievements of this thesis and how they are related to the posed hypotheses:

- **Expert surveys.** Surveys amongst experts which gives indication for the right application of 3D images synthesis in medicine and which also show up the problems of state-of-the-art direct volume visualization methods. These surveys show a strong evidence for the correctness of our observations and therefore also for our application-oriented hypotheses (*cf.*, *AOH1-4* in Section 1.1).

- **Multi-volume real-time polyhedral DVR.** To the best of our knowledge, our work is the first that enables the display of dozens of volumetric datasets with complex intersecting and clipping geometry on a consumer GPU with real-time frame rates (Section 3.2). This approach shows that *TH1* holds, since our approach scales only with the available GPU memory.

- **Volumetric and geometric real-time CSG interaction.** In Section 3.2, CSG operations on a volume segmented into multiple regions with polyhedral boundaries can be used to assign individual transfer functions to each region. Each region can be transformed individually, allowing exploded views. This approach eases the general work flow with volumetric data and supports *AOH1* and *AOH2*.

- **Combined real-time rendering of translucent geometry and volumes.** Volumes can intersect and be intersected with transparent, concave geometry in Section 3.2.2. To combine volumetric data with for example segmentation results in a straight forward way and supports *AOH1* and *AOH2*.

- **Illumination model improvements.** In Section 3.4 we show that our presented systems are easy to extend, for example with online scatter approximation. This approach demonstrates the flexibility of our polyhedral DVR system.

- **Perceptual feature evaluation.** In order to put different emphasis on different object features we evaluate a known set of features and categorize them based on their image quality enhancement (Section 4.5). Knowing which features are important for perceptually motivated sampling strategies forms the basis for an implementation of *TH2*.

- **Perceptually motivated ray setup.** We present a method which allows to optimize the ratio between the sampling rate of the scene and its resulting perceptual quality (Section 4.3). Boosting a given maximum frame rate with this sampling strategy substantiates *TH2*.

- **Guaranteed frame rates for hard real-time constraints.** We present an algorithm which guarantees frame rates while maximizing visual quality within an available time frame. (*cf.*, Section 4.4). This approach fulfills *TH2* also in terms of *guaranteed frame rate rendering*.

- **Volumetric tumor accessibility planning.** We have successfully applied our algorithms in Section 6 to a metaphor from nature where the tumor is regarded as light source in scattering media. Save access paths can be identified easily with this method in multiple accuracy levels. This idea supports *AOH2* and *AOH3*, because we use a familiar natural phenomenon as visualization method.

- **Augmented Reality tumor accessibility visualization.** We have applied our tumor accessibility method to an interventional AR see-through prototype in Section 6. This approach shows additional evidence that *TH1* and *TH2* are possible.

- **Registration initialization X-Ray target.** To achieve accuracy and image generation speed for specialized tasks like 2D/3D registration, we have developed a specialized X-Ray target, which allows a registration initialization from a single image within one second with already high accuracy in Section 7. Having a registration initialization forms the basis for a working *TH3*.

- **Highly accurate registration with specialized DVR.** We have developed a method in Section 5.4, which is based on an iterative nonlinear optimizer, where the gradient of the underlying similarity measure is computed analytically by automatic differentiation. By doing so, we substantiate *TH3*.

- **Automatic generation of separable transfer functions of multiple volumes.** We show in Section 8 a data centric method, combining the alpha histogram and partial range histogram approach to prevent the tedious trial and error work of manual transfer function design which supports the feasibility claim of *AOH3* and *AOH4*.

1.4 Individual publications about this thesis and collaboration statement

Results of this work have been previously published. The following paper references describe the preliminary outcome of the work and the contribution of collaborating colleagues: Besides the below mentioned work, the author supervised four graduate students during their Master's thesis from whom Markus Muchitsch implemented the presented transfer function design gallery approach in Section 8 and Bernhard Roth implemented a preliminary version of the light transfer approximation as presented in Section 3.4. The author of this thesis refined, optimized and extended their implementations and taught them scientific writing.

- Rostislav Khlebnikov, **Bernhard Kainz**, Bernhard Roth, Judith Muehl and Dieter Schmalstieg: **GPU-based on-the-fly light emission-absorption approximation for direct multi-volume rendering.**, in Proceedings of Eurographics 2011, posters.

  While Rostislav Khlebnikov refined the final version of the paper and presented this work at EG2011, the author of this thesis refined a basic implementation from Bernhard Roth and added his own ideas about the light transport approximation. Furthermore he wrote the first submission of this work and produced the corresponding video. Judith Muehl contributed to the proposal for the EU project IMPPACT and ensured therefore the funding of this work. Dieter Schmalstieg refined the content of this work and the final version of this publication. This work has been used in this thesis in Section 3.4.

- Rostislav Khlebnikov, **Bernhard Kainz**, Judith Muehl and Dieter Schmalstieg: **Crepuscular rays for tumor accessibility planning**, submitted to IEEE Visualization 2011.

  Rostislav Khlebnikov implemented the GPU 3D accessibility volume computation and figured out novel information accumulation schemes besides his contribution to the paper and video content. The author of this thesis contributed the main idea, the 3D and 2D rendering system implementation, the evaluation of this work and paper and video content. Judith Muehl did additional research on related work and contributed to the proposal for the EU project IMPPACT. Dieter Schmalstieg refined the content of this work with his experience and coordinated the team. This work has been used in this thesis in Section 6.

- **Bernhard Kainz**, Markus Steinberger, Stefan Hauswiesner, Rostislav Khlebnikov, Denis Kalkofen and Dieter Schmalstieg: **Stylization-based ray prioritization for guaranteed frame rates**, submitted to NPAR 2011

- **Bernhard Kainz**, Markus Steinberger, Stefan Hauswiesner, Rostislav Khlebnikov, Denis Kalkofen, Dieter Schmalstieg: **Using Perceptual Features to Prioritize Ray-based Image Generation**, in Proceedings of Symposium on Interactive 3D Graphics and Games 2011 (I3D), posters

  While the second work is a preliminary sketch of the main system, the first publication described the whole rendering system in detail, to which the author contributed the main idea, large parts of the implementation and most of the content of the corresponding paper and video. Markus Steinberger implemented the queue and sampling pattern which is necessary to achieve guaranteed frame rates and added this part to the paper content. Stefan Hauswiesner investigated different image-based GPU reconstruction algorithms and did a large part of the necessary evaluation of the investigated perceptual features. Rostislav Khlebnikov revised the paper and helped with the video. Denis Kalkofen and Dieter Schmalstieg added their experience to the content of the paper, the video and the overall system. This work has been used in this thesis in Section 4.

- **Bernhard Kainz**, Rostislav Khlebnikov, Alexander Bornik, Dieter Schmalstieg: **Multivariate Beam Ray Obstacle Visualization for Brain Tumor Resection**, 2010 IEEE Visualization Contest, IEEE Spectrum, Brain Beauty Contest

  The author of this thesis did the necessary implementation, paper writing and video production. Rostislav Khlebnikov helped revising the paper and the video and Alexander Bornik contributed the space carving algorithm for the used rendering system. Dieter Schmalstieg added his experience to the content of the paper, the video and the overall system. This work has been used in some parts of the introductory sections of this thesis.

- **Bernhard Kainz**, Markus Grabner, Alexander Bornik, Stefan Hauswiesner, Judith Muehl, Dieter Schmalstieg: **Ray casting of multiple volumetric datasets with polyhedral boundaries on manycore GPUs**. ACM Trans. Graph., Volume 28, Issue 5 (December 2009), Proceedings of ACM SIGGRAPH Asia 2009, Article No. 152

  The author contributed the main idea, most parts of the core implementation, the transfer function editing system, tests for proper rasterization algorithms, parts of the light transport algorithm, produced the corresponding video, wrote a major part of the final publication and organized the necessary teamwork. Markus Grabner developed a novel rasterization algorithm besides the highly sophisticated utilized software design for GPU programming and added those parts to this publication. Alexander Bornik developed and implemented the constructive solid geometry (CSG) algorithms of this work and added those parts to the paper. Stefan Hauswiesner refined the rendering core and tested several versions of fragment sorting on different GPU memory hierarchies. He also added those parts to the final publication. Judith Muehl contributed to the proposal for the EU project IMPPACT and ensured therefore the funding of this work. Dieter Schmalstieg added his experience, coordinated the proof reading and wrote a considerable part of the final publication. This work has been used in this thesis in Section 3.

- Judith Muehl, **Bernhard Kainz**, Alexander Bornik, Markus Grabner, Stefan Hauswiesner and Dieter Schmalstieg: **The Future of Volume Graphics in Medical Virtual Reality**. World Congress on Medical Physics and Biomedical Engineering, IFMBE Proceedings, pp. 1349-1352, 2009.

  The author contributed the part about multi-volume rendering systems, Alexander Bornik wrote the parts about volumetric CSG operations, Markus Grabner added considerations about GPU technology, Stefan Hauswiesner investigated the possibilities of image-based rendering systems and Dieter Schmalstieg refined the final version of this work. This work has been used in some parts of the introductory sections of this thesis.

- **Bernhard Kainz**, Markus Grabner and Matthias Ruether: **Fast Marker-Based C-Arm Pose Estimation**. LNCS, proceedings of the 11th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2008), New York, pp. 652-659 (2008)

  The author implemented and tested the developed X-Ray target and optimized the used LFSR code to provide a minimal pattern size for pose estimation. Furthermore he wrote the corresponding publication and coordinated the input of the co-authors. Markus Grabner contributed the main idea and a first prototype of the used X-Ray target. He refined also the final publication and integrated the outcome of this work into the intended registration system. Matthias Ruether added algorithms for pose estimation and camera calibration and advised the author in robot vision methods. This work has been used in this thesis in Section 7.

- Markus Grabner, Thomas Pock, Tobias Gross and **Bernhard Kainz**: **Automatic Differentiation for GPU-Accelerated 2D/3D Registration**. in: Lecture Notes in Computational Science and Engineering 64, Proceedings of the 5th International Conference on Automatic Differentiation, pp. 259-269 (2008)

  Markus Grabner and Thomas Pock developed the main idea for this work and the mathematical background. Tobias Gross implemented the basic version of the software framework and the digitally reconstructed radiograph algorithm. The author revised the final publication and optimized the implementation for real-time scenarios. Furthermore, he developed a novel GPU-based image in-painting and distortion model for a final visually improved result. This work has been used in this thesis in Section 5.

1.5 Additional publications beyond the scope of this thesis

Besides the work from Section 1.4, the author contributed also to additional work. This work originates mainly from his interests in biomedical engineering, which is the author's second foothold besides computer graphics. In this section the collaboration statement only outlines the contribution of Bernhard Kainz because the following secondary work is just exemplary considered in this thesis.

- S. Koestenbauer, P. Stiegler, V. Stadlbauer, U. Mayrhauser, B. Leber, D. Blattl, **B. Kainz**, O. Reich, R. H. Portugaller, I. Wiederstein and K.H. Tscheliessnigg: **An easy way to visualize large-scale sections**, to appear in Journal of Surgical Radiology

  The author developed the scan method and calculated the necessary light reflectance of the scanning device.

- Tuomas Alhonnoro, Mika Pollari, Mikko Lilja, Ronan Flanagan, **Bernhard Kainz**, Judith Muehl, Ursula Mayrhauser, Horst Portugaller, Philipp Stiegler, Karlheinz Tscheliessnigg: **Vessel Segmentation for Ablation Treatment Planning and Simulation**. Medical Image Computing And Computer-Assisted Intervention – MICCAI 2010 Lecture Notes in Computer Science, 2010, Volume 6361/2010, 45-52

  The author implemented parts of the used medical data interaction system and collaborated with the first authors for several weeks at their institution and contributed to the core algorithms.

- Judith K. Muehl, **Bernhard K. Kainz**, Horst R. Portugaller, Philipp B. Stiegler and Christian Bauer: Computer oriented image acquisition of the liver: **Toward**

a better numerical model for radiofrequency ablation. Proc. of the IEEE Engineering in Medicine and Biology Society Annual Conference., pp. 3755-3758, IEEE, September 2009

The author wrote a considerable part of the final publication, did the necessary data evaluation and implemented the data processing pipeline.

- **Bernhard Kainz**, Ursula Reiter, Gert Reiter, and Dieter Schmalstieg: In Vivo Interactive Visualization Of Four-Dimensional Blood Flow Patterns. DOI 10.1007/s00371-009-0315-7, in: The Visual Computer, Volume 25, Number 9 / September, pp. 853-862 (2009)

  The author implemented the core visualization and data management system and the used visualization algorithms. Besides writing the major part of this publication, he also evaluated different data sets of sick and healthy patients and participated as healthy subject in various studies of the collaborating medical partners.

- Gert Reiter, Ursula Reiter, Gabor Kovacs, **Bernhard Kainz**, Karin Schmidt, Robert Maier, Horst Olschewski, Rainer Rienmüller: Magnetic resonance-derived three-dimensional blood flow patterns as marker of manifest pulmonary hypertension. European Journal of Radiology, Proc. 21st European Congress of Radiology, March 2009.

- Ursula Reiter, Gert Reiter, Gabor Kovacs, **Bernhard Kainz**, Karin Schmidt, Robert Maier, Horst Olschewski, Rainer Rienmüller: Non-invasive measurement of elevated mean pulmonary arterial pressure. European Journal of Radiology, Proc. 21st European Congress of Radiology, March 2009.

  For both publications the author implemented and tested the underlying visualization system which is still in use for those kind of applications.

- Daniel Wagner, **Bernhard Kainz** and Dieter Schmalstieg: Realtime 3D Graphics Programming Using the Quake3 Engine. in: CGEMS Computer Graphics Educational Materials Source, SIGGRAPH Educational Committee and the Eurographics Education Board (2008), ACM Honorable Mention

  The author was strongly involved into the corresponding lecture and wrote a considerable part of this publication.

- Gert Reiter, Ursula Reiter, Gabor Kovacs, **Bernhard Kainz**, Karin Schmidt, Robert Maier, Horst Olschewski, and Rainer Rienmüller. MR Derived Three-Dimensional Blood Flow Patterns In The Main Pulmonary Artery As Marker Of Pulmonary Hypertension And Measure Of Elevated Mean Pulmonary Arterial Pressure. in: Circulation: Cardiovascular Imaging. 1:23-30; (2008)

- Gert Reiter, Ursula Reiter, **Bernhard Kainz**, Andreas Greiser, Horst Bischof and Rainer Rienmüller. MR vector field measurement and visualization of normal and pathological time-resolved three-dimensional cardiovascular blood flow patterns. in: Journal of cardiovascular magnetic resonance 9 (2007) 2, S. 237 - 238

For both publications the author implemented and tested the underlying visualization system which is still in use for those kind of applications and contributed the description of the system to the final publications.

Chapter 2

Clinical state-of-the-art and related work

This Chapter summarizes general scientific findings in extracts related to this thesis in two main sections. Section 2.1 gives an overview over work which has been done in direction of the general questions from Chapter 1. Subsequently Section 2.2 gives an impression of the vast amount of research which has be done to solve technical problems with medical 3D image synthesis algorithms when hard-real time requirements have to be fulfilled. The following sections skip research on computer vision methods which may be applied between image acquisition and visualization. This research area would be far beyond the scope of this thesis.

2.1 State-of-the-art 3D image synthesis in medicine

Since in the 1980s the first direct 3D image synthesis algorithms for medical datasets have been introduced, researches have found useful areas of application for the clinical practice and clinicians have tried to integrate these new methods into their workflow. Obviously the literature gives a lot of examples for future and perhaps possible application of newly presented methods. However, only a few research articles deal with our general questions from Chapter 1 in a retrospective way. McCormick and DeFanti [53, 153] predicted in the late 1980s a broad range of applications and research opportunities for medical image visualization. Most of their research questions like *"a smooth integration of useful 3D images for diagnostics"* are still not completely solved. One reason for that is the very long time for new medical methods until they are established [57]. Another reason is, that the data resolution in medicine grows with the same speed as the computational power of ordinary workstations [169, 225]. Both facts complicate a retrospective view on the general use of 3D image synthesis in medicine. Therefore we further concentrate on selected successful examples for applications of 3D image synthesis in medicine from volumetric data.

CT-Angiography has also been the most stated answer for question 12 in our survey. This application is probably the most established application of direct 3D volume visualization in medicine. Introduced by Rubin *et al.* [203] in the early 1990s and later improved to the current state-of-the-art of multi-detector row imaging by Catalano *et al.* [37], several dozens of medical studies have shown the usefulness of this method in the clinical routine. Rossum *et al.* [231] showed the applicability of angiograms for pulmonary embolism, however not yet with interactive 3D methods. Later studies by Messenger *et al.* [156], Anxionnat *et al.* [9], and Lell *et al.* [125] evaluated direct and interactive 3D volumetric methods for their diagnostic value for coronary vessels and intra cranial aneurysms.

The most common 3D image synthesis algorithms for this problem are currently texture slicing methods for direct volume rendering and maximum intensity projection after certain preprocessing steps (e.g. bone removal). Despite its problems with high Nyquist frequencies and under-sampling artifacts [58] it is widely used because of its rendering speed and interactivity [116]. Only recent versions of CT workstations like the Toshiba Vitrea®[*] and Siemens Syngo®[†] allow also interactive ray-based image generation like ray-casting, allowing a representation with less rendering artifacts. A daily life example using 3D texture slicing from the clinical practice for brain angiography investigation on a Toshiba CT system is shown in Figure 2.1.

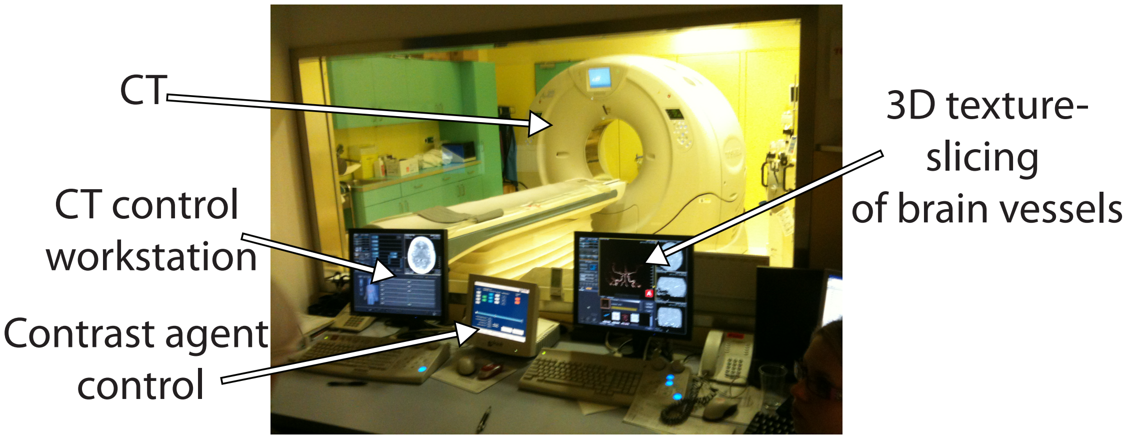


Figure 2.1: This picture shows a daily life example using 3D texture slicing for brain angiography at the University Hospital Graz. While the left workstation is used to control the CT, the right workstation pre-processes and reconstructs the scan with 3D texture slicing. Please note, that we cannot present close-ups of the workstations in this image because of confidential patient information.

[*] http://www.toshiba-medical.eu/en/Our-Product-Range/CT/Workstations/Vitrea-fX/

[†] www.medical.siemens.com

Functional brain analysis is one of the strongest arguments for observation 2 in Chapter 1. Proposed by Torrey *et al.* [223] and practically introduced by Moseley *et al.* [162], this method revealed as one of the most important tools for brain lesion analysis and brain tumor resection planning [20, 224], the study of multiple sclerosis [202] and some more diagnostic procedures [66, 140]. 3D image synthesis is essential for this method, even though often only geometric representations together with cutting planes trough multi planar reconstructions are used for fibrotic structures instead of direct volume rendering methods.

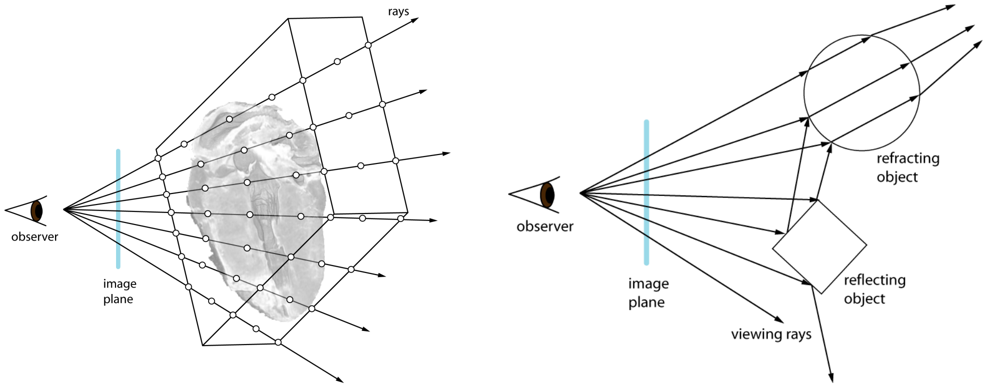
4D phase contrast (PC) MRI is used for the assessment of blood flow properties. This is crucial for the understanding and diagnosis of many cardiovascular diseases. The MRI through-plane phase contrast method provides a lot of useful information from flow through cross sections or velocities in preferred directions. However, its usefulness in situations involving complex fluid dynamics - as for example in the cardiac chambers - is limited, because the main directions of flow are neither known nor constant in time. Conceptually the easiest way to acquire three-dimensional blood flow data is to measure both through-plane and in-plane velocity components via MRI phase contrast sequences. Velocity vectors are determined on each imaging plane: In the case of the combined through-plane and in-plane measurement this happens for each pixel. A correlation between vascular diseases and hemodynamics has been demonstrated early by Falsetti *et al.* [64] and Hamlin *et al.* [174] and by several later contributions for reconstruction Computational Fluid Dynamics (CFD) [161]. Primarily systolic flow was studied in the last decade since the functional efficiency is the crucial factor for the pumping function of the heart [78]. Newest results from G. Reiter and U. Reiter showed that certain flow patterns in the right ventricular outflow tract correlate exactly for example with pulmonary hypertension [195] and that a DVR Line Integral convolution to visualize these flow patterns in 3D [105].

2.2 Ray-based image synthesis on the GPU

This thesis deals predominantly with the technical improvement of ray-based image synthesis directly from volumetrically scanned data. A lot of research has been done in this direction, at which *feasibility* can be directly related to *rendering speed*, as noted in Chapter 1. Besides algorithmic improvements of Direct Volume Rendering (for example early-ray termination [127, 149] or binary space partitioning [127, 129, 213]), the parallelization of ray-based image generation in general and the use of programmable GPUs contributed most to the rendering speed and interactivity [4, 246]. Hence, the following paragraphs concentrate on the research which has been done for using the parallelism of modern graphics hardware for ray generation and traversal with special attention on volumetric materials.

We introduce the term *ray-based image synthesis* in this section and break it down into its sub-algorithms and applications. In contrast to general 3D image synthesis, which describes all kinds of perspective projections, ray-based image synthesis can be seen as an umbrella term for all image synthesis algorithms which use the metaphor of rays which are *shot* through each pixel in the image plane. This concept is outlined in Figure 2.2 and examples are shown in Figure 2.3. Rays can either reflect, refract, scatter or accumulate translucent object interior in an arbitrary scene. Note that in computer graphics, other effects of light, which are not noticeable under normal conditions, are neglected (e.g. *diffraction*). It is not necessary to model those effects for highly realistic representations [107].

Ideally, the distance between the sampling positions along a ray are infinitely small. At each sampling position, infinitely many secondary rays might be *shot* further in the scene in all directions. A perfect scene further consists of infinitely many and infinitely small data points.

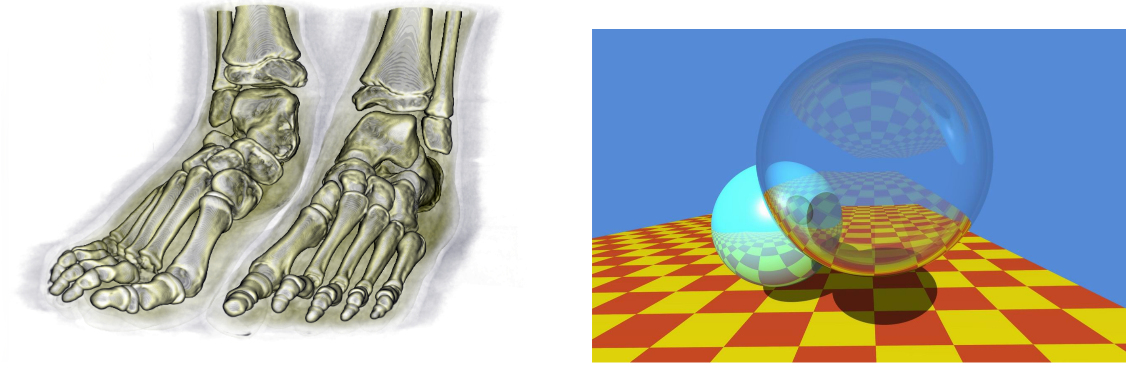


(a) Volumetric object.

(b) Geometric object with physical properties.

Figure 2.2: Ray-based image synthesis principle. Viewing rays are *shot* through every pixel in the image plane. Their virtual origin is defined by the eye position. In literature case (a) is called volumetric *ray-casting*, approximating a full integral along the ray with Riemann sums (Eq. 2.2) and case (b) *ray-tracing* defined by analytically calculated ray/geometry intersections (multiple secondary rays have not been considered in the illustration). Note that in (b) no volume sampling and value accumulation is usually done. Figure 2.3 shows practical examples for these methods.

Obviously, this ideal scene and ray propagation is neither realistic with limited computational power and memory resources nor practical. Hence, ray propagation stops at a certain level of accuracy with a sufficiently small number of secondary rays which pass discrete objects. The literature distinguishes further between the kind of traversed objects for ray-based approaches. On the one side *ray-tracing* is commonly referred to the photorealistic evaluation of refractions and reflections on/through geometric objects. On



(a) Volumetric object, ray-casting with one of our implementations.

(b) Geometric object with physical properties, ray-tracing with Nvidia's OptiX [183].

Figure 2.3: Ray-based image synthesis examples. (a) and (b) show simple practical examples for the methods presented in Figure 2.2. Both use ray-based image synthesis to generate the output. While in (a) voxel intensities are mapped to color values and accumulated along the rays, (b) uses analytically calculated intersections of the rays with the scene objects and accumulates all physical properties of the hit objects until a certain recursion depth is reached. Both are rendered in real-time.

the other side, the accumulation and mapping of values along a ray passing a volumetric object is referred to as *ray-casting*.

Like ray-casting, ray-tracing

> [...] determines the visibility of surfaces by tracing imaginary rays of light from viewer's eye to the object in the scene. [68]

Hence, the used term depends on the desired data input. This thesis targets medical application of ray-based image synthesis for volumetric image modalities. Because of the volumetric nature of the input data and the desired translucent representation, a rigid classification of this work would use predominantly the term *ray-casting*. However, ray-casting can be classified as a special implementation (subclass) of ray-tracing, including only primary rays with no (ray-tracing-) recursion as outlined in algorithm 1. Indeed, while ray-casting has also been used to determine the visibility of geometric structures, we follow the definitions of Levoy [126], who uses ray-casting for *volumetric objects* and Drebin and Max [56, 150], who have extended that approach to a translucent accumulation of mixtures of materials. This approach is outlined in algorithm 2 as pseudo code. Nevertheless, we often just use the more general term *ray-based image synthesis* in this thesis, because we will blur the boundary between volumetric objects and geometric properties in Chapter 3 and Chapter 4 by combining ray-casting with ray-tracing features.

To approximate an ideal ray distribution, ray-casting uses the *"Emission-Absorption Model"*, which allows particles to emit and absorb light, but omits scattering and indirect illumination effects. It offers a good trade-off between generality and computational efficiency. For this model, the following equation – *the volume-rendering integral* [150] – calculates the radiance of a ray traversing a volume at the ray's exit point:

$$I(D) = I_0 e^{-\int_{s_0}^{D} k(t)dt} + \int_{s_0}^{D} q(s) e^{-\int_{s}^{D} k(t)dt} ds \tag{2.1}$$

Thereby s describes the position along the ray, $s = s_0$ and $s = D$ denote the positions at which the ray enters and leaves the volume respectively. I_0 is the initial radiance of the ray at s_0 and $I(D)$ the radiance of the ray at position D, where it leaves the volume after all accumulation has been done. Moreover $k(t)$ and $q(s)$ represent the absorption and emission coefficient. The first term of the equation calculates how the initial radiance I_0 of a ray at s_0 is attenuated throughout the transition of the volume. The result stands for the contribution of I_0 to the radiance $I(D)$ which leaves the volume at D. The second term calculates the combined emission contributions of the single particles along the ray to its final radiance. Thereby the emission of each particle is attenuated by the remaining absorption coefficients along the ray, from location s of this particle to the point D where the ray leaves the volume.

Since it is not generally possible to evaluate the volume-rendering integral analytically, it must be approximated numerically. A common approximation of the volume-rendering integral is defined by a Riemann sum,

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j \tag{2.2}$$

by which the integration is split into n discrete equidistant intervals (intervals of different lengths are possible as well but are not considered here). Thereby T_j and c_i describe the transparency and color contribution of the i^{th} segment, which reaches from sample position s_{j-1} to s_j, respectively. They are approximated by $T_i \approx e^{-k(s_i)\Delta x}$ and $c_i \approx q(s_i)\Delta x$, where $\Delta x = (D - s_0)/n$ describes the length of the intervals. Moreover, $c_0 = I(s_0)$ holds, which means that c_0 stands for the radiance (color) of the ray at the entry point of the volume [60].

When evaluating rays, samples are usually not at the same position as voxels on the volume grid. As a result the continuous volume must be reconstructed in order to obtain data values of the samples along the ray. In general a successful reconstruction depends on two major factors: Firstly, the sampling frequency used to discretized the original volume must be high enough (at least Nyquist frequency) to allow a later reconstruction without a loss of information. Secondly, the actual data value at the sampling position has to be interpolated. In practice, trilinear interpolation is used because it is directly available in

Algorithm 1 A generic ray-tracing kernel in pseudo-code.

```
function traceRay(ray direction)
    get intersection with first object in ray direction
    $\alpha$ = get object ambient and diffuse color
    if object is reflective & maximum recursion level not reached then
        calculate reflective ray direction
        $\beta$ = traceRay(reflective ray)
    end if
    if object is refractive & maximum recursion level not reached then
        calculate refractive ray direction
        $\gamma$ = traceRay(calculate refractive ray)
    end if
    return combineColors($\alpha, \beta, \gamma$)
end function

function RayTracing
    for each pixel in image plane do
        generate ray in viewing direction
        traceRay(ray direction)
    end for
end function
```

Algorithm 2 A generic ray-casting kernel in pseudo-code.

```
lookup volume entry position
lookup volume exit position
compute ray of sight direction
while in volume do
    lookup data value at ray position          ▷ value interpolation
    apply transfer function to data value      ▷ Eq. 2.3
    accumulate color and opacity               ▷ Eq. 2.2
    advance along ray                          ▷ step size fulfills Nyquist criterion
end while
```

hardware and sufficiently accurate.

Today, all major ray-based image synthesis algorithms have been implemented on parallel stream processors – in most cases GPUs – to achieve real-time performance. However, the $O(n^3)$ complexity of algorithm 2 makes it still hard to extend these approaches with special functions while preserving real-time performance or a certain level of accuracy. Ray-based image synthesis nearly fully occupies all computational resources of a high-end state-of-the-art PC to fulfill these constraints. For example, a simple extension of Algorithm 2 to support multiple volumetric objects in the same scene might already result in low or absolutely no interactive frame rate. Therefore, the optimal use of system resources and optimal integration of algorithmic extensions is still an active topic of research and covered in this thesis.

2.2.1 Ray setup

Standard ray-tracing methods calculate all viewing rays starting from the pixels in the image plane and evaluate secondary rays recursively. While it is possible to extend algorithm 1 with a volumetric object case and integrate the while loop from Algorithm 2, current ray-casting algorithms require an additional ray setup step to determine the required accumulation direction through the volume. Algorithm 2 assumes known ray directions and volume entry and exit points. This would only be the case for a full integration of Algorithm 2 in Algorithm 1 (i.e. a ray-tracing engine) as mentioned above. However, due to performance reasons ray-casting rendering systems rely on a faster ray setup requiring multiple render passes. This setup scheme is outlined in the following. It supports in the simplest case, only scenes consisting of a single object.

Rays can be constructed by rendering a bounding geometry around the volume where its colors represent the volume entrance and exit coordinates. For the simplest case, the bounding box of the volumetric object can be used. Assigning primary colors to the corners in object space will result in continuously altered coordinates along the geometry's surfaces as outlined in Figure 2.4 (assumed that the render engine supports color interpolation). The necessary ray direction for each pixel is computed by subtracting the back face from the front. The required sampling length is defined by the coordinates between the front and back face. The sampling distance is defined by the Nyquist criterion to avoid undersampling and can be static for all rays. In practice this approach requires three render passes. The first frame contains the projected color-coded bounding geometry. The second frame the same geometry but with enabled front face culling (only back faces are rendered). The third pass uses frame one and two as lookup textures for the volume entry and exit position and execute Algorithm 2.

This standard method has been improved several times, for example by terminating the ray traversal if a certain opacity has been reached (early ray termination) [127, 149] or by using binary space partitioning [127, 129, 213] to skip empty space which is predominant in e.g. medical data sets.

2.2.2 Transfer functions

Transfer functions are essential for ray-casting, since they have a major impact on the output image. Transfer functions usually map scalar intensity values in volumetric datasets to optical properties (usually color and opacity). In general, a transfer function can be considered as a mapping from n data values, which are properties defining each sample s in a volume, to m optical properties:

$$c = \tau(s), \tau : \mathbb{R}^n \to \mathbb{R}^m \tag{2.3}$$

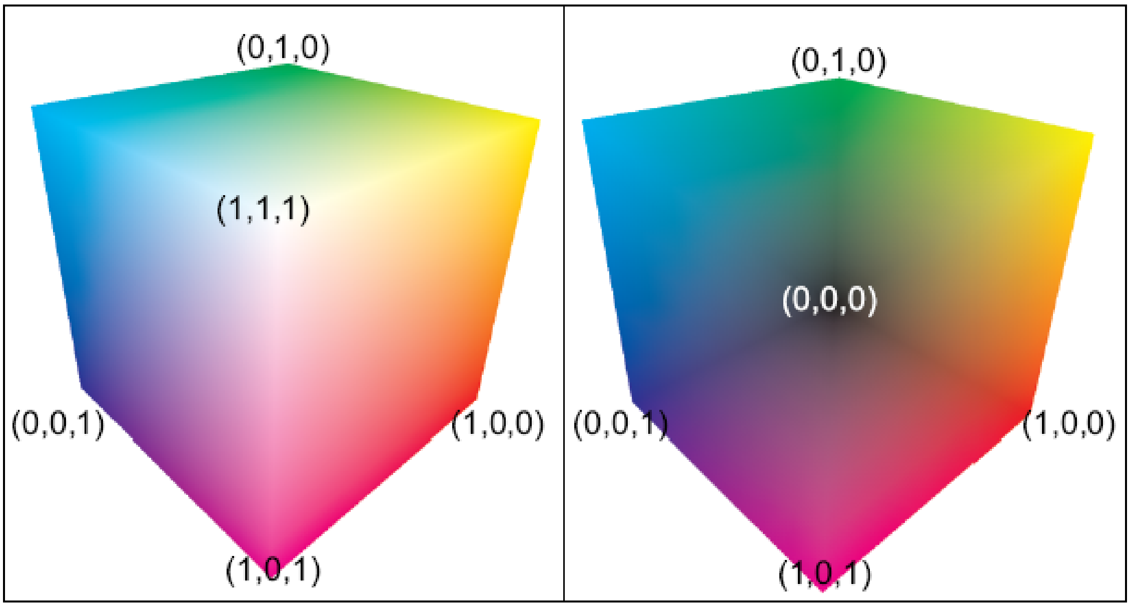


Figure 2.4: The front and the back faces of a bounding cube, which can be used to determine start and end points of rays. The coordinates are thereby encoded in the colors of the projection. ($xyz = RGB$)

Thereby τ represents the transfer function, s describes a sample for which the optical properties should be determined and c is a vector holding of the optical output properties. In this work we predominately use $m = 4$ and $n \geq 1$. Hence a transfer function describes a mapping from a set of n arbitrary data values to four optical properties stored in the vector c. Here c represents the red, green and blue channels representing the color as well as the opacity α and of the format $RGB\alpha$. In the simplest case, when the volume is only represented by a scalar field, a transfer function performs a mapping from intensity to $RGB\alpha$. These values are accumulated along the viewing rays by a Riemann sum as shown in Algorithm 2 and Equation 2.2.

2.2.3 Texture-based volume rendering

Using 3D textures for volume rendering is currently the most popular method for Direct Volume Rendering. In the meantime quasi all GPUs support 3D texture filtering in real-time, which makes the former approach of 2D texture slices as *proxy geometry* [32] more or less obsolete.

This technique renders 2D planes intersecting a 3D volume texture. Independent of the number of measured slices, a constant number of planar cuts are made through a 3D texture which contains the actual volumetric data. These slices are always directed towards the viewer and perpendicular to the viewing direction. Rotation of the volume will only affect the mapping of these slices - with the constraints of aliasing artifacts - whereas the 3D volume texture position is considered as fixed. Consequently, different viewing directions will result in different data mapped onto the 2D texture planes. A constant transparency value, for example

$$\alpha = \frac{1}{num_{slices}}$$

for each slice – will result in the same as the projection of the averages from the sampled values on a ray, starting from the eye point. Other transparency functions will allow arbitrary combinations of the evenly spaced sampling points of the volumetric texture. Figure 2.5 from [60, page 49] illustrates these principles.

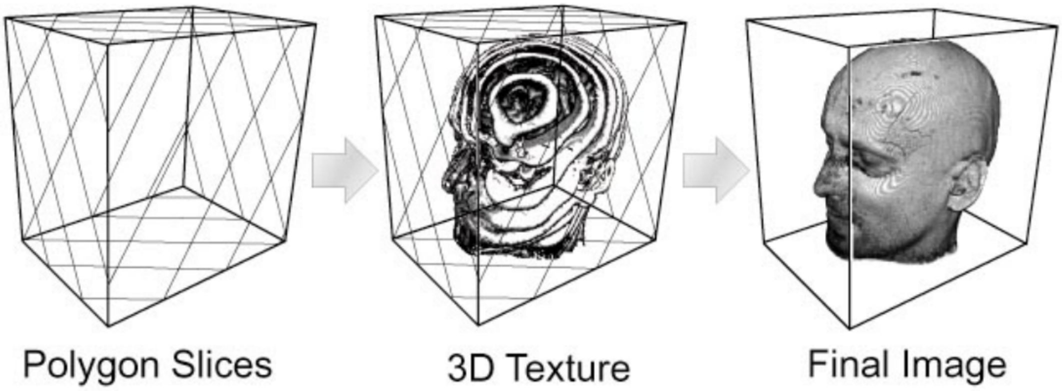


Figure 2.5: Generating semi-transparent slices always perpendicular to the viewing direction leads to an evenly spaced sampling of a fixed 3D-texture volume. This will provide the impression of a semi-transparent volume depending on the transparency distribution or more complex mapping functions [59].

Because of the computationally cheap nature of that approach and the resulting images of reasonable quality, this method is currently the most common in the clinical practice. However, this method often shows a noticeable transition during camera transforms and the static sampling distance per ray introduces problems with high Nyquist frequencies especially because of the non-constant sampling distances for different viewing rays as illustrated in Figure 2.6 during perspective projections. Ray-based image generation approaches are therefore considered as more accurate as texture-based volume rendering. It can be expected, that volume rendering based on any kind of textured planes will be regarded as outdated as soon as ray-based image generation reaches the same rendering speed and level of feasibility.

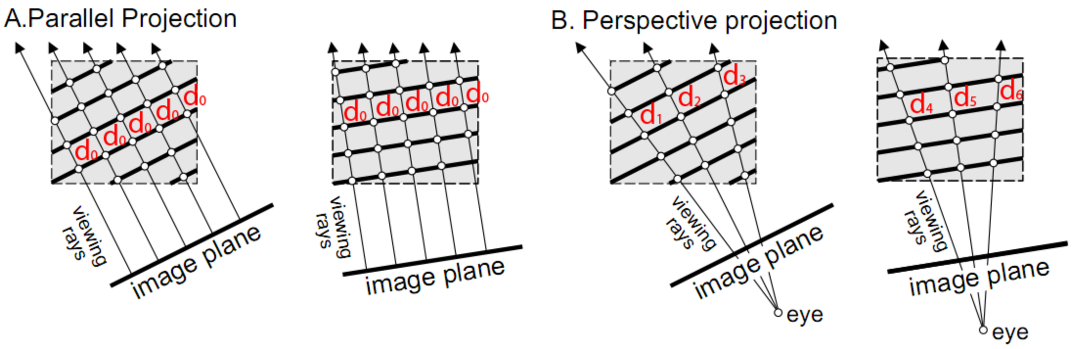


Figure 2.6: Non-constant sampling and varying sampling distance $d_{1,2,3,4,5}$ for perspective projection (B) of view-aligned slices compared to constant sampling positions d_0 during parallel projection (A) [59].

2.2.4 Volume rendering on the GPU

For a comprehensive overview of state-of-the-art volume rendering techniques we refer the interested reader to Engel *et al.* [60] and Weiskopf *et al.* [246]. While these two books describe basically shader-based volume visualization approaches, many recent methods use general purpose graphics hardware programming languages like Nvidia's *Compute Unified Device Architecture* CUDA [1] or OpenCL [112] or a combination with classical shader languages. Marsalek *et al.* [147] proposed first ray casting with front-to-back traversal, pre-integration, and early ray termination completely relying on CUDA. At the same time CUDA ray casting has been extended to volume rendering of unstructured grid data by Maximo *et al.* [152]. Compared to early shader-based approaches like Westermann *et al.* [249] who used the GPU only for first-hit computation with subsequent CPU ray-casting, new GPU APIs allow a fully flexible utilization of the available processing units. Therefore these novel methods often outperform the classical pure shader-based approaches in terms of performance and quality not only because of the quickly increasing computational power of GPUs.

GPU Computing Technologies The Cg language developed by Nvidia [143] allows the replacement of typical computations performed in the graphics pipeline by customized operations written in a C-like programming language. It makes use of the *stream programming model*, i.e., it is not possible to communicate with other instances of the program running at the same time or to store intermediate information for later use by subsequent invocations. Moreover, local memory is limited to a few hundred bytes, and there is no support for indirect addressing. This makes it impossible to use arrays with dynamic indices, stacks, and similar data structures. However, this computation scheme (which is free of data inter-dependences) allows a very efficient parallel execution of many data elements. Furthermore, Cg has full access to all features of the graphics hardware, including the texture addressing and interpolation units for access to 3D texture data. Similar functionality as in Cg is available in the *OpenGL Shading Language* (GLSL), which is not separately discussed here.

The *Compute Unified Device Architecture* (CUDA) by Nvidia [175, 179] is specifically designed for the use of graphics hardware for general-purpose numeric applications. As such, it also allows arbitrary read and write access to GPU memory and therefore seems to be the ideal candidate for our implementation. However, early versions of CUDA have lacked native support for 3D textures, so access to CT data would have to be rewritten as global memory access, which neither supports interpolation nor caching. Such a workaround would impose severe performance penalties, we therefore decided to use Cg since the intermediate storage problem can more easily be overcome as we will see in the remainder of this section. Today, CUDA supports 3D textures fully, which would also allow to implement the presented algorithms with comparable performance. However, the memory transfer overhead for an actual render pass is still not neglectable, which still argues for a direct shader-based implementation of this approach. A CUDA-based approach would currently also not allow a higher performance as given with the Cg-based AD implementation because of a still missing full C++ feature set.

Rendering on manycore GPUs Recent trends indicate that GPU programming for graphics is rapidly moving away from fixed-function approaches towards general compute languages executing on manycore architectures [100]. Rather than redesigning existing techniques to fit the highly constrained execution environment found in conventional GPU programming using shader languages, software rendering methods can become competitive again by executing them on manycore GPUs.

A noteworthy example is the Larrabee manycore architecture [206], which is based on multiple x86 cores and uses a flexible software renderer including a recursive polygon tiling algorithm [80]. This approach is highly optimized towards Larrabee's hardware capabilities, in particular its vector processing units, which are used, e.g., for on-the-fly coverage mask computation.

Developing on GPUs with CUDA is not quite as flexible as Larabee's execution environment, but allows similar results to be achieved using mainstream graphics hardware. For convenience, we review the main aspects that influence our work. CUDA executes kernels written in C in a massively multi-threaded fashion on the GPU's multiple processing cores. Threads are grouped into blocks, which are executed with round-robin scheduling on a multi-processor consisting of several single processors and a limited amount (16k) of fast shared memory, which acts as a cache. Threads also have access to the GPU's global memory at a slower speed. Fully utilizing all processors while avoiding memory bottlenecks requires careful design of algorithm and parameters.

A brief report on ray casting single volumes with CUDA is given in [147], concentrating on the best choice of execution parameters for basic ray casting. Single volume ray casting with moving least squares was considered in [123]. This method achieves very high quality reconstruction, but seems more appropriate for unstructured volumes. Ray casting of large deformable models composed of opaque polygons was presented in [185]. This work sorts all polygons into a view dependent 3D grid at every frame, and therefore focuses mainly on the sorting.

Rendering performance is always linked to the obtained image quality and volume sizes. Advanced systems need to consider multiple rendering nodes additionally to out-of-core and multi-resolution memory management to deal with current datasets. A sort-first rendering system was developed by Moloney *et al.* [160] for a single workstation with multiple GPUs for high frame rates with optimal load balancing. Sort-last parallel systems within a volume visualization context have been analyzed by several authors [158], [166] and [141] by using a cluster of commodity workstations equipped with programmable graphics hardware.

To the best of our knowledge, our work in Chapter 3 is the first multi-volume ray caster developed specifically for the manycore execution model represented by CUDA.

Multi resolution ray-casting Weiler et al. introduced multi-resolution ray-casting on the GPU [241] and described the benefits of mixing different resolutions and dealing with artifacts at the borders between different resolutions. An approach for online decompression of volume datasets using wavelet transformation is described in [87], where it is demonstrated that a few bits per voxel are sufficient for good rendering results.

Multi-frame rate volume rendering The concept of multi-frame rate rendering [209–211, 214] was introduced to cope with situations where the computation load exceeds the interactive rendering power of a parallel system. These situations occur when user interaction imposes real-time constraints on a system. Multi-frame rate rendering decouples display updates from image generation in a pipeline with asynchronous communication. The display update stage can guarantee fast frame rates and nearly latency-free responses to user interaction, while one or more GPUs in the back end stage can produce new high quality images at their own slower pace. Volume rendering has been included into a multi-frame rate system by Springer *et al.* [215], but they support static viewpoints only. To allow for viewpoint motion, image-based rendering can be employed. Existing image-based volume rendering approaches, such as pixel ray images [208] or light field representations [89, 196], produce rich descriptions of the volume, but they require considerable processing and are therefore less suitable for changing transfer functions interactively. The work from Hauswiesner et al. [95] alleviates these restrictions, and it describes a flexible, asynchronous rendering architecture for volumetric datasets.

GPU utilization optimization A GPU is a multi-processor system for which two scheduling strategies can be distinguished: space-sharing and time-sharing [39]. The latest generation of GPUs [180] can be classified as running a hybrid scheduler: multiple kernels can be run in parallel on different multi-processors (space-sharing), whereas a fine-grained 'warp' (thread block) scheduler [1] follows time-sharing principles. This lightweight and low-level scheduler is only responsible for the utilization of a single multi-processor and several improvements of this low level have been proposed by Fung *et al.* [70].

One way of incorporating scheduling ideas at a higher level is to use the CPU as a synchronization point between dependent execution steps, as implemented by the BSGP [100] model. Although this approach eases the work of mapping complex algorithms to GPU code, switching between the GPU and CPU imposes a severe overhead, leaving the GPU's utilization low. The concept of a so-called *"mega kernel"*, can be used to leave the control flow at the GPU, while switching between dependent tasks arbitrarily. This idea already exists in current systems, for example, RenderAnts [253] or OptiX [183]. In this case, the dependencies between the task/pipeline stages need to be dealt with explicitly, for instance by work queues. This idea can be found in the RenderAnts system and in a General Runtime/Architecture for Multicore Parallel Systems (GRAMPS) [217]. The GRAMPS architecture is an abstract rendering architecture that allows the construction of an arbitrarily connected directed graph, in which the nodes are connected by queues. The architecture is general, but it is only implemented as a simulator and is not tested on real hardware. Although GRAMPS introduces a multi-layer scheduler, its scheduler is mainly concerned with keeping queue lengths short, and it has little insight concerning the details of the task execution. The low-level scheduler found in the RenderAnts system is also concerned exclusively with queue management. RenderAnts also includes a higher level scheduler, which supports load balancing between multiple GPUs by work stealing of rather homogeneous workloads.

The most evolved approach dealing with GPU scheduling is the Nvidia OptiX framework [183]. A fixed priority-based scheduler for a mega kernel execution is used to keep the number of diverging branches low on each multi processor. A fixed size stack is used to support recursion within the mega kernel. Load balancing between different GPUs is dealt with by appropriately filling work queues on each GPU. Each execution unit draws elements from this global GPU work queue to fill a local work queue.

Multi-volume rendering Rendering of multiple volumes is a recurring topic of research. While multi-volume rendering was previously limited to static images (for example [167]), CPU-based rendering has relatively recently achieved real-time frame rates through the use of cache-coherent, multi-threaded strategies [85]. However, the processing power of CPUs has been overtaken by GPUs, which have more parallel execution units. Volume rendering is now commonly based on representing regular volumetric models as 3D textures in the GPU memory, and casting rays in the fragment shader. The setup of rays is done through rasterization of the volume's bounding geometry [117, 200]. With the advent of branching and looping in the shader, a ray can now be traversed inside the shader rather than through multi-pass rendering [216].

Through the use of depth peeling [62], handling of more complex scenarios becomes possible. In the context of GPU-based volume rendering, depth peeling has been used to clip volumes [247] and to intersect individual volumes with geometry [22, 221]. Recently multi-volume rendering has been combined with depth peeling and dynamic shader generation into a very efficient framework [199]. Depth peeling is used on the bounding geometries of multiple volumes to perform what are essentially CSG operations, and every distinct volumetric area is handled with optimized shader code that is derived from an abstract representation. A similar approach combining depth peeling and dynamic shader generation is taken in [188]. The work in [24] is unique in that it is able to combine multiple volumes with arbitrary layers of translucent, concave geometry.

Fragment processing Depth peeling is a widely used multi-pass technique that relies on manipulation of the z-buffer to progressively reveal layers of occluded geometry. Generally speaking, multi-pass techniques are methods used to overcome the limitations on the resources of the GPU, in particular in terms of available storage. Given unlimited memory, an approach like the A-buffer [35] can store all fragments in a list sorted by depth. Such an implementation is not really feasible in hardware, and therefore various approaches operating with bounded memory rely on multiple passes. Achieving effects involving multiple fragments often requires sorting of fragments by depth. Hardware-accelerated depth peeling [62] is such a method, but requires N rendering passes of N objects, leading to undesirable $O(N^2)$ complexity. Several pieces of work try to at least reduce this complexity by a constant factor, by making use of additional memory. This can take the shape of specific extensions of the fragment stream processing model of the GPU [2, 145] or by relying on existing extended framebuffer capabilities [15, 16, 33, 131]. Some work combines such an approach with an approximate pre-sorting on the CPU, in order to approach linear time behavior [33, 36]. However, such a pre-sorting is not feasible for general scenes. In contrast, our approach avoids conventional buffers and does not require pre-sorting for handling very complex models.

A common acceleration technique for polygon rasterization are coverage masks [67], which encode the pixel-wise inside condition with respect to a half plane in any possible orientation (typically for 8×8 pixel squares). Arbitrary convex polygons (and triangles in particular) can be processed by computing the intersection of the half planes associated with the edges, which is equivalent to a bit-wise AND operation of the corresponding coverage masks. In the literature, there are examples both for precomputed masks [67, 80] and for on-the-fly computation [3, 63, 206].

2.2.5 Perceptional considerations for ray-based image synthesis

Previous researchers have been concerned with the real-time performance of ray-based image generation algorithms. Recent work has introduced the exploitation of modern GPUs for solving the brute-force full-resolution ray traversal interactively, while coarse adaptive and progressive sampling approaches have been discussed since ray-tracing algorithms first became available. We give a brief overview of recent GPU methods and adaptive progressive rendering methods in Section 2.2.5.1 and discuss possible scene feature computation strategies in Section 2.2.5.2. A further overview of the reconstruction techniques for sparsely sampled data is given in Section 2.2.5.3, and the attempts to guarantee a minimal frame rate are outlined in Section 2.2.5.4.

2.2.5.1 Interactive ray-based rendering

Exploiting the GPU: Numerous rendering engines have been developed to deal with one of the most computationally expensive problems of computer graphics: ray-tracing. Besides CPU-based libraries [182, 237], most recent GPU approaches reach remarkable frame rates for low- to medium-complexity scenes [184, 206] in full quality. However, screen filling scenes or scenes with high complexity are still too slow for hard real-time constraints.Furthermore, rendering algorithms that aim at achieving real-time performance for the full-quality ray-casting of volume data use empty-space skipping [129], iso-surface ray-casting [236, 238], ray pre-integration [58], homogeneous region encoding [69] and many kinds of direct GPU implementations [65, Chapter 39].

Adaptive progressive rendering: Adaptive approaches, such as the one presented in Chapter 4, aim instead for the best possible trade-off between interactive frame rates and the loss of image quality instead of finding the maximum achievable frame rate for a full quality image. Finding this trade-off is still an ill-defined problem because the perception of quality differs between human beings and between applications. However, several algorithms exist to accelerate rendering speeds through ray reduction. The simplest method is a regular sub-sampling with a nearest neighbor interpolation. As discussed by [157] and still used in many interactive ray-based rendering systems [204, 235], this method is prone to strongly perceivable aliasing artifacts during the interaction. To deal with this problem, most related work has investigated the impact of different sampling pattern strategies in image space [55, 178, 181]. The sampling pattern is usually visually noticeably refined over time until a desired quality level is reached.

Levoy reformulated the front-to-back image order volume rendering algorithm to use an adaptive termination of ray-tracing [128]. The subdivision and refinement process is based on an ε threshold and does not consider human feature perception and temporal coherence. Later work altered the ray termination criteria [49] depending on the required rendering time or used texture-based level of detail [242], topology guided downsampling [115] or multiple resolutions of the same dataset [21, 120].

2.2.5.2 Important image areas

The choice of a suitable sampling pattern is crucial for adaptive rendering. For non-trivial systems, the pattern refinement strategy is usually chosen depending on features that are prominent for the human visual system. In the following paragraphs, we discuss our selected methods to find those regions.

Image space methods: Most methods refine the sampling pattern based on image intensity variances. Image areas with high frequencies require a denser sampling than large uniform areas do [124, 181] to gain a visually acceptable result.

During the development of our algorithm, we also experimented with the extraction of features from an image space by using a visual saliency-based approach [102].The saliency of an image is usually defined as a measure of how much a particular location contrasts with its surroundings in dimensions such as color, orientation, motion, and depth. The resulting feature classifications are very similar to the object-space results but at a lower performance and accuracy, and they must be computed for every frame. For these reasons, we first chose the object-space approach of analyzing meshes. In Section 4.8.1, we discuss the incorporation of saliency computations for scenes that show mainly textured objects or hard shadows and strong reflections. One main problem is that for an optimal result, the final image would be needed at full resolution before the scene is rendered. However, a full resolution image is only available from previous frames, which can be used to generate a spatiotemporal gradient cross-hair as introduced by [51]. This approach introduces severe spatial and temporal noise. However, the core of our method does not require any temporal coherence.

Non-Photorealistic Rendering (NPR) of line features: NPR deals with salient object features, often directly in object space. Related rendering techniques are mainly used for illustrative rendering and in cognitive science. Cole *et al.* [47], for example, show that object contours including the object silhouette are also used by artists to outline scenes. Several visualization algorithms show that these features can be used to simplify complex scenes for a better understanding of the essential parts [27, 31, 52].

The features of an object or a scene can be extracted in various ways: meshes can be analyzed in object space or, after rendering, in image space. The same method applies to volumetric data sets, where the object space contains a voxel grid instead of a set of geometric primitives. Finding features in a rendered image has the advantage of including textures and other effects, while in object space, more accuracy is usually available because the data have not been discretized into pixels. Moreover, the features in object space may be view-independent, which allows their reuse without recomputation. Figure 2.7 gives a visual impression of some sparse object features that we evaluate for ray decimation in this work. Our definitions of object features are similar to those from [52].

2.2.5.3 Sparse data reconstruction

Computing only rays for important areas means that the final image has to be reconstructed from those sparse samples. Numerous approaches exist besides the simplest,

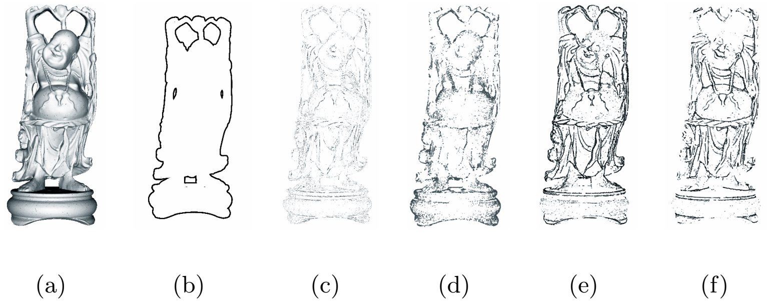


Figure 2.7: The happy Buddha object (a) rendered with different sparse line features. Silhouettes (b), suggestive contours (c), suggestive highlights (d), ridges (e) and valleys (f) are evaluated for ray decimation in this work.

conventional approach of regular subsampling. This attempt requires a nearest neighbor computation or a linear interpolation and leads to perceivable block artifacts. A good general overview of non-homogeneously sampled data reconstruction is given in [8].

Specialized approaches for computer graphics can be found for point-based rendering. The widely used *pull-push algorithm* [79] utilizes a pyramid algorithm for surface reconstruction. It has been adapted for the image-space reconstruction of under-sampled point-based surfaces by [86]. [186] extended this approach to fill the holes between splats. Unfortunately, these approaches are not suited for direct GPU implementations, as stated by [146]. The approach of [146], who proposed a GPU implementation for large point-based models with elliptic box-filters and deferred shading, is also applicable to the reconstruction problem in this work. However, its computational overhead is still higher for large viewports than that of the method presented in Section 4.3.4.

2.2.5.4 Guaranteed frame rates

To the best of our knowledge, our method is the first that successfully implements an algorithm to guarantee a certain minimal frame rate and still maintains an acceptable image quality for ray-based image generation. [192] proposed that a guaranteed frame generation time would be essential for mixed reality TV studio ray tracing applications, but they did not implement such an approach. For non-ray-based rendering approaches, a few systems that guarantee a certain frame rate exist. For example, [104] replaces complex objects optimally by impostors. These examples show that several applications require guaranteed frame rates. However, this problem is not well researched.

2.2.6 Accuracy and speed improvements of ray-based image synthesis by analytical methods

Accuracy is besides computation speed one of the fundamental requirements for medical applications. Especially when a biopsy needle or interventional tools needs to be placed exactly at a designated position, the accuracy of the underlying navigation method can decide about the success of the procedure. Since a patient is not a rigid system which would show no changes in position, orientation or other degrees of freedom, medical intervention requires permanent alignment of all data in a common coordinate frame in the optimal case. Such an alignment is called registration. A common way to monitor a patient during an intervention is fluorscopy. This technique produces 2D X-Ray projection of the patient, which can be registered to projections of three-dimensional planning scans. This registration step however requires as much accuracy as possible to find a planned position again. Normally, this registration is based on optimization, which often relys on numerical approximations of value gradients. Numerical approximation is limited considering its accuracy because discrete differentiation positions and the supported bit-depth of the host system simply do not allow a perfect result. An analytical calculation of gradients is therefore desirable. Such analytical results can be obtained by differentiation of the program code directly. Tools for such tasks exist and their core methods are called Automatic differentiation (AD).

Automatic differentiation (AD) is a mathematical concept with increasing relevance to natural sciences in the last twenty years. Since differentiating algorithms is, in principle, equivalent to applying the chain rule of differential calculus [82], the theoretic fundamentals of automatic differentiation are long-established. However, only recent progress in the field of computer science allows us to widely exploit its capabilities which was first observed by Griewank *et al.* [83].

One of the major results of Griewank's research is the *source transformation* method [81]. Source transformation takes as input the source code of an algorithm which computes some function $y = f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, and outputs the source code of an algorithm which computes $\nabla f(\mathbf{x})$, i.e., the gradient of $f(\mathbf{x})$ with respect to its n parameters.

Another approach is operator overloading. Prominent examples of AD tools performing source transformation include "Automatic Differentiation of Fortran" (ADIFOR) [18], "Transformation of Algorithms in Fortran" (TAF) [72], and "Tapenade" [94]. The best-known AD tool implementing the operator overloading approach is "Automatic Differentiation by Overloading in C++" (ADOL-C) [84].

Previous approaches cannot efficiently handle the amount of image and volume data used in our case, or make simplifying assumptions which compromise accuracy (such as the volume gradient method [244], which is reported to be faster, but less accurate, than methods taking into account the raw volume data).

As already mentioned these AD methods can be used as input for pose registration optimizers. However, an optimizer like this requires also similar input for which at least the meaning of scalar values should be comparable. Such input is for example given with similar sources like, *e.g.*, X-Ray fluorcopy and CT. However, to generate a for the optimization with fluorcopy feasible projection from CT, further methods like digitally reconstructed radiograph (DDR) computation are required. Registration methods also often require a rough basic alignment which is normally achieved by registration targets. We give a short overview over the previous work which has been done for these kind of registrations problems with their registration targets and DDR computation in the following.

Pose Estimation/Registration Initialization Navab et al. [172] proposed a registration target with code words located side by side, formed by visible beads attached to a ring of acrylic glass to solve the C-Arm tracking problem for cerebral angiography. In contrast to this approach, our underlying clinical application does not allow a field of view containing a lot of beads up to a complete ring target. Therefore one of our requirements is to see only a very small part of the target and consequently no complete code words. Furthermore, we need a code which allows a certain robustness against occlusions caused by medical instruments. Finally we take pelvis images in lithotomy position for which a ring is not feasible. A ring it would be too large to deal with all types of patients and therefore sometimes too close to the X-ray source or too large for the bending range of the C-Arm.

Jain et al. [103] proposed a system to estimate the relative position of the C-Arm between two images from stationary natural features. However, X-Ray registration to a virtual patient model requires to know the camera position for **each** image relative to a **fixed origin**. Furthermore, the extraction of natural features is a time-consuming procedure. This would not be feasible for online processing of the C-Arm poses.

Digitally Reconstructed Radiograph (DRR) Computation Efficient DRR generation has recently been an active area of research. Due to the inherent parallelism in this task, a significant speedup can be achieved by the use of *graphics processing units* (GPUs). Early approaches of this kind were based on texture slicing [121], while more recent techniques make use of 3D texture hardware [76].

The generation of DRRs by the GPU was proposed by Khamene et al. [108], Kubias et al. [119] implemented the similarity measure on the GPU and investigated the effects of electronic post-processing of X-ray images. Khamene et al. [109] proposed a GPU-based registration method, performing a low-dimensional search in each of the projections.

X-Ray/CT Registration Several methods exist which demonstrate the feasibility of X-ray/CT registration [222, 244]. A highly parallel approach to the optimization problem was proposed by Wein et al. [244]. They perform a *best neighbor* search in any of the six degrees of freedom, i.e., they require 12 volume traversals per iteration, where each traversal is done on a separate processor.

Koehn et al. [110] presented a method to perform 2D/2D and 3D/3D registration on the GPU using a symbolically calculated gradient. However, they do not deal with the 2D/3D case (i.e., no DRR was involved in their method). Moreover, they manually compute the derivatives, which restricts their approach to very simple similarity measures such as the sum-of-squared differences (SSD) metric.

PART II - TECHNOLOGY

Chapter 3

Real-time multi-volume rendering on manycore GPUs

Direct volume visualization requires the ray casting algorithm in its base form (see Algorithm 2 in Section 2.2) and its corresponding ray setup (*cf.* Figure 2.4) which does not scale efficiently enough with the number of input volumes. Currently this problem prevents the use of multi-variate and multi-dimensional datasets in the clinical practice. To overcome these shortcomings and to attack *AOH4* from Chapter 1 we present a new GPU-based rendering system for ray casting of multiple volumes. Our approach supports a large number of volumes, complex translucent and concave polyhedral objects as well as CSG intersections of volumes and geometry in any combination. The system (including the rasterization stage) is implemented entirely in CUDA, which allows full control of the memory hierarchy, in particular access to high bandwidth and low latency shared memory. High depth complexity, which is problematic for conventional approaches based on depth peeling, can be handled successfully. As far as we know, our approach is the first framework for multi-volume rendering which provides interactive frame rates when concurrently rendering more than 50 arbitrarily overlapping volumes on current graphics hardware.

3.1 Problems with state-of-the-art ray casting algorithms

Ray casting has prevailed as the most versatile approach for direct volume rendering because of its flexibility in producing high quality images of medical datasets, industrial scans and environmental effects [60]. The large amount of homogeneous data contained in a volumetric model lends itself very well to parallelization. Today even a commodity graphics processing unit (GPU) is capable of ray casting a single high resolution volumetric dataset at a high frame rate using hardware-accelerated shaders.

However, rendering a single homogeneous volume is not sufficient for more advanced applications. Simultaneous rendering of multiple volumes is necessary when several datasets have been acquired. For example, in medical imaging complementary techniques such as anatomical image acquisition methods (e.g., computed tomography (CT), magnetic resonance imaging (MRI), ultrasound) and functional image acquisition methods (e.g., positron emission tomography (PET), single photon emission computed tomography, and functional MRI) can be used to examine a patient. Moreover, models of medical tools and polyhedral segmentation results in the correct geometric context are essential for computer aided surgery. Currently available methods have limited capabilities for interactive investigation of multi-modal volumetric data enhanced by polygonal models. The demand for sophisticated visualization of volume and surface data is, however, not unique to medical applications. Many other real-time graphics applications such as computer games and CAD are currently restricted to opaque surface mesh rendering and can not utilize the high visual potential of volumetric effects.

A special case of multi-volume rendering are exploded views, where multiple instances of a volume are rendered with individual clipping and transformation parameters applied to yield illustrative visualizations [28]. Multi-volume rendering is harder than rendering single volumes, because it requires handling of intersections and per-sample intermixing. Resampling the volumes to a single coordinate frame is not desirable because of the resulting loss in quality (or increased memory requirements) and limited flexibility of layouting. Another requirement of advanced applications is the combination of volumes with polygonal meshes. Polygonal models are useful for embedding foreign objects or reference grids into volumes, but also to apply clipping shapes, highlight areas of interest or display segmented geometry. We also note that the intersection of the bounding boxes of multiple volumes is given as a polyhedron. Such an intersection of bounding boxes is useful as a conservative approximation of the region for which intermixing of the shading contributions of intersecting volumes is required. For expressiveness and convenience, support for polygonal structures should not be limited to convex shapes or opaque materials.

Recent GPU accelerated multiple volume renderers [24, 199] perform ray casting in a fragment shader and rely on a depth peeling [62] approach to identify homogeneous ray intervals. Depth peeling is a highly redundant multi-pass technique, which repeatedly rasterizes the same geometry to sort all possible depth samples. Consequently, a ray caster based on depth peeling does not scale to a larger number of volumes, since depth peeling of a larger number of bounding boxes or of complex polygonal meshes quickly becomes the dominant performance factor.

The use of depth peeling is implied by the use of conventional shading languages, which represent an abstraction of a graphics hardware pipeline with fixed function portions. In a conventional shading language, the programmer has limited control over the information passed from the vertex shader to the fragment shader stage. Often an efficient sort-middle approach [159] is performed by the graphics hardware, but the strategy by which fragments are re-assigned from vertex shaders to fragment shaders is not exposed to the programmer.

It is therefore not possible to collect all depth samples obtained from rasterizing the complete bounding geometry or polygonal meshes and pass the sample collection on to a fragment shader to perform depth sorting and ray casting together in the natural order.

Instead, (repeated) rasterization and ray casting of segments are interleaved. The separate shaders for depth peeling and ray casting communicate via global GPU memory, which is orders of magnitude slower than the local memory of the GPU cores. Scalability of this approach is therefore ultimately limited by memory bandwidth.

In this Chapter, we investigate an approach suitable for current manycore GPUs, which overcomes the inefficiencies imposed by a fixed function pipeline. We employ the new Compute Unified Device Architecture (CUDA) [175, 179], a C-like language for general purpose computations on the GPU. Similarly to the OpenGL renderer described by [206], we implement a volume rendering pipeline based on polygon tiling entirely in software. This approach has complete control over the rendering process and executes an efficient sort-middle approach. Geometry is rasterized only once, and all depth samples are passed on to ray casting through on-chip shared memory, which is also significantly faster than global memory. Ray casting is coherently executed in tiles of 8×8 pixels with straight rays. Both geometry and fragment processing (ray casting) are executed in a massively parallel way using CUDA's multi-threaded execution model.

The efficiency gained by this approach allows us to support a very general multi-volume data model, thereby unifying a number of advanced volume rendering approaches. The model consists of a recursively defined constructive solid geometry (CSG) structure composed of arbitrary volumetric polyhedra, i.e., two-manifold (possibly concave) triangular meshes with volumetric texturing. Every polyhedron is associated with a volumetric 3D texture, a transfer function and a geometric transformation. This structure is similar to a volume scene graph approach [167] and supports any combination of the following use cases:

- Multiple intersecting volumes with individual coordinate systems and resolutions. CSG operations are used to distinguish overlapping and non-overlapping areas.

- Volumes can have arbitrary two-manifold bounding or clipping geometry. Concave polyhedra can conveniently be used as tight fitting bounding geometry. Selected areas in a volume can be highlighted interactively.

- Volumes can intersect and be intersected with transparent, concave geometry.

- Polygonal models can have volumetric effects, in particular they can be made of transparent homogeneous material (not just transparent surface rendering).

- CSG operations on a volume segmented into multiple regions with polyhedral boundaries can be used to assign individual transfer functions to each region. Each region can be transformed individually, allowing exploded views. Multiple instances of a region are possible for side by side comparison.

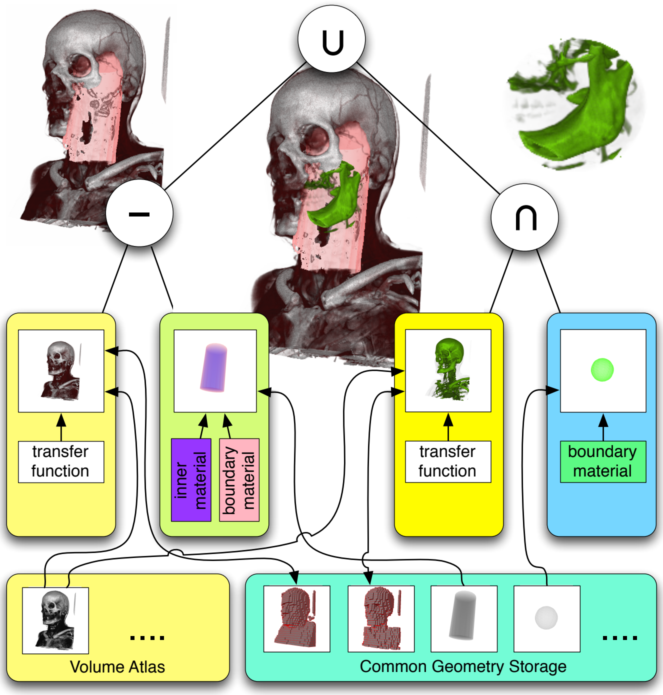


Figure 3.1: The scene graph consists of CSG operations and polyhedral objects with volumetric textures and/or material properties. Textures are stored in a texture atlas, polygonal geometry and volume boundaries in common storage, which allows for instancing. Note that the two bounding volumes of the head in the geometry storage are computed from different transfer functions.

3.2 Polyhedral rendering system overview

Our system is able to produce ray casting images of complex intersecting volumetric datasets with polyhedral boundaries. All computation executes on the GPU, the only information that is received from the CPU for every frame is the camera position. The scene that represents the input to the ray casting procedure is structured in the following way (see Figure 3.1):

- Raw volume data from multiple volumes is represented in a volume atlas.
- Polyhedral objects consist of a triangular mesh. A polyhedral object can be concave, but must be two-manifold, so that interior and exterior can be distinguished.
- A scene consists of a simple scene graph. Interior nodes are associated with CSG operations, whereas leafs refer to a polyhedral object as the boundary, a volume texture or single material property for the interior, and an affine transformation for global placement of the object. The transformations can also be animated. Instances can be created by referencing the same polyhedral object and/or volume texture more than once.

The ray casting algorithm follows the usual steps of a rendering pipeline: transformation of polygons, rasterization, and fragment processing. The latter is done in a pixel-parallel way per 8×8 tile and is responsible for traversing adjacent rays through the volumes stored as 3D textures. It therefore naturally exploits texture cache coherence.

Unlike depth peeling approaches, our rendering system traverses the scene only once. We implement all steps in CUDA software and therefore have access to all intermediate results of the pipeline, which are kept in shared memory wherever possible for maximum performance. We still need two separate kernels (one for triangle processing and one for fragment processing) since the ray caster requires a sorted list of all relevant fragments at each pixel, which is only guaranteed to be complete after processing all triangles.

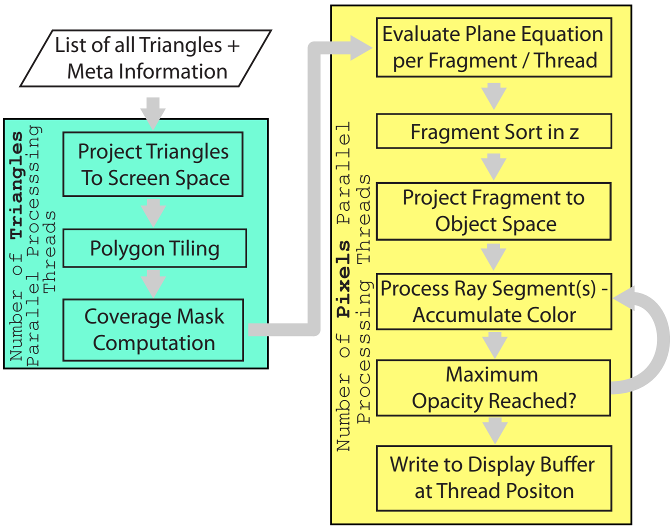
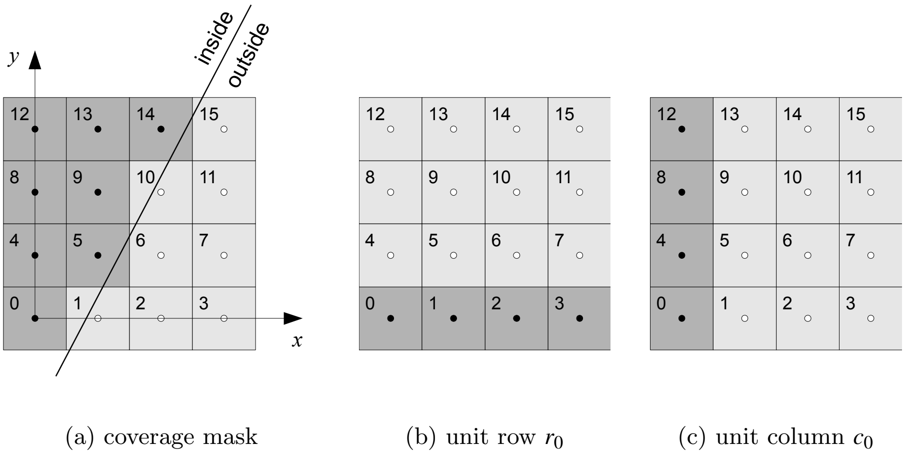

List of all Triangles + Meta Information
Number of Triangles Parallel Processing Threads
Project Triangles To Screen Space
Polygon Tiling
Coverage Mask Computation
Number of Pixels Parallel Processing Threads
Evaluate Plane Equation per Fragment / Thread
Fragment Sort in z
Project Fragment to Object Space
Process Ray Segment(s) - Accumulate Color
Maximum Opacity Reached?
Write to Display Buffer at Thread Positon

Figure 3.2: The flow-chart shows an overview of our rendering system. The main steps - triangle processing and pixel processing - are executed in separate compute kernels.

The triangle kernel is responsible for geometric transformations, assignment of triangles to viewport tiles, and computation of coverage masks (see Section 3.2.1). The pixel kernel receives the fragments covering a pixel in arbitrary order. They must be sorted in z-direction before they can be used to split a ray into homogeneous segments (see Section 3.2.2). The final ray casting and shading step requires the blending of several intersecting volumetric objects and therefore the efficient accumulation of the individual volumes' contributions along the rays (see Section 3.2.3). Support for concave bounding geometry simplifies empty space skipping (see Section 3.2.4). Volume samples are taken from a volume texture atlas (see Section 3.2.5). The following sections describe details of the individual steps of the system. For an overview see Figure 3.2.



(a) coverage mask (b) unit row r_0 (c) unit column c_0

Figure 3.3: On-the-fly half plane inclusion test for a 4×4 pixels square. The coverage mask (a) is composed of shifted copies of the unit row $r_0 = 0\text{x}000\text{F}$ (b) or the unit column $c_0 = 0\text{x}1111$ (c). The numbers refer to bit positions in the coverage mask.

3.2.1 Triangle rasterization

In conventional rendering of scenes dominated by opaque objects, high performance is related to being able to identify occluded portions of the scene early in the pipeline. In contrast, our volume rendering must unconditionally rasterize all polygons and pass information on the screen coverage of a polygon as quickly as possible to the pixel processing. After transforming a triangle into screen space, for each tile of 8×8 pixels intersecting the triangle's bounding box, a 64-bit coverage mask [67] is computed. This simple strategy maps well to CUDA hardware due to its low resource requirements and outperforms more sophisticated approaches (such as [206]) for typical scenes.

It is essential for our approach that every fragment covered by the triangulated volume boundary is visited exactly once since we initialize the ray segments for our ray caster at the rasterized surface fragments. Coverage masks are well suited for this purpose due to the simple bit-wise AND operation of the contributions of the corresponding half planes. However, precomputed coverage masks can not be used in this case since a particular mask is selected from the lookup table based on a discretized orientation of the half plane, which might result in a few misclassified pixels. We therefore compute the covered pixels for each triangle on-the-fly. Since the GPU cores in our target platform (NVIDIA GT200) are scalar processors, the method proposed by [206] is not favorable in our case. Instead, we make use of bit shift operations to compute one row (or column) of the $N \times N$ pixel coverage mask in constant time.

Consider the example in Figure 3.3(a). The outside half plane is defined by $ax+by+c > 0$. A bit in the mask is set to one (dark squares in Figure 3.3(a)) if the corresponding pixel center (filled circle) is located inside the half plane, and otherwise set to zero. For plane coefficients $a > 0$ and $|a| > |b|$ (such as in this example), we compute the number $n_1(y)$ of "one" bits in a given row $y \in \mathbb{Z}$, which equals the number of corresponding pixel centers to the left of the line separating the inside and outside half planes:

$$x = -\frac{by+c}{a}, \quad n_1(y) = \mathrm{clamp}(0, \lceil x \rceil, N), \tag{3.1}$$

where $\mathrm{clamp}(i, j, k) = \max(i, \min(j, k))$. The bit mask $r(y)$ of a single row y can be obtained in constant time from the unit row mask r_0 (Figure 3.3(b)) by means of two bit shift operations:

$$r(y) = (r_0 >> [N - n_1(y)]) << [N \cdot y]), \tag{3.2}$$

where ">>" and "<<" denote the bit shift operation to the right and left, respectively. The entire coverage mask m is then

$$m = \sum_{y=0}^{N-1} r(y). \tag{3.3}$$

For $|a| < |b|$ we proceed in a similar way column-by-column, using the unit column mask c_0 in Figure 3.3(c) instead. The case $a \leq 0$ is handled by symmetry and bit-wise inversion.

Note that all constants in equations (3.1) and (3.2) can be precomputed, such that equation (3.3) can handle both the row-by-row and column-by-column case without conditional expressions to distinguish them. It is therefore possible to process triangles with arbitrary edge orientation in parallel in a SIMD-efficient way. On current NVIDIA hardware, this method is four times faster than a straightforward per-pixel computation of the mask as proposed by [206].

The triangle kernel's completion ensures an application wide synchronization point before entering the pixel kernel. However, communication from the triangle kernel to the fragment kernel must use relatively slow global memory. Similar to the workflow presented by [3], we carry out computation of the coverage masks in the triangle kernel and not in the pixel kernel, because communicating a 64-bit coverage mask is cheaper than a full triangle description with 3 screen-transformed vertices.

Every triangle record contains the coverage mask and the id of the contributing triangle. No additional per-fragment information is stored in global memory, which makes the communication significantly more efficient than conventional buffer strategies [15, 145]. The triangle records are organized in chunks of 64 records each, which are drawn from a pre-allocated pool of memory. When a tile receives its first triangle (or the previous chunk is fully occupied), a globally unique 32-bit chunk index is appended to the list of chunk indices associated with this tile. We choose a maximum of 64 chunks per tile, which allows for a total number of 4096 triangles per tile. Parallel creation of this data structure is synchronized among the worker threads of the triangle kernel using atomic functions.

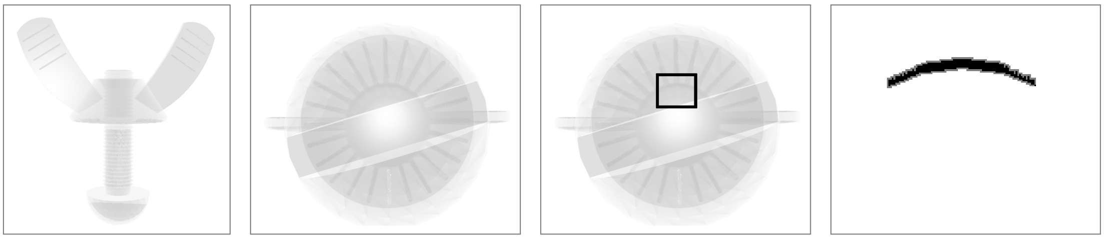
3.2.2 Depth sorting

A block of threads responsible for a tile consists of 64 threads, one for each pixel in the tile. A thread's first task is to build a representation of the ray, consisting of the intersections with the triangles, sorted by depth. The thread iterates through the list of triangle records and checks its bit in the coverage mask. If the bit is set, it computes the depth of intersection from the triangle's plane equation. The z-values and triangle IDs are stored as array in fast shared memory. We therefore allocate only a single 32-bit value per entry, 16-bit for the signed z-value and 16-bit for the triangle ID. The entries in this array are interleaved to avoid banking conflicts.

Moreover, the available shared memory of almost 16 KB is subdivided into 64 slots, allowing every thread to store a maximum of 63 entries. This limits the maximum depth complexity to 63. The array is maintained in sorted order by inserting new values at the appropriate position, with those entries closest to the camera first (more complex sorting algorithms are likely to perform worse on such a small dataset). It is therefore guaranteed that the array contains the 63 closest fragments, no matter in which order they appear in the input. It turns out that the time spent for sorting is small compared to the subsequent ray traversal time, even for a specially designed worst-case scenario where the entire list of fragments has to be reversed (i.e., has quadratic time complexity in the number of fragments). Moreover, due to the spatial coherence in moderately complex scenes, the insertion operation is likely consistent across many threads. Fragments with rank 64 or higher in the sorted order are discarded. The rationale of this approach is that fragments occluded by 63 closer layers will have minimal influence on the final image, and can therefore be omitted.

To illustrate this claim, we have tested the approximations on objects with very high depth complexity. A ground truth implementation using larger but slower global memory rather than shared memory was used for comparison. The object presented in Figure 3.4, which was chosen for the highest depth complexity from our sample scenes, has a maximum depth complexity of 81 in the present view. Fewer than 0.05% of pixels exceed the threshold of a 5% difference between the fully correct and the approximated image (i.e. 12.75 gray levels of the overall color range 0-255). The contrast enhanced difference image in Figure 3.4 confirms this observation.

Allocating 16 bits for a triangle id limits the number of simultaneously visible triangles to 64k. We have found this to be an acceptable compromise, since the visual complexity in a scene is usually dominated by the volumetric objects. There are two ways to trade off other features for a larger number of visible triangles. We can forsake the ability to access a triangle's normal for surface shading (see Section 3.2.3). In this case the triangle id can be replaced by an object id, and we can still perform all volumetric shading effects and CSG operations. Alternatively, we can limit the maximum depth complexity to 32 layers, and use the additional 32-bit per depth entry for a quantized normal.



(a) (b) (c) (d)

Figure 3.4: (a) Thumbscrew (max. depth complexity of 81). (b) Image from our system (depth complexity limited to 55). (c) Reference image with full depth complexity (slower algorithm). (d) Enlarged differences. Fewer than 0.05% of the pixels show errors > 5% of the color range.

3.2.3 Ray casting of homogeneous segments

After sorting, a pixel thread iterates through the above-mentioned depth sorted array in order, from nearest to farthest. The depth interval between two consecutive entries in the array defines a homogeneous segment of the ray with respect to intersected objects. While traversing the array, we maintain information on the set of currently intersected scene objects, which are leafs of the CSG tree defining the scene. Figure 3.5 illustrates an example of the CSG operations enabled in this way.

An empty set of scene objects means the segment can be skipped. If one or more scene objects are intersected, the starting point of the ray has to be transformed into the model coordinates of each scene object, thus forming a set of object space rays. Each object space ray has to use the same step size in world coordinates to correctly blend the samples from each scene object. Since scene objects can have different size and resolution, we use the minimum step size of all objects along the ray segment to guarantee that no features are missed. Since this can lead to oversampling of objects with lower resolution, we provide an option to adjust this parameter. Objects which would require a high sampling rate might not provide more information at that rate, so we let the user balance between speed and accuracy. However, oversampling does not significantly influence performance due to texture cache coherence. For blending sampling points of different objects, we query the transfer functions separately and multiply the resulting colors with their transparency and a sampling factor which accounts for oversampling. The results are summed up and accumulated to the ray. For purely polyhedral objects made from a homogeneous material, no volume texture sampling is necessary. If a ray segment consists only of polyhedral objects, the loop can be avoided while still yielding high quality translucency. This can yield performance improvements of up to 10%. Once a ray segment has been sampled, the intermediate result is blended into the final result color for this pixel. If a certain opacity threshold is exceeded, the ray can be terminated.

Since polyhedral geometry plays an important role, both surface shading of the polygons and volumetric shading from 3D textures are combined. Optionally, conventional surface material parameters can be assigned to every scene object to add Phong shading and surface transparency effects at object boundaries.

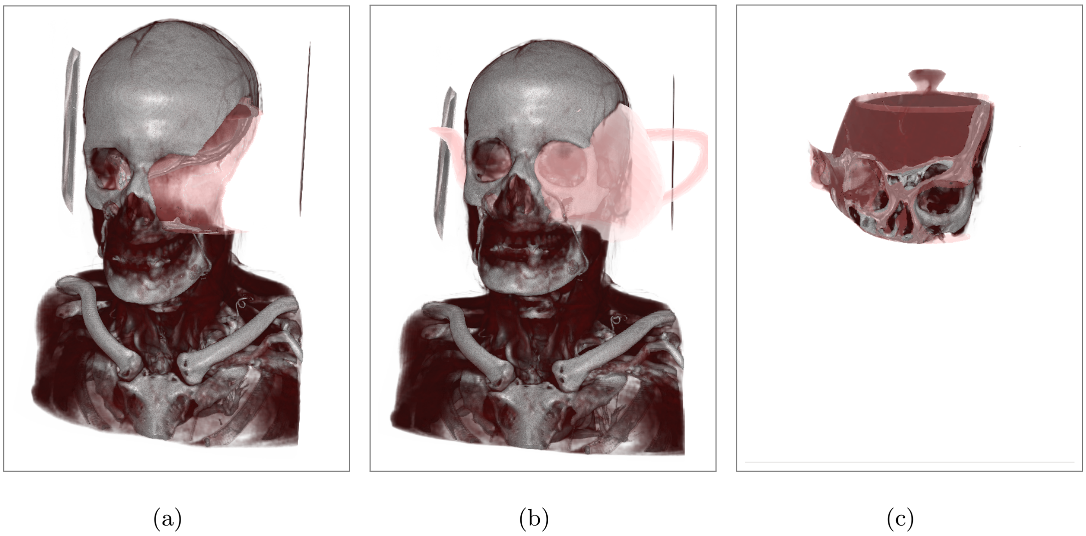


Figure 3.5: CSG operation examples: boolean difference (a), boolean union (b) and boolean intersection (c). Note that visualization of the boundary surface for highlighting cutting region(s) can be toggled by including or not including the boundary in the operation as shown in (a) and (c). In both cases the particular choice results in cutting surface being highlighted.

3.2.4 Empty space skipping

Empty space skipping is an essential acceleration technique for volume rendering. In conventional hardware accelerated ray casting, exterior empty space skipping is typically done by calculating bounding geometries [10, 129]. The bounding geometry is rendered with vertex coordinates encoded as RGB values, so that the graphics hardware computes the ray entry and exit points for accumulation. Interior empty space skipping either requires a multi-pass ray casting approach similar to depth peeling or sampling of an additional lower resolution texture, which encodes empty and full bricks (sub-volumes) of a volume.

The efficient handling of bounding geometry in our rendering system allows exterior and interior space to be skipped in a uniform way. A lower resolution mask volume is calculated decomposing the original volume into uniform bricks, which are masked out if the transfer function evaluates to zero at each underlying dataset voxel. The brick size is user configurable per volume, for medical datasets a size of 8^3 resulted in maximum performance. The actual bounding geometry polygons enclosing full bricks are obtained using a traversal similar to a marching cubes algorithm [133] of the mask volume, which avoids generation of duplicate polygons. Detection of empty bricks and the subsequent geometry calculation for a particular volume requires evaluation of the transfer function for all voxels in a volume and must be repeated every time the transfer function changes. For interactive transfer function editing, short calculation times are essential. This can easily be achieved by performing the calculation in parallel on the GPU, thereby avoiding additional slow data transfers from system memory. Computation times for a 512^3 dataset are around 5 milliseconds on an NVIDIA 280 GTX.

New bounding geometry polygons are inserted into the global triangle list in global GPU memory and associated with the relevant scene object. During ray traversal using the sorted fragment list, bounding geometry polygons allow for simultaneous exterior and interior space skipping based on a set of active objects, which is updated whenever a bounding geometry fragment is processed, adding or removing a particular object. Note that the set also contains in/outside information from all polygonal objects and is the basis for the evaluation of the CSG-tree. The algorithm may directly skip to the next fragment, whenever the set is empty and/or the CSG expression evaluates correspondingly.

The use of tight fitting bounding geometry improves frame rate by up to 15%, in particular for transfer functions which make large portions of a volume transparent. In general, the optimization affects both the triangle and the pixel processing kernel, since smaller triangles can be processed more efficiently than large ones in the triangle kernel, and the pixel kernel needs to process fewer ray segments, both in screen space and along the rays.

3.2.5 Volume texture atlas

To overcome the limited possibilities concerning reallocation of arrays in global memory, we use a 3D texture atlas for the volumetric datasets. A texture atlas is a single large 3D texture, which is assigned to one texture unit. The maximum size of the texture atlas depends only on the available graphics memory.

Finding an optimal memory layout is an instance of the cuboid packing problem [101]. Since volume data sets often have a square cross section with a power-of-two edge length, we can (without significant waste of memory) reduce the packing problem to a single dimension, where the solution is trivial. The texture atlas memory is organized into slots of a certain size, depending on the application's needs. A volume can occupy one or several contiguous slots. The border voxels of every slot are duplicated to allow the use of unclamped texture coordinates in the innermost loop (this is not related to OpenGL texture borders, which facilitate seamless stitching of separate textures).

3.3 Results

We tested our implementation using CUDA 2.0 on a desktop PC (Intel 3.16 GHz Dual Core2-Duo with 3 GB RAM) running 32-bit Windows Vista and 64-bit Linux. We used two alternative GPUs, a GeForce GTX 480 with 1GB RAM and a Tesla C1060 Computing Processor with 4 GB RAM. The latter is able to work with volumes up to 1024^3.

3.3.1 Performance comparison

For performance evaluation, we compared our framework with two recently published state-of-the-art multi-volume renderers [24, 199], which were kindly provided by the authors. In the following, we refer to these tools as *Roessler* and *Brecheisen*, and *Ours* for our own work. The comparison was done on the GTX 280.

C1 represents a high quality medical scan. C2 and C3 are medical scenes with multiple volumes, which use multiple modalities or scans from different body parts.

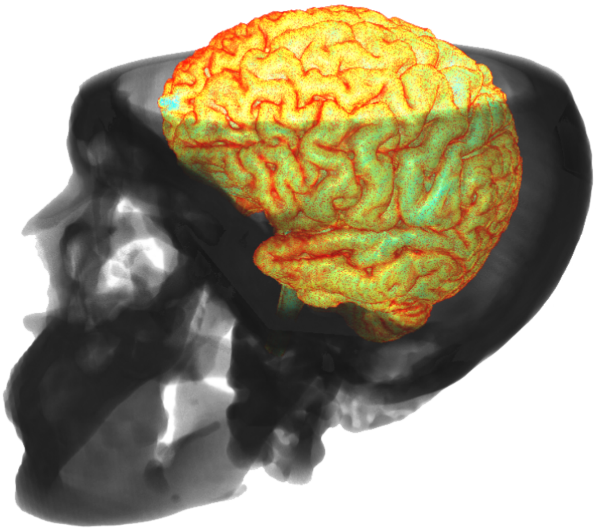


Figure 3.6: A segmented brain from MRI in an open skull from CT, each contain approximately 512^3 voxels.

C2 is shown in Figure 3.6 and C4 and C5 show an anatomical scan involving a time resolved scan, for example of blood flow, leading to a larger number of volumes. C6 is a combination of an anatomical scan with a high resolution iso-surface. C7 is a volume of the abdomen combined with a 10k polygonal model representing the liver surface, a 30k polygon model representing the liver portal vessel tree and a 5k polygon model representing a tumor.

Table 3.1 and Figure 3.7 compares the frame rates achieved with *Roessler* and *Brecheisen* and our approach. *Roessler* generates shader code for every scene and is therefore quite efficient for a few volumes. However, it cannot handle geometry and

No	Volumes	Polygons	Roessler	Brecheisen	Ours
C1	1×512^3	none	14 - 46	9 - 17	10 - 21
C2	2×256^3	none	24 - 38	6 - 12	17 - 25
C3	4×256^3	none	5 - 9	1 - 3	16 - 25
C4	1×256^3 7×128^3	none	1.5 - 3	n.p.	15 - 23
C5	1×256^3, 20×64^3	none	n.p.	n.p.	10 - 17
C6	1×512^3	$1 \times 30k$	n.p.	$1 - 2$	$7 - 15$
C7	1×512^3	$1 \times 30k$ $1 \times 10k$ $1 \times 5k$	n.p.	$0 - 1$	$7 - 15$
C8	50×64^3	$1 \times 30k$	n.p.	n.p.	$8 - 16$

Table 3.1: Overview of test scenes used for performance comparison with state-of-the-art tools.

therefore cannot be used for cases C6 through C8. It could also not run C5 on our system since the generated shader code was too large. *Brecheisen* is able to handle up to four volumes combined with an arbitrary number of intersecting geometric objects, and therefore cannot handle C4, C5 or C8.

In our experiments, we varied viewport size (between 512^2 and 768^2), transfer functions and camera positions. Reported minimum and maximum frame rates are averaged over the tested viewport sizes, since the tools used for comparison do not allow arbitrary choices of viewport size. Lighting was done with pre-calculated gradients, since *Roessler* only provides this method. Progressive rendering was disabled in all evaluated tools. The tests were performed with 8-bit value per sample datasets due to the restrictions of *Roessler* and *Brecheisen*. A transfer function with high transparency was chosen to avoid early ray termination. Camera zoom was set to a value such that most screen pixels were covered. Table 3.1 indicates that our approach compares favorable with shader based systems in particular for scenes with many volumes or complex geometry.

While not its main objective, our system can also render correct transparency in scenes with high geometric depth complexity. We compared this capability to NVIDIA's Dual Depth Peeling (*DDP*) demo [16]. The results of these tests are outlined in Figure 3.8. We used three scenes with different depth complexity and memory bus consumption for the geometry passes. Table 3.2 confirms our expectation that it can outperform depth peeling for more complex scenes. Note that unlike *DDP*, we use high quality per-pixel Phong shading. Figure 3.9 shows the test scenes from a viewpoint with average depth complexity. S1 contains the Stanford dragon reduced to 20k polygons. S2 contains 20

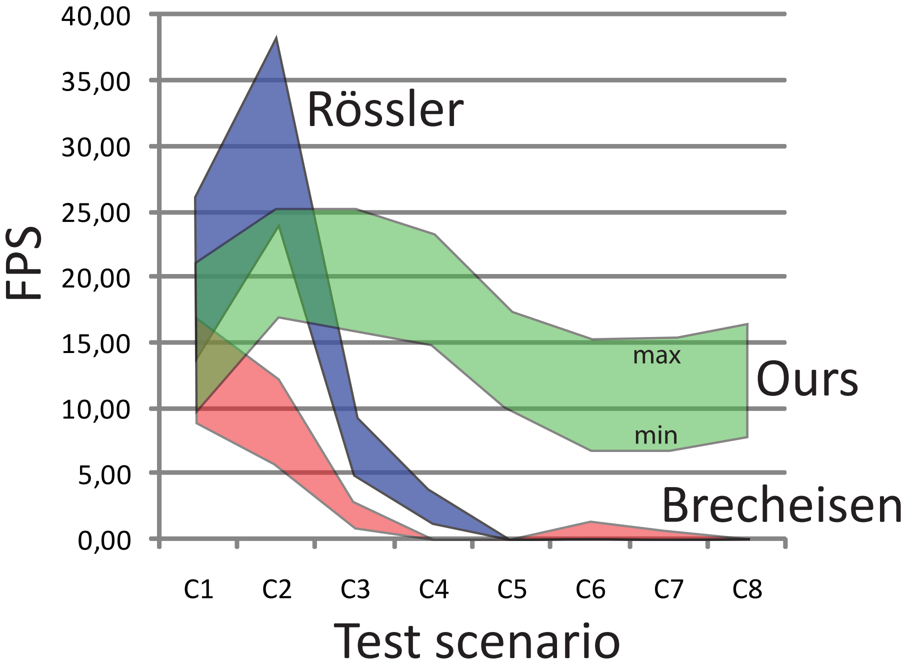


Figure 3.7: Overview of test scenes used for performance comparison with the state-of-the-art tools *Brecheisen* and *Roessler*

Scene	MaxDepth	DDP	Ours
S1	20	123 – 150	52 – 62
S2	40	14 – 16	18 – 28
S3	81	25 – 46	31 – 52

Table 3.2: Comparing Dual Depth Peeling (DDP) to our approach for three scenes with high depth complexity (MaxDepth); screen size: 1050×800. In scenes with high and large-area depth complexity, our rendering system outperforms DDP.

boxes in a row. S3 contains screws with polygonal winding. Our system scales very well when the geometry is filled with a volumetric texture or when complex objects intersect. Table 3.2 supports this observation.

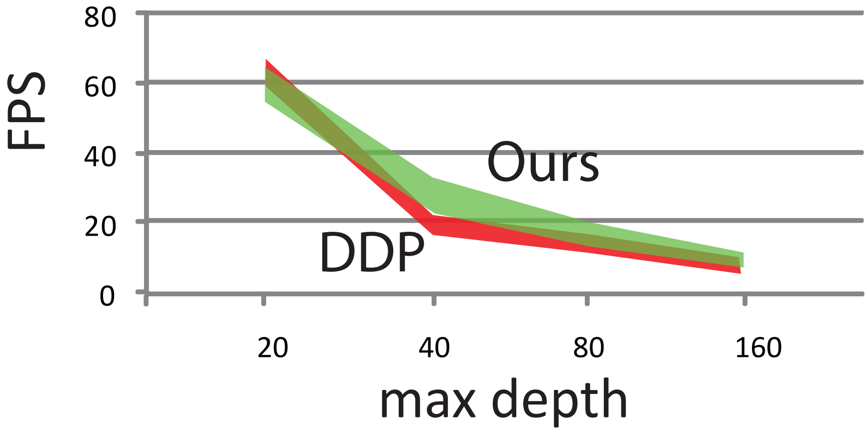
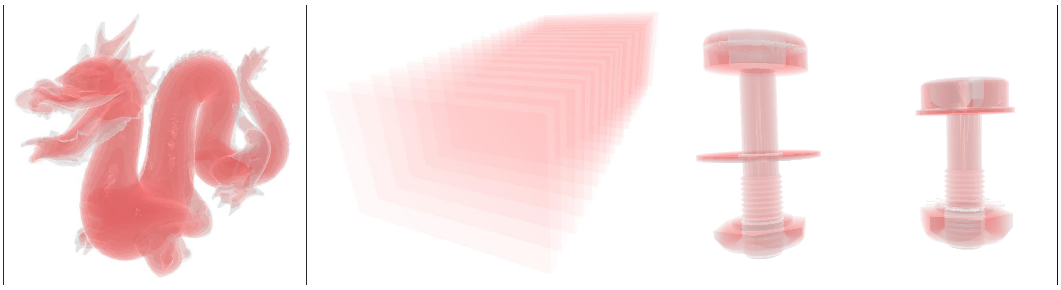


Figure 3.8: Comparison to Depth Peeling (***DDP***) [16]. Our approach is comparable and slightly faster for highly complex screen filling scenes.



(a) **Scene S1** (b) **Scene S2** (c) **Scene S3**

Figure 3.9: Scenes with a high depth complexity for a comparison with the NVIDIA Dual Depth Peeling Demo [16].

3.3.2 Workload distribution

In order to better understand our rendering system, an analysis of the workload for various conditions was undertaken. The triangle kernel uses only 2-10% of the computation time, depending on the triangle/voxel ratio of the scene. The NVIDIA Visual Profiler revealed that the triangle kernel is able to efficiently occupy all processors of the GPU, thus optimally hiding global memory latency. Moreover, projecting triangles and computing coverage masks are very uniform tasks with few diverging branches, which allows peak rates to be obtained. Overall, the pixel kernel uses up to 90% of the computation time. Since kernels are executed in threads on the GPU asynchronously with respect to the CPU, the GPU code was instrumented using the *clock()* API function to query the number of GPU cycles. Unfortunately, this function does not report the time spent per thread, but rather the clock of the processor, while the processor performs time slicing of multiple threads. Low consumption of register space and shared memory allows more threads to be active (i.e., time-sliced) concurrently and therefore better hiding of memory latency. This improves performance, but at the same time makes clock measurements unreliable.

We therefore experimented with both low and heavy loads on shared memory. A low load can be achieved by limiting the maximum complexity of the scene. For example, lowering the maximum depth complexity or the number of intersecting objects frees register space and shared memory per execution block, and therefore enables the GPU scheduler to run other blocks while waiting for memory transfers. Through this strategy, we could obtain overall performance improvements of up to 40%. We also selected setups which create a heavy load on shared memory, giving no opportunity for time slicing. This allowed us to perform the following analysis.

For scenes with significant geometry, but relatively little volume data, the results indicated that up to 50% of the pixel kernel's work consists of sorting fragments in z-direction. The rest consists of Phong shading of geometry surfaces (<3%), accumulating pure geometry interiors (<3%) and ray traversal of volumetric data (23%).

As an example for a scene with heavy volumetric load, consider the head/brain scene from Figure 3.6. In this example, computing and sorting depth values amounts to 7% of computations, while ray traversal takes up to 60%, which can be split into 12% for ray setup and 48% for the accumulation loop. Tests with various scenes confirmed that ray traversal is generally the most costly part. This is reassuring because it seems likely that the $O(n^3)$ complexity for handling the volumetric data should dominate the system's performance. However, it also means there is limited room left for improvements.

3.3.3 Memory consumption

The memory requirements of the described method are mainly related to the texture atlas and the triangle records used to communicate between triangle kernel and pixel kernel. The texture atlas size depends on cumulated sizes of the volumes contained in the scene. We commonly use a texture atlas of 528^3, which can accommodate all the scenes used in the evaluation, in about 562 MB of global memory. Pre-calculation of gradients for lighting increases the required memory by a factor of four, because for every sample three additional values need to be stored.

The triangle record memory consumption for a viewport size of 768^2 pixels is 96^2 (tiles) $\times$ 64 (chunks per tile) $\times$ 4 (bytes per index) = 2.25MB for the fixed-size set of chunk indices, plus the variable-sized chunk pool (768 bytes per chunk). Our experiments were run with a total chunk pool size of 6MB, which can be raised for more complex scenes up to the limit imposed by the available amount of memory on the graphics card.

3.4 Flexible multi-volume single-scattering

An extension to the presented method, shows the flexibility of our rendering systems. The common Phong light model can easily be replaced in our systems with more sophisticated but still simple to implement models. Influenced by the visually appealing efforts of the algorithm proposed in [201] we present a concept to improve the visual results of our CUDA-based renderer in a similar way and show our initial attempt to upgrade our renderer.

The approach of Ropinski's team makes use of a separate lighting volume which we have decided to avoid for a first evaluation of incorporating light transport assumptions into our own rendering solution. The lighting calculation in this example is based on the lighting models covered by Max [150], which are widely used in direct volume rendering solutions. We use a simplified version of the absorption and emission scheme including single scattering and shadowing. As a phase function for the scattered light, we also follow Max' primary suggestion and use the Henyey-Greenstein function shown in Equation 3.4 from [97].

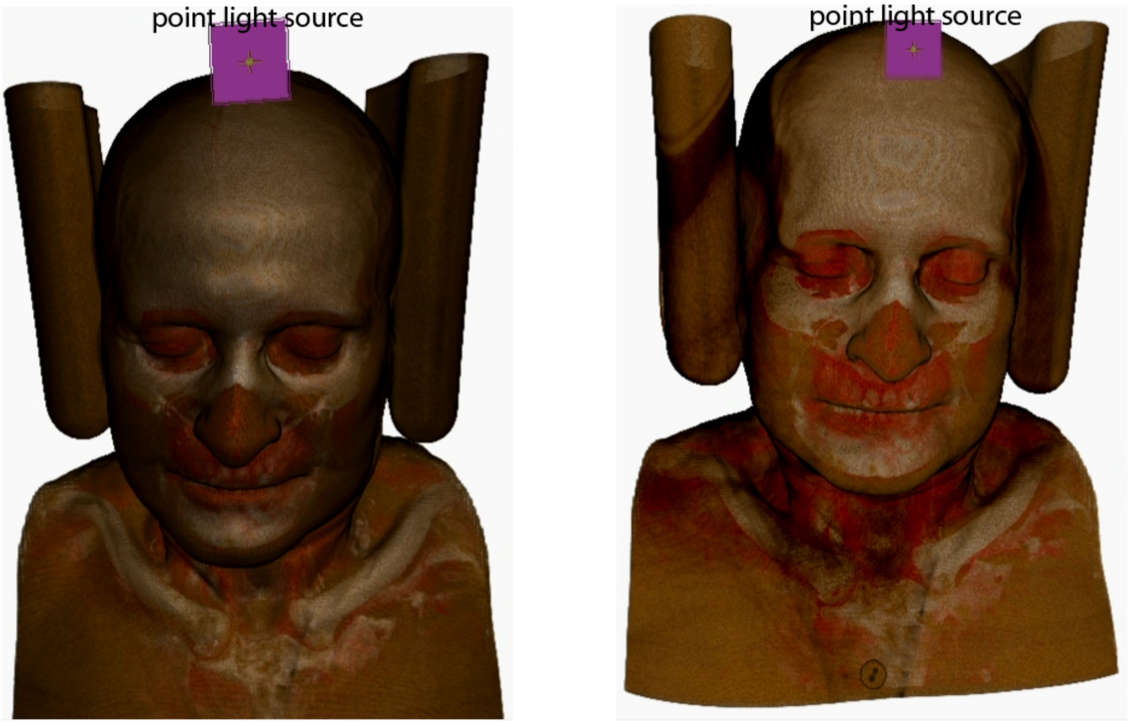
$$p(\theta) = \frac{1}{4\pi} * \frac{1-g^2}{(1+g^2+2g\ cos\theta)^{\frac{3}{2}}} \tag{3.4}$$

In this equation g stands for the average cosine of the scattering angle, reaching from -1 to 1. A negative value would mean backward scattering, whereas a positive one means forward scattering. A value of 0 reduces the formula to $\frac{1}{4\pi}$ which results in isotropic scattering, i.e. light scatters equally in all directions. Since we do not really know the exact scattering behaviour of the data or respectively which kind of tissue the light passes when rendering the transferred values of the volume set, we choose a value of 0.3, i.e. slight forward scattering. For the calculation of the light transfer through the volume, we use the standard lighting absorption calculation showed in Equation 3.5, which is also mentioned in [150]. T is the resulting transparency when integrating from the voxel x in the direction of the light source, where 0 in this case is the point where the light ray enters the volume, since the way between light and volume is considered to be in vacuum and therefore does not absorb any light. τ is the tissue's extinction coefficient at the current position of the ray, which here means its opacity.

$$T(x) = \exp(-\int_0^x \tau(t)dt) \tag{3.5}$$

After calculating the voxels result color according to the absorbed light, we do the same calculation for the scattering points and add their contribution to the final calculation. Currently we use up to light scatter samples which are guided by the density gradient. More samples are taken in areas with a high density fall-off. These areas are likely to be highly scattering media.

This lighting approach has a clear impact on the rendering performance. However,



(a) Phong gradient approximation illumination.

(b) Single scatter online evaluation.

Figure 3.10: Comparison of common Phong shading by gradient approximation (a) online scatter approximation approach (b). Depth perception and illumination quality are improved in (b) for a point light source as shown in this example. The data set consists of approximately 512^3 scalar density values and was rendered at 55 fps (a) and 31 fps (b) in a 1024x768 viewport on a system containing three Nvidia Quadro 6000 GPUs.

considering that our presented rendering systems shows an average frame rate between 40 and 60 fps, halving will still result in an interactive system. Experiments showed, that the light transport approximation presented here requires between 20% to 35% percent of the rendering performance with Phong shading only. Depending on the application, this might be an issue. However, the results differ clearly in terms of illumination quality, as shown in Figure 3.10, which compares common Phong shading by gradient approximation with the presented online scatter approximation attempt for a 512^3 dataset and a 1024x768 viewport.

The scattering sampling points and the the scattering angle are determined from a local neighborhood of the voxel, guided by the gradient approximation at that position in our algorithm. We assume that a strong gradient indicates a less scattering, more reflecting surface and that the gradient points in the direction of less dense, more scattering media. The scatter angle an therefore the amount of in-scattering to the current sample position is defined by the angle between the gradient approximation at a scatter sample point and the gradient approximation at the current sample position. The cosine of the scattering angle gets zero, only if the direction of a scatter sample gradient is identical to the current sampling point's gradient vector. We use up to 15 scatter sample positions, depending on the strength of the gradient at the current volume sampling position and weight their influence by the distance to the actual sampling point and the gradient magnitude at the given scatter sampling position. This scatter sampling point scheme is outlined in Figure 3.11. For the further calculation of the light transfer through the volumes, we use the standard lighting absorption calculation which is also mentioned in [150]. All the required computations are performed on-the-fly on the GPU without the need for further parameter buffers.

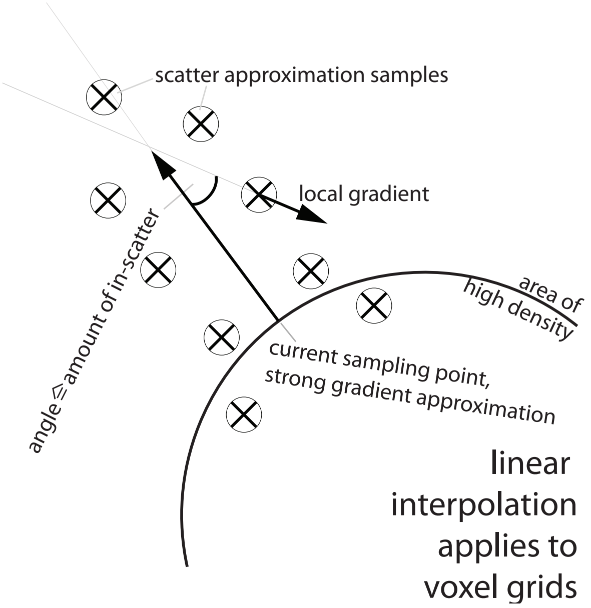

scatter approximation samples
local gradient
angle ≙ amount of in-scatter
area of high density
current sampling point, strong gradient approximation
linear interpolation applies to voxel grids

Figure 3.11: Our scatter approximation sampling scheme based on local gradients.

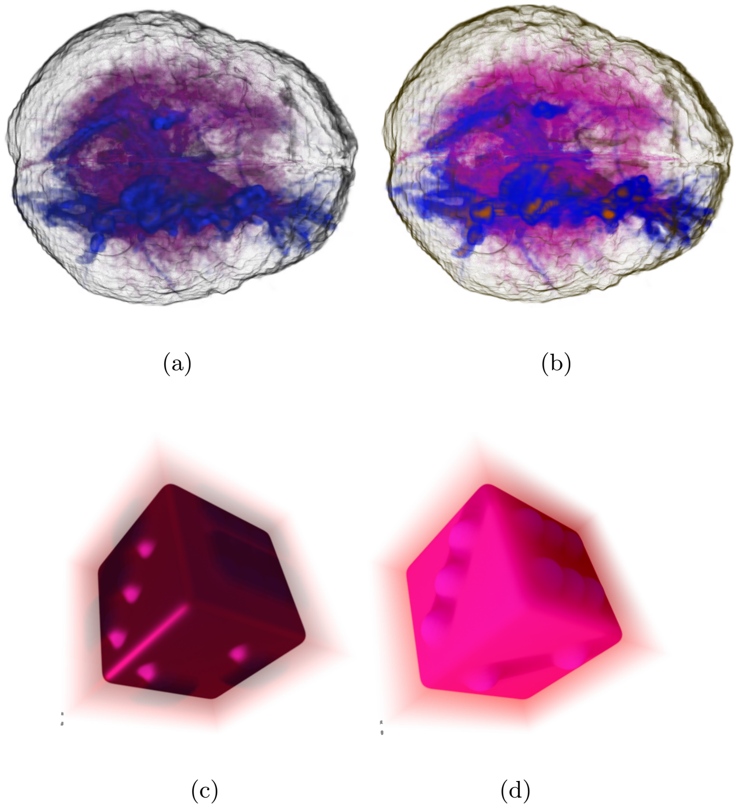
3.4.1 Single scattering results

We tested our method on an Intel QuadCore 3,16 GHz, supplying three Nvidia Quadro 6000 GPUs.

As shown in Table 3.3, the performance impact of the new method is significant, however it is still an interactive algorithm. For Table 3.3 we compared different transfer functions (TF) with different opaque/reflective – translucent/scattering ratios. The test dataset is shown in comparison to a local Phong illumination model in Figure 3.12.

	T1	T2	T3	T4
Phong Model	45	41	41	39
L1	21	19	20	18
L2	11	9	11	8

Table 3.3: Performance comparison for our method in average frames per second for different transfer functions with different opaque/reflective – translucent/scattering ratios compared to a local Phong illumination model. For this test we used two registered 256^3 datasets in a 1024 x 768 viewport. (T1 = both datasets are opaque; T2 = dataset is 1 opaque, dataset 2 is translucent T3 = both datasets are half-opaque – half-scattering; T4 = both datasets are translucent; L1 = light transport; L2 = light transport including scattering.)



(a) (b)

(c) (d)

Figure 3.12: One of the used test datasets: Two registered volumetric datasets (256^3 MRI and DTI-MRI of the brain), using the common Phong model (a) and our method (b) and the same transfer function. The difference is additionally illustrated with a generic dataset and an opaque transfer function peak in (c), Phong and (d) our method

3.5 Discussion

Due to recent improvements in multicore GPU programming models, we have been able to develop a new approach for multi-volume and geometry rendering and substantiated *TH1* significantly. CUDA is not specifically designed for graphics, but allows the development of a software rendering pipeline, which is executed in a massively parallel way. A key property of this approach is the efficient use of scatter operations, which allows a scene's polygons to be rasterized only once.

This approach is relevant to a broad range of applications. It increases the practical applicability of volume rendering to very complex scenes, including intersecting volumes and geometry in clinical practice or in computer games and art. As shown in this Chapter, our system is easy to implement, robust, and readily runs on consumer hardware. Since our implementation is designed for manycore graphics hardware, we are looking forward to future general purpose GPU architectures such as Larrabee [206], which should significantly increase the flexibility of the algorithms and enable new design choices. A first step in direction of this new flexibility of rendering engines is already shown in this chapter by replacing the normally used Phong lighting approximation by a novel multi-volume single scattering approach in Section 3.4.

Chapter 4

Optimizing ray-based image synthesis for the human visual system

This Chapter presents a new method to control scene sampling in complex ray-based rendering environments. It proposes to constrain image sampling density with object features which are known to support the comprehension of three-dimensional shape. The presented method uses Non-Photorealistic Rendering (NPR) techniques to extract features such as silhouettes, suggestive contours and highlights, ridges and valleys. In order to map different feature types to sampling densities, we also present an evaluation of the features' impact on the resulting image quality. In addition, we present a discussion of three different strategies to reconstruct the image from sparse sampling data. In particular, we compare linear interpolation on an adaptively aligned fractal pattern to frame blending and frame warping techniques. Furthermore, this Chapter presents an algorithm which uses our method in order to guarantee a desired minimal frame rate. Our scheduling algorithm maximizes the utilization of each given time slice by rendering features in the order of their corresponding importance values until a time constraint is reached. We demonstrate how our method can be used to boost or stabilize the rendering time in complex ray-based environments consisting of geometric as well as volumetric data.

4.1 The interactivity vs. quality problem

A common challenge of high-quality ray-based image generation is maintaining the interactivity of the applications. This interactivity is normally achieved by sacrificing some of the image quality during the interaction and by progressively refining the result as soon as the scene interaction stops. The simplest method in this context is regular sub-sampling: rendering the scene in a small viewport during interaction and stretching the resulting image to the target image size using linear interpolation. This method indiscriminately discards features and results in a blurred render frame or block artifacts.

Our main focus is to provide highly interactive frame rates for the high quality medical visualization of huge volumetric datasets without a severe loss of image quality. In a medical context, this feature is essential because physicians track and orient themselves based on small details of the dataset during the 3D scene interaction. If those details get lost at any time, the method is unsuitable for intervention planning or for similar tasks requiring 3D visualization methods. To preserve the required details, adaptive scene-sampling techniques can be used.

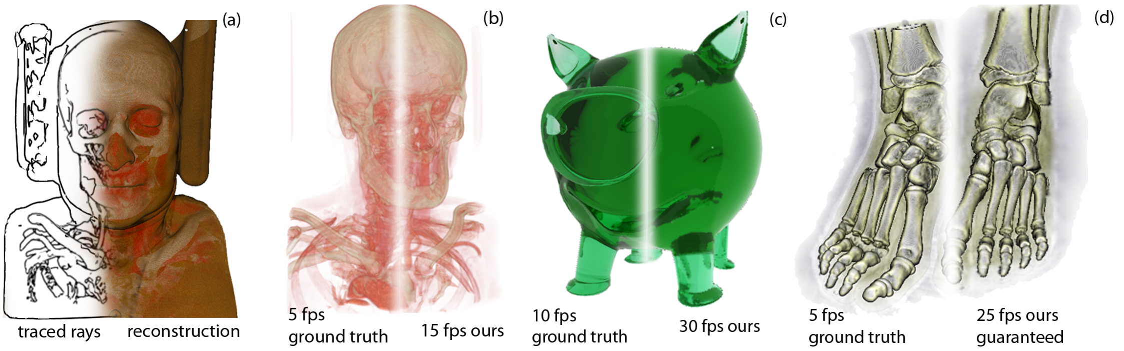


Figure 4.1: Four different scenarios rendered with a conventional ray-based rendering engine and with the presented adaptive approach in a 1024×768 viewport. (a) shows the areas in which the rays are fully computed (left) and the resulting reconstruction by our algorithm (right). (b) shows how our approach can boost the frame rate while preserving the quality level. Our algorithm can also be applied to ray-tracing scenes with complex materials (c). Priority sorting of the expected visual quality of a pixel allows us to guarantee frame rates for every type of ray-based environment while maintaining a visually appealing result (d).

Adaptive sampling approaches try to assign the computational costs to regions with high image fidelity and to approximate the remaining image parts. Typically, these techniques use features that have been detected in the image plane. These approaches obviously require the final result as an input for the optimal result, which is impossible. Hence, image regions from previously rendered frames [51] or sparsely sampled regions [181] are used. However, image space methods suffer from less accuracy than object space methods because of the required projection to discrete image pixels. Furthermore, many image space algorithms may cause more computational overhead than benefit because of the already high GPU utilization of modern ray-based image synthesis systems [184].Therefore, we investigated the key element of adaptive approaches, which is the determination of which elements of an object can be coarsened and which must be preserved. Much perceptually based research has been performed in this area by researchers from the Non-Photorealistic Rendering (NPR) community. However, these results have only been used for scene stylization and enhancement. We present a new sampling strategy for ray-based image synthesis, which uses information about object space features that are known to support the comprehension of 3D shapes [47]. In this Chapter, these NPR techniques are used to control the reduction of ray samples and thus to achieve a higher image quality while maintaining the same level of interactivity.

Our implementation produces a feature buffer for every frame that is efficient enough for use during the ray generation as a lookup table for the required ray density. We derived a feature priority map from the feature buffer that consists of silhouettes, suggestive contours, ridges and valleys, all of which affect the ray density differently (see Figure 4.1(a)). Because different features generate different ray densities, our method is able to support an importance-driven rendering to guarantee the minimum desired frame rate. Even though our main focus is the visualization of volumetric datasets, we also demonstrated a way to apply our method to geometric objects with highly complex materials in ray-tracing scenes. Figure 4.1 shows some selected examples using this approach.

The main contributions of our method can be summarized as follows:

- A method that allows the optimization of the ratio between the sampling rate of the scene and its resulting perceptual quality (Section 4.3).
- A progressively refineable sampling pattern, which is used to reconstruct sparsely sampled regions of the image without the need for frame-to-frame coherence (Section 4.3.4).
- An algorithm that uses our method to guarantee frame rates while maximizing the visual quality within the available time frame (Section 4.4).
- An evaluation of different object space line features to categorize them based on their abilities to enhance the image quality (Section 4.5).
- A discussion of an addition to our method, which uses a fast computation of the image space visual saliency. With this method, we can also include features that can only be evaluated in image space (e.g., textures and shading details) (Section 4.8.1).

4.2 Two pass ray setup

Our approach consists of two passes. First, a set of line features is extracted from the data set's corresponding meshes and projected to the screen space, resulting in a feature buffer. According to our evaluation (see Section 4.5), we assign priorities to different types of features. The feature buffer is evaluated during the ray setup and traversal, which forms the second pass. Given that we can assign a priority value to every ray, it is possible to construct a rendering system that aims at producing the best image quality within a given time frame as outlined in Figure 4.2.

We adaptively adjust the image space sampling frequency according to the feature buffer. More rays are sent into the scene in feature-rich areas and their vicinity, while the sampling frequency for feature-poor areas is strongly decreased. The same strategy can be

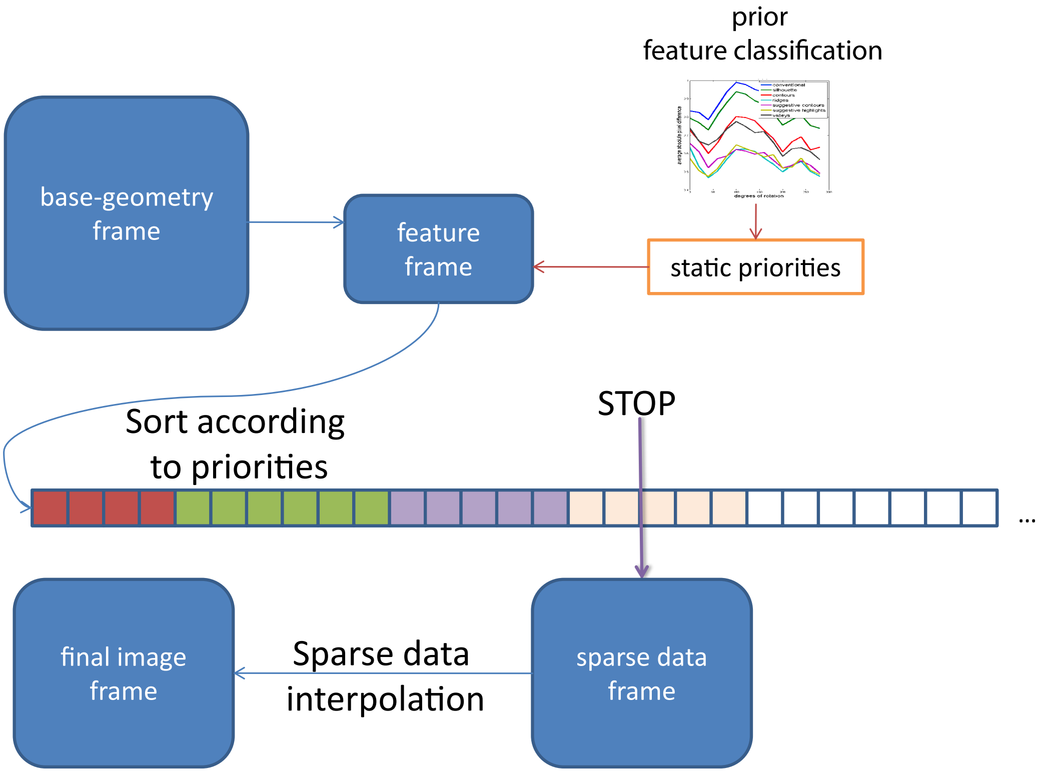


Figure 4.2: Overview of our prioritized rendering algorithm. Every ray's priority is computed according to Section 4.3.2 and is used to sort the pixels in a one-dimensional priority queue. If a certain time limit is reached, the rendering process stops.

used for per-ray quality parameters (such as ray bounces, object space sampling frequency or stopping threshold). Finally, we reconstruct the image by filling in color values for pixels to which we have not previously assigned a ray. In Section 4.3.4, we present a suitable method for a full image reconstruction using a fractal reconstruction pattern and an adaptive linear interpolation.

Our method is applicable to a ray-based rendering of geometric surface meshes and to volumetric data sets. The only difference is given by the feature extraction step. For surface meshes, the feature-forming geometry is defined by the mesh itself. Using volumes requires the extraction of multiple iso-surfaces based on an evaluation of the given transfer function before the line features are rendered.

4.3 Importance-driven sampling and reconstruction

To control the frequency of the sampling pattern based on different types of features, we render an importance buffer in each frame (Section 4.3.1). This importance buffer is filled by a projection of each single line element to the screen space followed by a computationally cheap falloff estimation. This buffer defines the importance of pixels (Section 4.3.2). We derived the corresponding sampling density using a fractal sampling pattern (Section 4.3.3).

4.3.1 Object space importance buffer

To provide a sufficiently high frame rate in the first render pass, we have extended the approach from [52] with selective GPU acceleration techniques, which are described in Section 4.6. For our method, the feature projection buffer must be produced in a time frame that is shorter than the savings from the main rendering pass. In practice, we can render this step at several hundred frames per second because the features are rendered as simple OpenGL lines. These lines are also reused from frame to frame, and the vertices are only either removed or inserted. To simulate smooth features and to gradually decrease the priority in the vicinity of features, we can replace these lines by textured triangle strips, compute a distance transform on the feature buffer, or use a fractal pattern as described in Section 4.4, the last option providing the highest flexibility. In this way, we generate an intensity falloff around the features. To remove any hidden features, we also render the underlying base mesh with a homogeneous white surface and a fully opaque one. Next, we encode the resulting feature projection buffer as follows:

- Red pixels define a pure background,
- Black pixels define primary salient features,
  (*luminance* = *0* $\widehat{=}$ *contours*)
- Gray to black pixels define secondary salient features,
  (*1.0 - luminance* = *feature priority* $\widehat{=}$ *ability of the feature to improve visual quality* as defined in Section 4.5)
- White defines homogeneous object areas with no salient features.

4.3.2 Adaptive subsampling

When the feature frame is available, the actual ray traversal starts. Every pixel of the output frame defines a possible ray starting point which is equal to one thread in terms of GPU stream processing. If the feature frame contains a red pixel at the thread's position, this ray's thread will immediately return and fill its corresponding position in the output buffer with the background color. If the feature frame contains a pure black pixel, i.e., a primary feature, the ray will be processed completely until the given convergence criteria are fulfilled. If the feature-buffer shows a secondary feature, we consult an adjustable ray priority table, which is used to determine the image space sampling pattern (see Section 4.3.3). The combination of feature priority p_{feat}, which is deduced from the intensity of the feature buffer, combined with the pattern priority p_{patt}, which is read from the sampling priority table, is used to determine the ray's priority p_{ray}. Every arbitrary function f_{map} with two input parameters is possible to combine these two independent priorities:

$$p_{ray} = f_{map}(p_{feat}, p_{patt}) \tag{4.1}$$

For an efficient implementation, we use a second-order Taylor series approximation, because its implementation consists only of basic and fast algebraic operations. It is defined by six fixed parameters $\alpha_{i,j}$:

$$p_{ray} = \sum_{i+j \leq 2} \alpha_{i,j} \cdot p_{feat}^{i} \cdot p_{patt}^{j} \tag{4.2}$$

The rays are traced according to their priority p_{ray}. If the mapping function captures every ray's contribution to image quality, a fixed threshold can be used to only trace rays with a high contribution. In this case, $-\alpha_{0,0}$ can be used as the threshold and rays with a priority above zero are traced. Another option to sort rays according to their priority and use the available time slot to draw the rays with the highest contribution (see Section 4.4). The way that f_{map} and thus the α values are chosen, controls the influence of the feature priorities on the output image. Low weights for terms dominated by p_{feat} will result in a nearly uniform pattern, while a high contribution of p_{feat} creates samples at only the feature areas. A good trade-off between these extremes is a method that creates a dense sampling pattern along important features while reducing the number of samples along the transition from a feature to feature-less areas. Lower priority features would thus receive a lower sampling density than would higher priority features, and homogeneous areas would contain only a few sampling points (see also Figure 4.7).

In practice, we have used a rather simple choice for the α values: $\alpha_{2,0} = \alpha_{0,2} = 0$ and $\alpha_{i,j} = 0.5 \pm 0.2$ for all remaining terms. However, an optimal mapping function f_{map} takes information about the rendered objects into consideration. Images of strongly transparent objects naturally show few homogeneous areas; thus, increasing the influence of p_{patt} (increasing $\alpha_{0,1}$ and $\alpha_{0,2}$) will have a positive influence on the image quality. Nearly opaque objects with low color variation will benefit from an increasing influence of p_{feat} (increasing $\alpha_{1,0}$ and $\alpha_{2,0}$), as most variation in color appears along the feature regions.

For the remaining case – a white pixel in the feature buffer, which indicates no salient feature at that position – we assign a feature priority of zero and only use the ray pattern priority to determine whether the ray should be traversed. All non-background rays, which have not been traced, are subject to a reconstruction step as described in Section 4.3.4.

4.3.3 Sampling pattern

The choice of an incrementally refineable sampling pattern is crucial to smoothly add detail to transitions between fully traced areas and a coarsely traced background. The design of this sampling pattern should further consider the possibility of interpolating the resulting ray pattern efficiently. Both problems can be addressed by defining a fractal sampling scheme that considers only two local shapes: a square and a diamond (45° rotated square). We start with the coarsest sampling density, which only needs to be sufficiently coarse, and a power of two, and create a square pattern by placing a ray at every 8×8 square of pixels. We then place a sample at the center of the square that exactly matches the center of a pixel and splits every square into four triangles. Together with the surrounding squares, which are augmented with an additional sample, a diamond pattern results. The density of this pattern can be increased by placing a ray at the center of each diamond. This procedure leads again to a uniform square pattern. Repeating these steps places rays at exactly the centers of pixels until every pixel is covered with a single sample. The associated priority values are deduced by starting with the maximum priority and linearly decreasing the priority with each new shape. The whole procedure is outlined in Figure 4.3. The pattern can be refined locally and thus increase the sampling density for arbitrarily sized regions.

4.3.4 Image reconstruction

To improve the quality of the reconstructed image, we have investigated methods to fill areas for which no rays have been traversed. We focus on reconstructing without losing much performance. Our approach linearly interpolates samples based on the fractal pattern presented in Section 4.3.3.

Various methods exist for interpolating non-homogeneously sampled data [8]. However, our sampling pattern allows us to combine the choice of ray locations with their interpolation and compute both steps efficiently. A pixel contained in the interior of a square pattern can be constructed with a bilinear interpolation from the four anchor points defining the square. Because the diamond shape is a rotated square, we only need to rotate the pixel position accordingly to enable a standard bilinear interpolation for this shape.

The transition from one interpolation density to the next requires an additional step, as up to three anchor points might be missing. In this case, we have to interpolate the missing anchor points from the coarser pattern first. From another point of view, we add a ray according to the pattern described in Section 4.3.3, but instead of tracing it, we interpolate its value from the already given rays. As this new anchor point is placed exactly in the middle of the already existing ones, the linear interpolation breaks down to an evenly weighted mixture of the four anchor points. For an efficient implementation, we have to make sure that we do not create a dependency chain when gradually decreasing the sampling density. If the sampling density is decreased too abruptly, there will not be enough lower density rays available to construct the missing points for the next higher density.

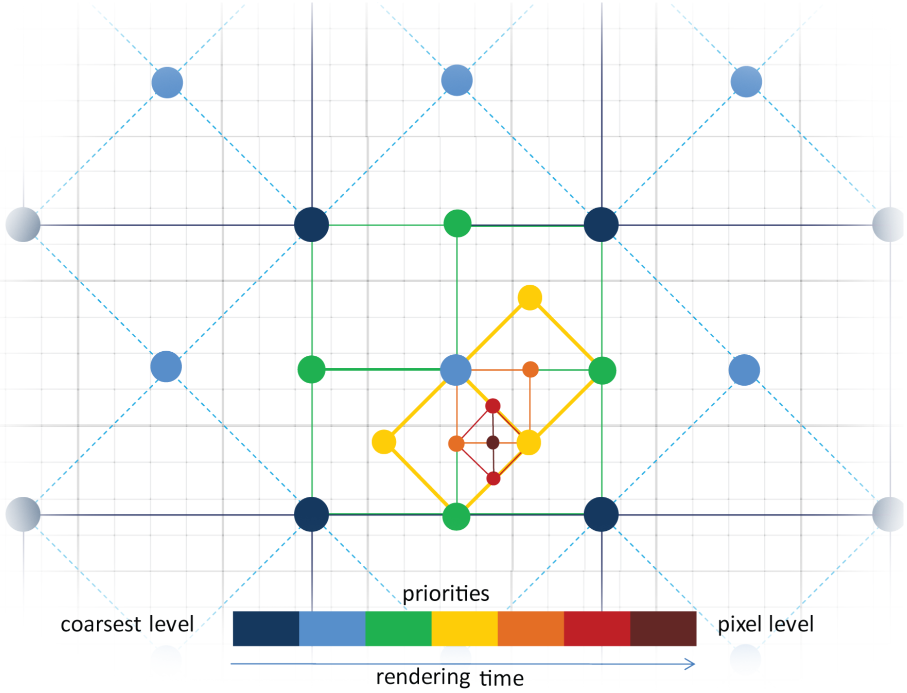


Figure 4.3: We use a sampling priority pattern similar to that outlined in this figure. For illustration reasons, this figure shows a much finer grid than would be used in reality. Blue defines the initial rays with priorities 1.0. With every sampling step, the pattern becomes finer, and the priority decreases (color coded in the figure).

Thus, the missing point has to be constructed from another density level first. Therefore, for maximum performance, we make sure that the features are smoothed sufficiently such that enough rays are traced to construct all of the missing anchor points in a single step.

4.4 Guaranteed frame rate rendering

For a continuous rendering scenario with a good frame-to-frame coherence, we can build a reactive rendering system that adjusts the number of traced rays depending on the time needed for previous frames. This step is possible by dynamically adjusting the threshold that defines which rays shall be rendered, i.e., by decreasing $\alpha_{0,0}$ by a fixed value if the frame rates are too low. If the scene is static and the camera is still, the rays traced in the previous frame are reused, and we progressively add new rays by increasing $\alpha_{0,0}$. Thus, the image converges to the highest quality.

For hard real-time scenarios with low frame-to-frame coherence or with little still image renderings with progressive refinement over time, the aforementioned approach fails. An unexpected load on the GPU, complex objects popping up in the scene or the simple lack of a previous frame prohibits us from deducing enough information for the next frame. However, in these cases, we can still rely on the ray priorities to guarantee the given update rates. We require an additional sorting step before the actual ray tracing is conducted. Every ray's priority is computed according to Section 4.3.2 and inserted into a one-dimensional priority queue. In this scenario, the parameter $\alpha_{0,0}$ is irrelevant because it has no influence on the sorting order. During the following rendering step, each block of threads fetches a set of rays from the front of the queue and processes them. This step is repeated until the available time frame is nearly over. The use of this priority queue guarantees that the available time is spent on rays that have been classified as being the most important. For the sorting itself, we use a fixed number of buckets instead of completely sorting the queue to increase performance. As we cannot guarantee that all elements within a bucket will be processed, we randomize the order in every bucket. Otherwise, the render order for similar ray priorities would match the insertion order, and the sampling density might thus only increase locally.

Our experiments have shown that the reconstruction step's execution time is very short and has little variation. We can thus measure an upper bound for this step in the initialization phase and reduce the time frame during the rendering accordingly to have enough time for the reconstruction step. This setup enables us to output a frame within the desired latency. The time measurement is performed on the graphics card itself, which allows each block of threads to work autonomously without synchronization via the host. For static scenes, we can again use our system for a progressive rendering. As low priority rays are still present in the queue after the time frame is over, we can simply re-launch the rendering kernel right after presenting the current quality level. In this way, the next set of rays are traced within the next time frame, and we are able to progressively update the scene with the next lowest ray importance level.

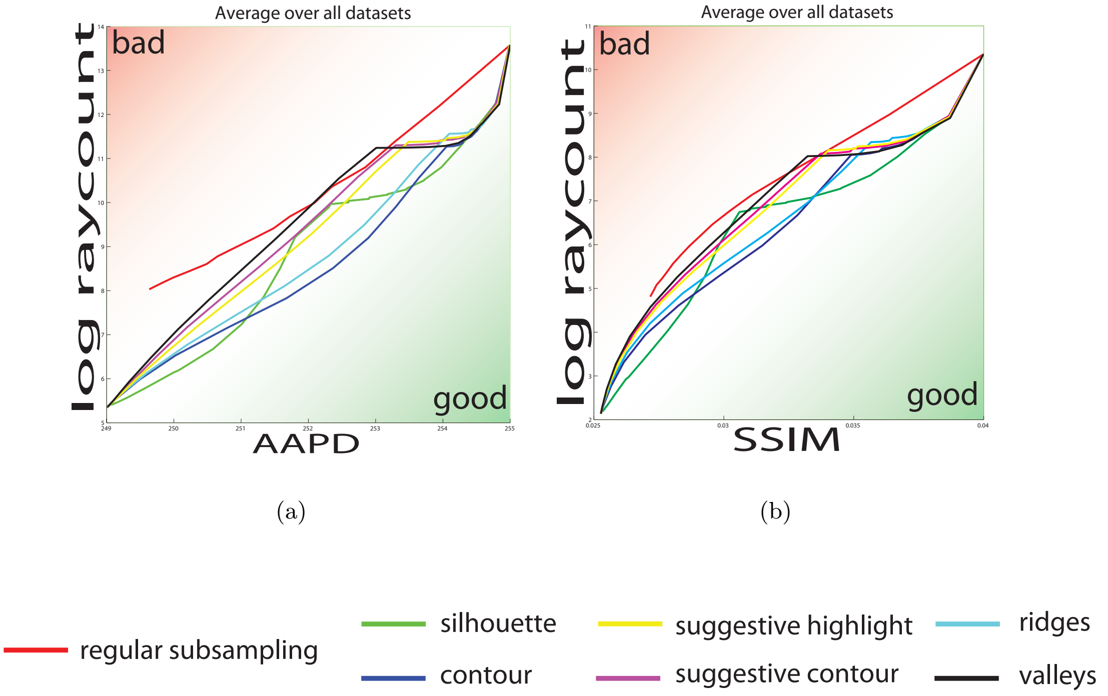
4.5 Feature Classification

To evaluate how features improve the visual quality of the result, we have tested ray traced objects and selected volume-rendered objects with different transfer functions. We assume as the lower visual quality bound the simplest subsampling approach: rendering at a lower resolution with a block filter kernel reconstruction. The upper visual quality bound is given by the ground truth (ray tracing at full resolution). We evaluate subsequently how the image quality improves when pixels, which are marked as a certain feature, are rendered with an increased sampling density.

To quantify the visual improvements of different features, we use two image comparison metrics (comparing the feature-enhanced image to the ground truth image): the average absolute pixel difference (AAPD) and the Structural Similarity Index (SSIM).

The first quantity is simple, parameter-free, and easy to compute. Moreover, it has a physical meaning similar to the mean square error (MSE), the energy of the error signal. Problems with averaging methods arise when the subjective human's perception of image quality has to be quantified. As it was shown in [239], these methods are very similar despite the differences in image distortions.

The second quantity, SSIM, is a generalized form of the Universal Quality Index and was proposed by [240]. This metric shows distinct values for different kinds of image distortions according to well-defined luminance, contrast, and structure comparison measures.



(a)

(b)

Figure 4.4: These graphs evaluate the render quality in terms of the AAPD (a) and the SSIM (b) compared with the required logarithmic ray count that is needed to reach that quality. Both plots are an average of our test datasets (ray-traced geometry as well as volumes). For both evaluation methods, the conventional regular subsampling approach performs worst, whereas rendering along contour lines performs best.

Figure 4.4 shows that all object space features improve the image quality during interaction, compared with the conventional regular subsampling rendering. We observed that using the exterior silhouette leads to the best result for objects with a low interior feature count. Otherwise, using ridges and contours yields the best outcome. In the case of objects with a clear boundary (e.g., as used for ray-tracing), the external silhouette alone shows the highest relative quality improvement. For translucent volumetric objects, the external silhouette does not necessary improve the image quality because it might cover the whole dataset and exclude the (probably more important) internal features. Based on our observations from automatic tests, we first roughly classified the features into *strong visual features* and *medium visual features*. We did not introduce the class *weak visual fea-*

tures here because we have already considered regions on/in an object with no response to any feature extraction algorithm as regions with a low image signal frequency. Our experiments show that *contours* in particular lead to a much better relative perception during the scene interaction for our tested scenes. Ridges and (subjectively) *exterior silhouettes* also yielded a good result in all of the tested scenes, especially for transparent volumetric objects. Therefore, we categorize *contours* as *strong features* (along with the *external silhouettes* as a subset of *contours*) and the remaining ones as *medium features*. In Section 4.4, we directly use the results from Figure 4.4 to define static feature priority lookup tables for each separate feature in a numerical way. Using the values measured in this section, we can provide good default values for the priority lookup tables. However, a user is still able to alter these priorities in our system.

It is desirable to evaluate the importance of different features for every frame independently. For a fully automatic evaluation of the image quality improvement of different line features per frame, obtaining the ground truth is necessary, which is of course not feasible during runtime. Therefore, users can alter the proposed feature ranking through the user interface of our system. Preliminary experiments with this feature have shown that users tend to fully disable features such as suggestive contours, suggestive highlights and valleys to gain higher frame rates. These features have also shown a low visual improvement during our offline evaluation as outlined in Section 4.5.

4.6 Extending the OptiX engine with our method

We have implemented our method as part of the OptiX SDK [184], which we have enhanced with volume rendering abilities. OptiX provides a C++/CUDA-based programming interface, which is specialized for ray-tracing applications. Because several materials, like the glass effect, which has been used in this work, are already implemented in the SDK, we only had to extend the framework to include a volume material and specialized ray generation programs (*cameras*) to support our method.

In OptiX, a simple ray-tracing program normally consists of a combination of a *hit function*, a *trace function*, a *miss function*, and a *camera* for the ray setup. The main functions are executed per ray. The *hit function* is used to intersect rays with object surfaces in the scene. The *trace function* evaluates the color contribution of a ray between two intersections, and the *miss function* fills rays that hit no geometry with a defined background color. These functions are implemented in separate CUDA files, which are preprocessed by the OptiX SDK. Using the Optix engine allows us to soften the border between ray-tracing and ray-casting and enables a combination of complex ray-traced materials with volumetric materials as shown in the generic example in Figure 4.5.

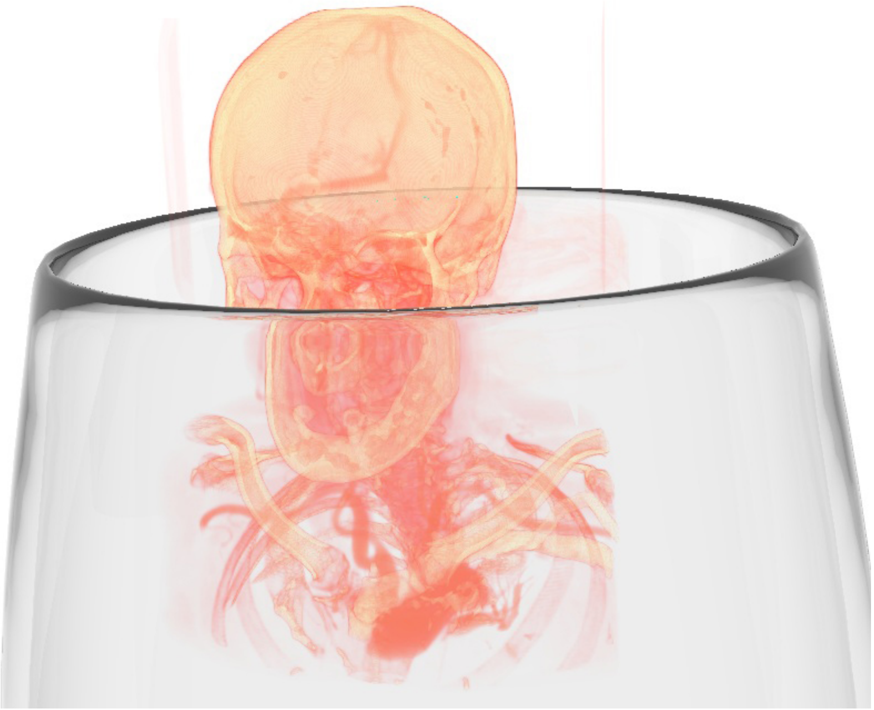


Figure 4.5: A generic showcase for the new possibilities using Nivida's OptiX engine as ray-based image synthesis system. We extended that engine to combine highly complex refractive/reflective materials with conventional volume rendering. Here the volumetric material is rendered inside a part of a wineglass to emphasize the refraction effects achieved by ray-tracing.

4.6.1 Feature preserving adaptive sampling camera

The rays are set up by a ray generation program, which can be seen as a *camera*. We use a pinhole camera model as the basis for our implementation and alter the ray generation scheme by our adaptive approach. The feature frame is rendered by using an extended version of the publicly available framework provided by [52], which is based on the Princeton *Trimesh2* library. To provide high frame rates for very complex objects, we extended this library with GPU-accelerated calculations. Therefore, we moved all per-frame calculations (e.g. $\hat{n}(p) \cdot \hat{v}(p)$) to CUDA kernel functions and attached the line-output to an OpenGL Vertex-Buffer object. This vertex buffer is subsequently rendered into an *OpenGL Framebuffer object*, which is concurrently mapped as a texture in the OptiX context. The ray setup is then done according to this texture as described in Section 4.3.2.

For volume rendering, we have also attempted a direct feature line extraction as proposed by Burns et al. [31]. Experiments with the Burns system have shown that the frame rates are not as high as those obtained with iso-surfaces, which are used in the DeCarlo system, especially for very large volumes and multiple transfer function peaks. This fact can be explained by the difference in the order of complexity when processing a surface

($O(n^2)$) compared with when processing a volume dataset ($O(n^3)$). Because iso-surfaces for the feature frame generation only have to be recalculated when the transfer function changes, we have decided to use the feature extraction approach from De Carlo et al. However, it would also be possible to use the approach of Burns et al. because it also shows frame rates that are high enough to meet our two-pass rendering criteria.

4.6.2 Volume rendering

We have implemented a standard ray-casting approach as a *material trace function*. In contrast to tight-fitting bounding geometry volume rendering systems, the volume bounding geometry can be a simple cube or a sphere instead of a tight fitting one. This detail reduces the necessary intersection calculations and maximizes the thread coherence, which was stated by [184] to be one of the most important factors for an efficient execution. Because every ray can store a certain amount of payload, we save the entrance point and the exit point of the *hit function* in every ray's payload structure. After transforming these two points to the volume object space, we let every ray accumulate all of the values in between, depending on the given transfer function. In addition, the values are shaded according to the Phong illumination model depending on the approximated volume gradient.

Mesh extraction Our method requires a smooth surface mesh for feature rendering. At that stage, we distinguished between a pure geometric input for ray-tracing applications and volumetric data sets for a direct volume ray-casting. In the first case, the input mesh can be used directly by the feature extraction step. The second case requires an intermediate step depending on the used volume transfer function and the volume histogram. In our case, a transfer function is defined by several color gradients, mapping a certain intensity range to a defined color and opacity. One can also define high dimensional transfer functions for a better visual result (e.g., using the gradient magnitude as a second dimension, as proposed by [113]). However, for an iso-surface extraction, the peak value of the direct (1D) mapping gradients or the projection of the color gradient's opacity peak to the intensity axis for high-dimensional transfer functions together with the peak of the volume histogram is sufficient. Consequently, we define the necessary iso-values for a multi-iso-surface extraction as the highest peaks of the volume's histogram if the transfer function is not zero at that position. We use a fast GPU-accelerated marching cubes implementation (CUDA version of the [133] algorithm) to extract the iso-surfaces whenever the transfer function changed. Because the resulting meshes are over-tessellated, we simplify this mesh with a GPU-based simplification method as described in the following paragraphs.

Mesh preparation The quality and size of surface meshes, used as inputs for our algorithm, vary dramatically. Ray-tracing applications are often applied to high quality meshes with hundreds of thousands of triangles. The iso-surface meshes extracted from volumes are known to be noisy and often contain lots of small triangles, which can be merged without a loss in quality. We thus use a combination of mesh smoothing [220] and mesh simplification [134] to generate meshes that fulfill our demands: (a) the mesh contains little noise, (b) the number of faces is low enough to generate the feature buffer quickly, and (c) main features from the original mesh are conserved at the according position.

Our algorithm successively applies Taubin smoothing, mesh simplification and another instance of Taubin smoothing. The first smoothing step is especially important for isosurfaces extracted from a volume. Taubin smoothing preserves the volume of the mesh and thus also conserves the location of remaining features. Our implementation of the mesh simplification method is run once per mesh as a preprocessing step and does not include any view-dependent simplifications. A static simplification based on a fixed error metric turned out to be sufficient for our demands. Both algorithms allow a highly parallel GPU-based implementation, which enables low latency on input data changes. The overall process takes up to a second, depending on the mesh complexity and the number of peaks in the transfer function.

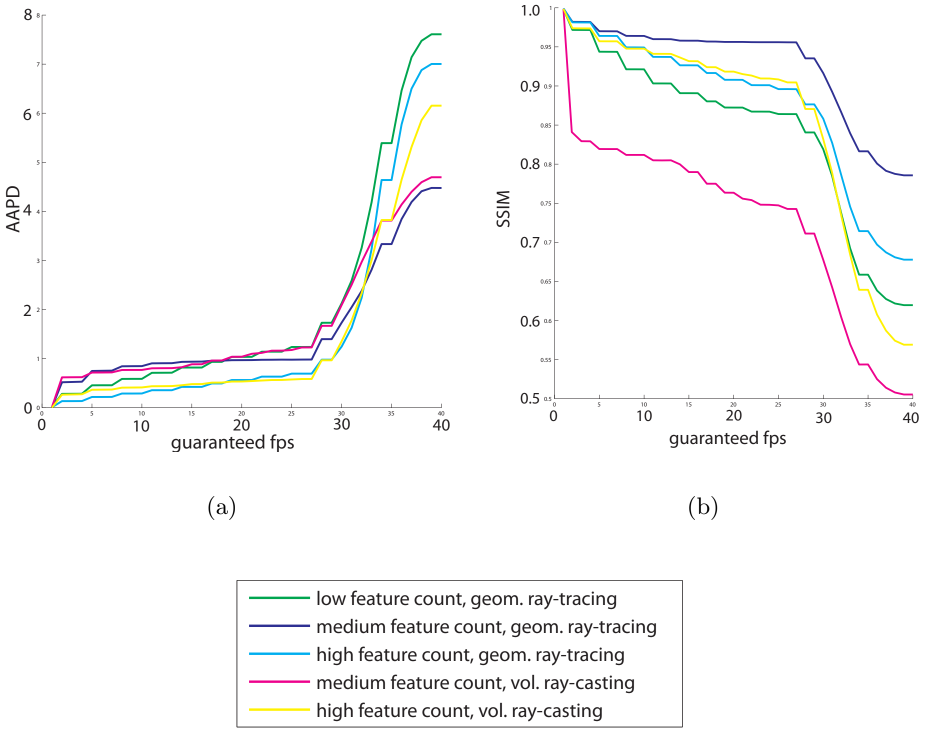
4.7 Results

In Section 4.5, we present our results on how much a certain object feature can improve the visual quality. In this section, we evaluate the performance of the overall system. Our test system is equipped with an Intel i7 Processor, 6 GB System memory and an Nvidia Quadro 6000 graphics card. Figure 4.6 shows the increase of the AAPD (a) and the decrease of the SSIM (b) for increasing guaranteed frame rates.

The feature lookup texture can be rendered with up to 1000 frames per second on a modern graphics workstation in a moderate viewport and it does not need to be of the same size as the render frame. Therefore, the computation time for this step can be neglected. For quality estimation, we use the fractal pattern interpolation image reconstruction method, as described in Section 4.3.4. Table 4.1 gives an overview of the overhead computation times of our method compared with the unaltered ground truth. Figures 4.7 and 4.9 show the decrease of quality with an increasing frame rate demand. The image quality remains stable as long as the frame rate is reasonably adjusted. Figure 4.7 also shows the pixels that are required to calculate a full ray traversal and compares our image reconstruction method to a regular subsampling with linear interpolation. Figure 4.9 shows the quality decrease for a volumetric object.

4.8 Discussion

This Chapter presents a novel method for integrating NPR features into a rendering environment as a quality hint for the required granularity during a ray-based rendering without frame-to-frame coherence requirements. We show that higher frame rates are achievable during a scene interaction without a severe loss of image quality. Our method outperforms



(a)

(b)

Figure 4.6: These plots show the increase of the AAPD (a) and the decrease of the SSIM (b) for increasing guaranteed frame rates averaged over different data sets. The viewport for this test was 1024 × 768. Note that, after a certain guaranteed frame rate, the time frame becomes too short to render the important features. This issue becomes apparent in the plots by the bend at approximately 35 fps.

the state-of-the-art implementation of adaptive rendering, for example, delivered with the OptiX SDK, in terms of speed and quality. It also reflects perceptual features more accurately due to its use of object space feature extraction than comparable approaches in image space.

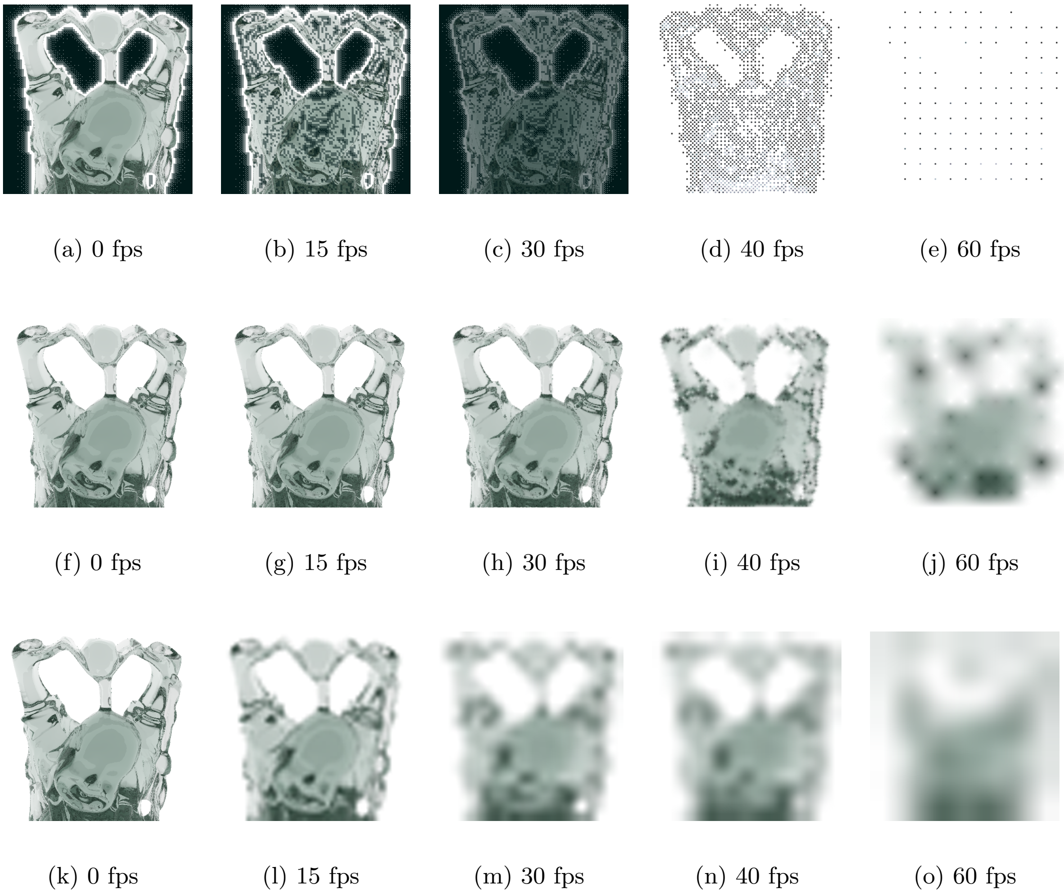
We have performed a quantitative evaluation of the perceptual features to determine their impact on the visual quality and to show which features are best suited for adaptive ray-based image generation. Our algorithm can therefore also be used to achieve guaranteed frame rates by sorting the image pixels according to the feature priorities, which substantiate *TH2* fully. Our algorithm is mainly intended for highly complex ray-based calculations, such as volume rendering, and for systems that require that object rendering does not occupy the whole computation unit (e.g., additional GPU-based simulation and segmentation).

	geometry [*ms*]	**volume** [*ms*]	**av. ray count** [*#rays*]
feature frame	2	3	-
rays on contours	11	12	34.970
rays on ridges	12	14	37.208
rays on silhouette	5	8	9.373
rays on sug. highlights	10	19	23.043
rays on sug. contours	6	11	18.793
rays on valleys	5	7	10.288
reconstruction	15	15	17.756
sum	66	89	151.431
ground truth	208	251	2.457.600

Table 4.1: An overview over the average rendering times for each step and different objects using our approach on our test system (variance $< 1\%$). For *geometry*, we tested the Stanford Dragon, Buddah, and Bunny datasets, our piggy dataset and some simpler drinking glass meshes, as shown in the accompanying video with the ray-traced 'glass' material from Figure 4.1(c). For *volume*, we tested the 512^3 datasets, *MANIX* and *FEET*, as shown in Figure 4.1(b) and (d) with different transfer functions. The measured times refer to a computation within a 1440×900 viewport. Note that not all rays on feature lines have to be computed. To obtain high guaranteed frame rates that maintain a visual appealing result, low priority features might be omitted by our algorithm from Section 4.4. The average ray count also varies with different viewports.

4.8.1 Image space saliency

Because pure object space features do not incorporate high frequencies in textures and hard shadows or reflections/refractions, we carried out additional experiments to enhance the feature-buffer with this information. If required, users can activate the saliency information for the features buffer. Therefore, we rendered the unaltered scene at a lower resolution and calculated the saliency of this image before the feature frame is generated. For that step, we make use of the OptiX rendering engine's ability to render small scenes at full resolution with very high frame rates. If the image space addition is selected, a third render-pass is added in the beginning of our method, which rendered the unaltered scene at a very low resolution (typically a factor of 4 to 8 smaller, depending on the graphics hardware used and target resolution). Subsequently, a GLSL shader implementation of the method from [102] is applied to compute the image's saliency. The feature buffer is then enhanced with a linearly interpolated version of this image. The saliency information defines some additional lower levels within our priority queue. This step usually requires 10-20 ms. However, for this step, the required rays are reused in the reconstruction step to further


(a) 0 fps (b) 15 fps (c) 30 fps (d) 40 fps (e) 60 fps

(f) 0 fps (g) 15 fps (h) 30 fps (i) 40 fps (j) 60 fps

(k) 0 fps (l) 15 fps (m) 30 fps (n) 40 fps (o) 60 fps

Figure 4.7: This figure illustrates the decreasing ray count with increasing requested guaranteed frame rates for a 1024×768 viewport. The top row shows the actual pixels that have been traced, and the middle row shows the result with our fractal pattern interpolation scheme. (i) and (j) of the top row are the edge images of the traced pixels to emphasize their positions in the printed versions of this thesis. The bottom row shows the results using a regular sub-sampling pattern with a linear interpolation for comparison.

improve the image quality. As an example, the feature buffer including the proposed saliency measure is shown together with the resulting reconstruction in Figure 4.8.

The visual saliency information of a small scene viewport is only a rough approximation of the full resolution; hence, we assigned the lowest priorities to the results of this preprocessing step. However, adding an image space method narrows the possible speedup of our approach. Our results show that for most objects (including the heavily shaded ones), the object space feature extraction step is sufficient for a visually pleasing rendering during the interaction step. Our image reconstruction scheme converges progressively within a few frames as soon as the scene interaction stops. Therefore, missing shading details are added quickly with low visual annoyance. However, because of the computational overhead, the maximum performance boost, with respect to the maximum guaranteed frame rate with a visually acceptable result, decreases by approximately 10 frames per second. Using only the saliency information (without the object space features) results either in a very bad reconstruction result or a low speedup because of the rough image approximation and the low resolution of the saliency frame.

We have also evaluated the possibilities of image warping [95] for reusing the saliency information from previous (fully resolved) frames. This approach has shown promising results for a limited scene interaction and few object disocclusions, thus, for scenes with a high frame-to-frame coherence. However, image warping and saliency computations are computationally more expensive than the pure feature frame generation in object space. This fact lowers the maximum reachable performance; however, a pure image space method is not as accurate as the proposed object space approach.

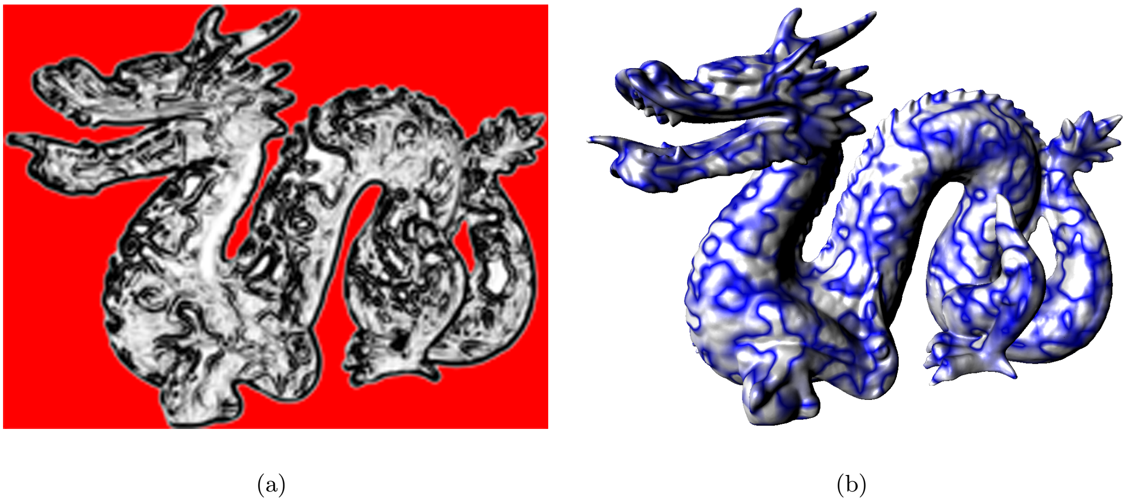
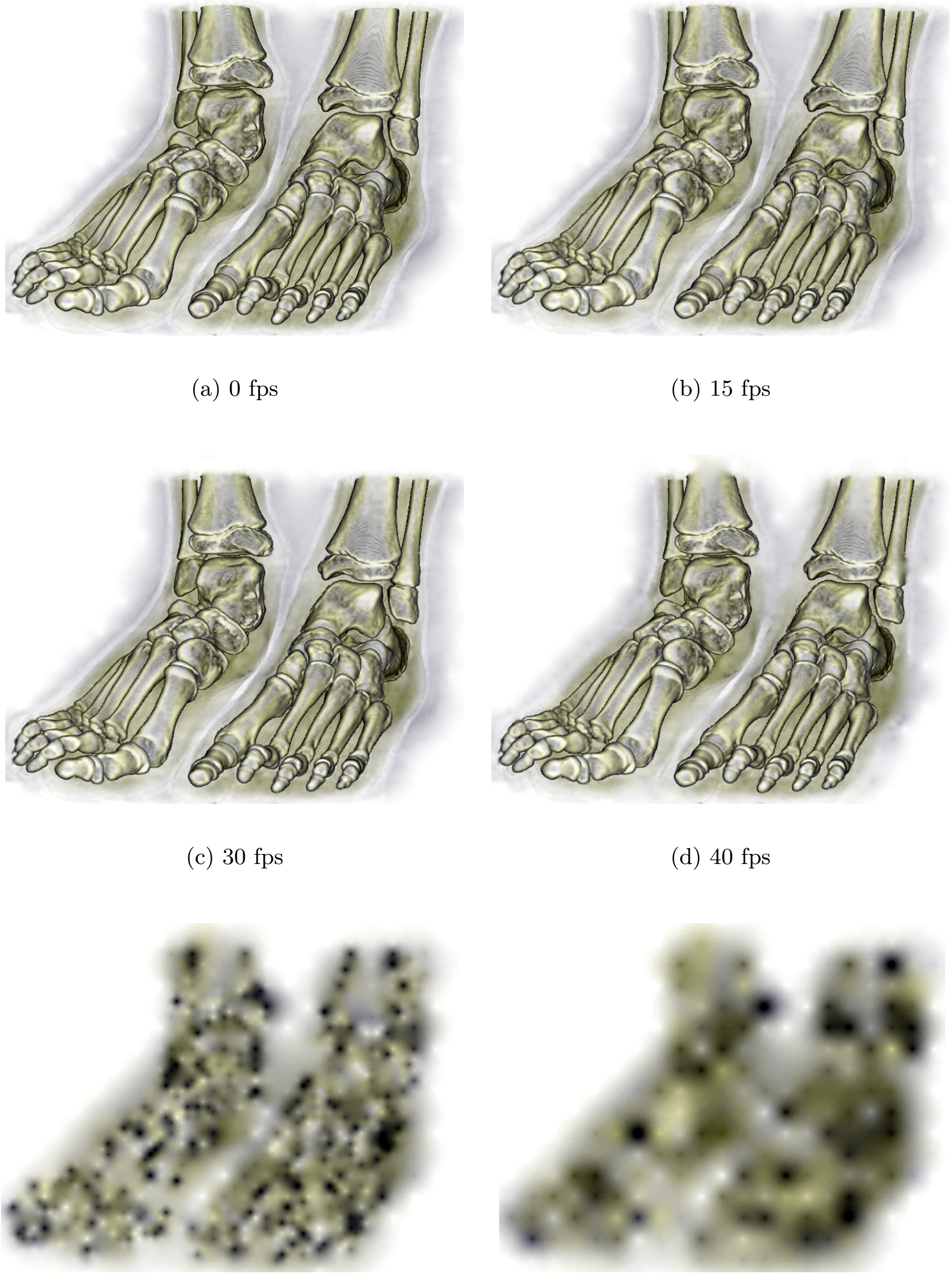


Figure 4.8: The visual saliency information can be included into our feature buffer (a) for the reconstruction of a ray-traced textured object (b). The saliency information was computed on a six-fold-smaller ground truth, the processed rays were reused for the final reconstruction. Colors in (a) have been encoded according to our definitions from Section 4.3.1.



(a) 0 fps

(b) 15 fps

(c) 30 fps

(d) 40 fps

(e) 50 fps

(f) 60 fps

Figure 4.9: This figure illustrates the decreasing quality with increasing requested guaranteed frame rates for a 512^3 volumetric dataset in a 1024×768 viewport.

Chapter 5

Specialization of ray-casting kernels

In this Chapter, we demonstrate the applicability of certain specializations of ray-casting kernels for image-based similarity measures to pose optimization problems performed on fast parallel stream processors (i.e., GPUs). Therefore, we utilize an analytically computed gradient by means of automatic differentiation (AD) for an iterative optimizer. To substantiate *TH3* we render directly the gradient image of the volumetric data with automatically generated shader-based ray-casting kernel code. This results in a significant speedup and accuracy improvement compared to numerical gradient approximations (such as forward differences). The motivating application for this chapter is X-Ray-based 2D/3D registration, *i.e.*, Computed Tomography (CT) and C-Arm registration which is outlined in more detail in Section 7. However, the use of AD techniques (*cf.* Section 2.2.6) is in fact not limited to a *e.g.* 2D/3D registration, but opens a new dimension to accelerate any registration task where an objective function has to be optimized with respect to the transformation parameters.

Variational methods [38] are among the most successful methods to solve a number of computer vision problems (including registration). Basically, variational methods aim to minimize an energy functional which is designed to appropriately describe the behavior of a certain model. The variational approach provides therefore a way to implement non-supervised processes by just looking for the minimizer of the energy functional.

Minimizing these energies is usually performed by calculating the solution of the Euler-Lagrange equations for the energy functional. For quite involved models, such as the energy functional we use in our 2D/3D registration task, their analytical differentiation is not a trivial task and is moreover error prone. Therefore many people bypass this issue by computing the derivatives by means of a numerical approximation. This is clearly not optimal and can lead to inaccurate results.

In [190] automatic differentiation methods have been studied in the context of computer vision problems (denoising, segmentation, registration). The basic idea is to discretize the energy functional and then apply automatic differentiation techniques to compute the exact derivatives of the algorithmic representation of the energy functional.

Recently, graphics processing units (GPUs) have become increasingly flexible and can today be used for a broad range of applications, including computer vision applications. In [189] it has been shown that GPUs are particularly suitable to compute variational methods. Speedups of several orders of magnitude can be achieved.

In this chapter we propose to take advantage of both automatic differentiation and the immense computational power of GPUs. Due to the limited computational flexibility of GPUs, standard automatic differentiation techniques can not be applied. In this chapter we therefore study options of how to adapt automatic differentiation methods for GPUs. We demonstrate this by means of medical 2D/3D registration:

Given a pre-operatively acquired CT volume of a patient and a set of X-ray images of the same patient taken during the intervention, we can search for a rigid transformation of the CT volume such that it is optimally aligned with the observed images (*2D/3D registration*).

Both image-based and feature-based approaches exist to quantify this alignment, summarized by [138, 255]. In the former case, the alignment is computed in a two step procedure. First, the acquisition of an X-ray image has to be simulated by a so-called *digitally reconstructed radiograph* (DRR) which is a specialization of a generic ray-casting kernel as outlined in Algorithm 2. This involves traversing the entire CT volume (at a sufficient resolution) along the same paths which are taken by the X-ray radiation according to the projection geometry of the C-arm. Once the DRRs are available, they have to be compared with the real X-ray images. To do so, we define a *similarity measure* which is maximal for two optimally aligned images. For an X-Ray-based system (including the CT volume acquisition), the *normalized cross correlation* is sufficient, since the X-ray images and DRRs are (almost) affinely related for perfectly registered data sets.

However, finding the optimal solution of this 2D/3D registration problem is not trivial, since the problem is not convex and might contain hundreds of local minima. Therefore often initialization techniques are used to place the virtual scene near the optimal solution. Such initialization can be achieved by using an artificial target with known parameters which is visible in at least one modality as for example shown in Section 7.

5.1 Pose optimization

After the pose is initialized, the remaining rigid registration task as outlined in Fig. 5.1 can be formulated as an optimization problem, where we try to find the parameter vector $\mathbf{x}_{\text{opt}} \in \mathbb{R}^6$ of a rigid transformation in 3D such that the n projections $I_i(\mathbf{x})$ of our CT volume (i.e., the DRRs) are optimally aligned with a set of n X-Ray images J_i, $i = 1 \dots n$.

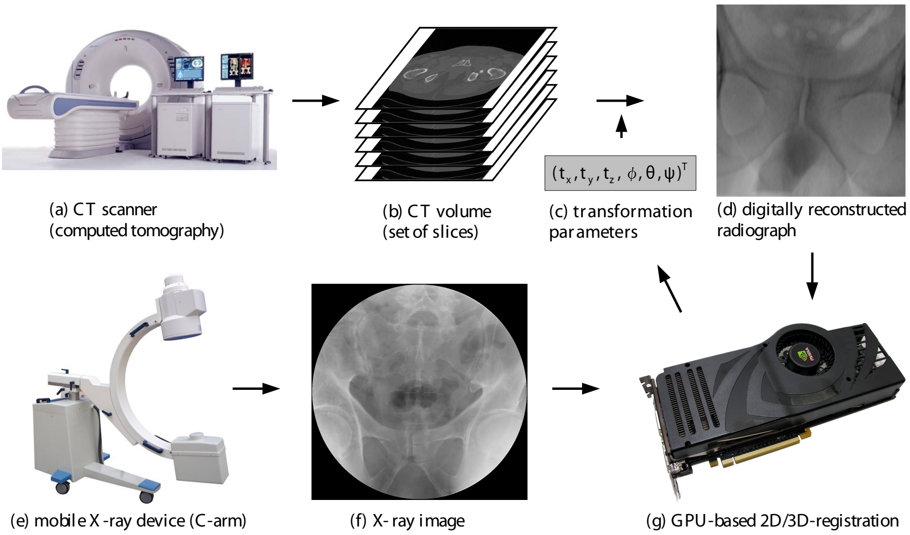

(a) CT scanner
(computed tomography)
(b) CT volume
(set of slices)
(t_x, t_y, t_z, φ, θ, ψ)^T
(c) transformation
parameters
(d) digitally reconstructed
radiograph
(e) mobile X-ray device (C-arm)
(f) X-ray image
(g) GPU-based 2D/3D-registration

Figure 5.1: Schematic workflow of 2D/3D registration. Before the intervention, a CT scanner (a) is used to obtain a volumetric data set (b) of the patient. With an initial estimate for the transformation parameters (c), which we seek to optimize, a DRR (d) is created from the volume data. During the intervention, a C-arm (e) is used to obtain X-Ray images (f) of the patient. The registration procedure (g) compares the DRR and X-Ray image and updates the transformation parameters until the DRR is optimally aligned with the X-Ray image (i.e., our distance measure is minimized). We use three DRR/X-Ray image pairs for better accuracy (only one is shown here).

Formally, we try to solve

$$\mathbf{x}_{\text{opt}} = \underset{\mathbf{x}}{\operatorname{argmin}} E(\mathbf{x}), \quad E(\mathbf{x}) = \sum_i D(I_i(\mathbf{x}), J_i) \tag{5.1}$$

where E is our *cost function*, and $D(I_i(\mathbf{x}), J_i)$ computes the non-negative distance measure between two images, which is zero for a pair of perfectly aligned images. Note that the X-Ray images J_i do not depend on the parameter vector $\mathbf{x}$, but each X-Ray image has an *exterior camera orientation* [92] associated with it (describing the recording geometry), which must be used to compute the corresponding DRR $I_i(\mathbf{x})$. The computation of the camera orientation is out of the scope of this thesis. In the following, we only deal with a single pair of images and leave the summation in Equation (5.1) as a final (implicit) processing step.

5.2 GPU-based digitally reconstructed radiograph from volumetric data

To adequately simulate the process of X-Ray image acquisition, one has to follow the basics of the image intensities of the radiograph. The X-Ray intensity I reaching the detector at a pixel $(u,v) \in \Omega$ in image space can be expressed using the following physically-based model [245]:

$$I_{\text{phys}}(u,v) = \int_0^{E_{\max}} I_0(E) \exp\left(-\int_{r(u,v)} \mu(x,y,z,E)dr\right) dE, \tag{5.2}$$

where $I_0(E)$ denotes the incident X-Ray energy spectrum, $r(u,v)$ a ray from the X-Ray source to the image point (u,v), and $\mu(x,y,z,E)$ the energy dependent attenuation at a point (x,y,z) in space. The second integral represents the attenuation of an incident energy $I_0(E)$ along the ray $r(u,v)$. The integral over E incorporates the energy spectrum of X-Ray cameras.

The above expression can be simplified in several ways [245]. First, the X-Ray source is mostly modeled to be monochromatic and the attenuation to act upon an effective energy E_{eff}. Second, due to the fact that X-Ray devices usually provide the logarithm of the measured X-Ray intensities, we can further simplify Eq. (5.2) by taking its logarithm. Finally, when using more elaborate similarity measures which are invariant with respect to constant additions and multiplications [139], we can omit constant terms and obtain the following pixel intensities for our DRR image:

$$I(u,v) = \int_{r(u,v)} \mu(x,y,z,E_{\text{eff}})dr \tag{5.3}$$

The pseudo code of the DRR rendering algorithm under the parameter vector $\mathbf{x} = (t_x,t_y,t_z,\phi,\theta,\psi)^T$ is given in Alg. 3. The rigid body transformation we seek to optimize in Fig. 5.1(c) consists of a translation $\mathbf{t} = (t_x,t_y,t_z)^T$ and a rotation $\mathbf{R} = \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi$ given in terms of three Euler angles ϕ, θ, and ψ, where

$$\mathbf{R}_\phi = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{R}_\theta = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}, \quad \mathbf{R}_\psi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{pmatrix}$$

5.3 Similarity measure

We investigate the *normalized cross correlation*, which verifies the existence of an *affine relationship* between the intensities in the images. It provides information about the extent

Algorithm 3 The DRR rendering algorithm. All transformations are given as 4×4 matrices in homogeneous coordinates, where $\mathbf{T}$ and $\mathbf{R}$ are translation and rotation, respectively, $\mathbf{C}$ describes the center of rotation, and $\mathbf{H}$ is the window-to-object coordinates transformation. d is the sampling distance in window coordinates. Ω is the set of pixels (u,v) which are covered by the projection of the volume, and $\Omega_{(u,v)}$ is the set of sampling positions along the ray through pixel (u,v) which intersect the volume. $\mu(\mathbf{p})$ is the volume sample at point $\mathbf{p}$.

```
Require: $\mathbf{C},\mathbf{R},\mathbf{T},\mathbf{H} \in M_4(\mathbb{R});\ d \in \mathbb{R}^+$
for all $(u,v) \in \Omega$ do
    $\mathbf{p}_{\text{win}}^{(0)} = (u,v,d/2,1)^T$                          ▷ ray start position in window coordinates
    $\mathbf{d}_{\text{win}} = (0,0,d,1)^T$                                  ▷ ray step vector in window coordinates
    $\mathbf{p}_{\text{obj}}^{(0)} = \mathbf{CR}^{-1}\mathbf{T}^{-1}\mathbf{C}^{-1}\mathbf{Hp}_{\text{win}}^{(0)}$    ▷ ray start position in object coordinates
    $\mathbf{d}_{\text{obj}} = \mathbf{CR}^{-1}\mathbf{T}^{-1}\mathbf{C}^{-1}\mathbf{Hd}_{\text{win}}$      ▷ ray step vector in object coordinates
    $I(u,v) = 0$
    for all $t \in \Omega_{(u,v)}$ do
        $I(u,v) = I(u,v) + \mu(\mathbf{p}_{\text{obj}}^{(0)} + t\mathbf{d}_{\text{obj}})$   ▷ take volume sample and update intensity
    end for
end for
```

and the sign by which two random variables (I and J in our case) are linearly related:

$$\mathrm{NCC}(I,J) = \frac{\sum_{(u,v)\in\Omega} \left(I(u,v) - \bar{I}\right)\left(J(u,v) - \bar{J}\right)}{\sqrt{\sum_{(u,v)\in\Omega} \left(I(u,v) - \bar{I}\right)^2}\sqrt{\sum_{(u,v)\in\Omega} \left(J(u,v) - \bar{J}\right)^2}} \tag{5.4}$$

Optimal alignment between the DRR image I and the X-Ray image J is achieved for $\mathrm{NCC}(I,J) = -1$ since we use Eq. (5.3) for DRR computation, but the actual X-Ray image acquisition is governed by Eq. (5.2). Our distance measure from Eq. (5.1) is therefore simply

$$D(I_i(\mathbf{x}), J_i) = \mathrm{NCC}(I_i(\mathbf{x}), J_i) + 1.$$

5.4 Automatic differentiation for a hybrid CPU/GPU setup

In this section we address several issues that must be considered when applying AD techniques to generate code that will be executed in a hybrid CPU/GPU setup for maximum performance. Note that we have used Nvidia's Cg shader language for the implementation of this method because of the lack of several necessary language features for AD of CUDA to the time of development.

Computing Resources In order to properly execute both the original algorithm and its derivative, we must determine which hardware component is best suited for each section of the algorithm. The following components are available to us:

- the host's CPU(s),
- the GPU's rasterizer, and
- the GPU's shader processors.

The CPU's involvement in the numeric computations is marginal (its main task is to control the workflow), we therefore did not consider the use of more than one CPU.

Due to the stream programming model in Cg, the output of the shader program is restricted to a fixed (and small) number of values, sufficient however to write a single pixel (and the corresponding adjoint variables) of the DRR. Therefore looping over all pixels in the image (equivalent to $(u,v) \in \Omega$ in Alg. 3) is left to the rasterizer.

The core of the computation is the traversal of the individual rays through the volume (equivalent to $t \in \Omega_{(u,v)}$ in Alg. 3). Since the CT volume is constant, data can be read from (read-only) texture memory, and the innermost loop can be executed by the shader processors. This procedure has good cache coherence since a bundle of rays through neighboring pixels will likely hit adjacent voxels, too, if the geometric resolutions of image and volume data and the sampling distance are chosen appropriately.

The loops in the similarity measurement code, corresponding to the sums in Eq. (5.4), are more difficult to handle, although the domain $(u,v) \in \Omega$ is the same as above. The reason is the data-interdependence between successive invocations of the loop body, which must not proceed before the previous run has updated the sum variables. We therefore employ a *reduction* technique [187], where the texture is repeatedly downsampled by a factor of two (n times for an image $2^n \times 2^n$ pixels large) until we end up with a single pixel representing the desired value.

Gradient Computation We need to compute the gradient of the (scalar-valued) cost function (Eq. 5.1) with respect to the transformation parameter vector $\mathbf{x} \in \mathbb{R}^6$. This can be done either by invoking the algorithm's forward mode derivative six times or by a single pass of the algorithm's reverse mode derivative. Every pass of the original and the derived algorithm (both in forward and reverse mode) requires access to all voxels visible under the current transformation parameters at least once (or even several times in case of texture cache misses). Since global memory traffic is the main bottleneck of many GPU-based algorithms, we choose reverse mode AD for our purposes to reduce the number of read operations from texture memory.

Reverse mode AD is known to be more difficult to implement than forward mode AD. Due to the above-mentioned memory limitations it is generally not possible to use techniques like taping [170] and TBR analysis [94, 171] to produce adjoint code in Cg. However, since the inner loop in Alg. 3 is simply a sum, we do not need to store intermediate values in the derivative code (i.e., the "TBR set" of our program is empty), hence a Cg implementation is feasible.

An additional benefit of the analytically computed gradient is its better numerical behavior. When computing a numerical approximation of the gradient (e.g., based on central differences), one has to carefully find a proper tradeoff between truncation error and cancellation error [19]. This is particularly true with our hardware and software platform (NVidia graphics cards and Cg), where the maximum available precision is 32 bit IEEE floating point.

Automatic Differentiation Issues The *operator overloading* technique [48] for generating the algorithm's derivative cannot be used in our case since it requires the target compiler to understand C++ syntax and semantics. However, both Cg and CUDA only support the C language (plus a few extensions not relevant here) in their current versions. Moreover, in order to apply reverse mode AD with operator overloading, the sequence of operations actually performed when executing the given algorithm must be recorded on a *tape* [48] and later reversed to compute the adjoints and finally the gradient. This approach cannot be used in Cg due to its limited memory access capabilities as explained in Sect. 2.2.4.

Therefore *source transformation* remains as the only viable option. We implemented a system which parses the code tree produced from C-code by the GNU C compiler and uses GiNaC [13] to calculate the symbolic derivatives of the individual expressions. It produces adjoint code in the Cg language, but is restricted to programs with an empty TBR set since correct TBR handling cannot be implemented in Cg anyway as stated above. The task distribution discussed in Sect. 5.4 was done manually since it requires special knowledge about the capabilities of the involved hardware components, which is difficult to generalize. Moreover, the derivative code contains two separate volume traversal passes, which can be rewritten in a single pass, reducing the number of volume texture accesses by 50%.

5.4.1 Registration with analytically derived gradients

We have chosen the L-BFGS-B [254] algorithm to accomplish this optimization task. This algorithm does not require any knowledge about the structure of the cost function, and it is possible to constrain the search to a given (rectangular) subset of the parameter space. To obtain a smoother cost function, we use *image pyramids* of the X-Ray image and the CT volume for rendering the DRRs [118, 137, 243]. Five levels are used for both the image and volume pyramid.

The optimization algorithm starts at the coarsest resolution level. After convergence, the optimizer is initialized with the parameters found in the precedent pass and operates on the next higher level of resolution. The procedure is repeated until the highest level of resolution is reached. It is evident that a *multi-scale* approach did not only smooth out the cost function, but effects also a significant registration speedup [137].

The value and gradient of our cost function Eq. (5.1) has to be provided to the L-BFGS-B optimizer. Both are computed on the GPU, where at the core of them a single-pass GPU raycasting method for uniform grids [88] is employed. The DRR computation and similarity measurement algorithms are converted to the corresponding derivative code through automatic differentiation. Due to the chain rule for differentiation, the gradient of the CT volume appears as an expression in this code. We use two methods to compute it. First, we approximate the volume gradient "on-the-fly" by central differences, which requires 12 volume samples. Second, we precompute the gradients and store them in texture memory along with the volume. This obviously requires more memory, but is twice as fast as "on-the-fly" gradient computation.

5.5 Results

Originally we have tested our implementation with an Intel® Core™2 Quad processor running at 2.66 GHz and an NVidia® GeForce® 8800 GTX graphics boards with 768 MB of GPU memory. A *Fortran* implementation of the *L-BFGS-B* algorithm is used for optimization and the *Cg* language [144] to perform calculations on the GPU(s). For this thesis we repeated the tests on state-of-the-art hardware. We have used an Intel QuadCore 3.06 Ghz System with 12 GB RAM and distributed the work on two Nvidia GTX 480 graphics cards. Although the algorithm's accuracy is still the same, the execution time is approximately two times faster.

A visualization of the contributions to the gradient of each $(u,v) \in \Omega$ (i.e., each pixel in the image) for the 2D/3D registration with a simple object is shown in Fig. 5.2. Figures 5.2(b) and 5.2(d) are the inputs to the final reduction pass, the sums over all pixels are the individual components of the gradient vector ∇E. The volume was sampled with 64^3 voxels, average registration time was 1.1 seconds on a GeForce GTX480 graphics card.

Figure 5.3 illustrates the convergence behavior of our 2D/3D registration method. The method performs very reliably for an initial error of less than 20mm, where the final registration error is less than 1mm in most experiments. For a CT volume data set with $512 \times 512 \times 288$ voxels and three X-Ray images (512×512 pixels each), the average registration time is 20 seconds, which is 5.1 times faster than with numerical approximation of the gradient. This confirms previous results that the computation of the gradient does not take significantly longer than the evaluation of the underlying function [14].

Further we have carried out experiments with the CT scan (512×512×288 voxels, 1×1×0.6mm voxel size) of a patient's abdomen and three corresponding X-ray images (512×512 pixels each). The step size for our ray-caster is 1 mm.

In order to assess the quality of the alignment of our implementation and to compare it to existing approaches, we apply a standardized method to determine the registration error. The *mean target registration error* (mTRE) is a common measure when no salient features are available in the images. Van de Kraats et al. [228] proposed an evaluation of the mTRE by means of a set of evenly distributed points inside a volume of interest. It is calculated as the mean distance between the points mapped to the world by the estimated transformation and the ground truth transformation.

For our case, mTRE ground truth for a pelvic bone dataset was computed from metal

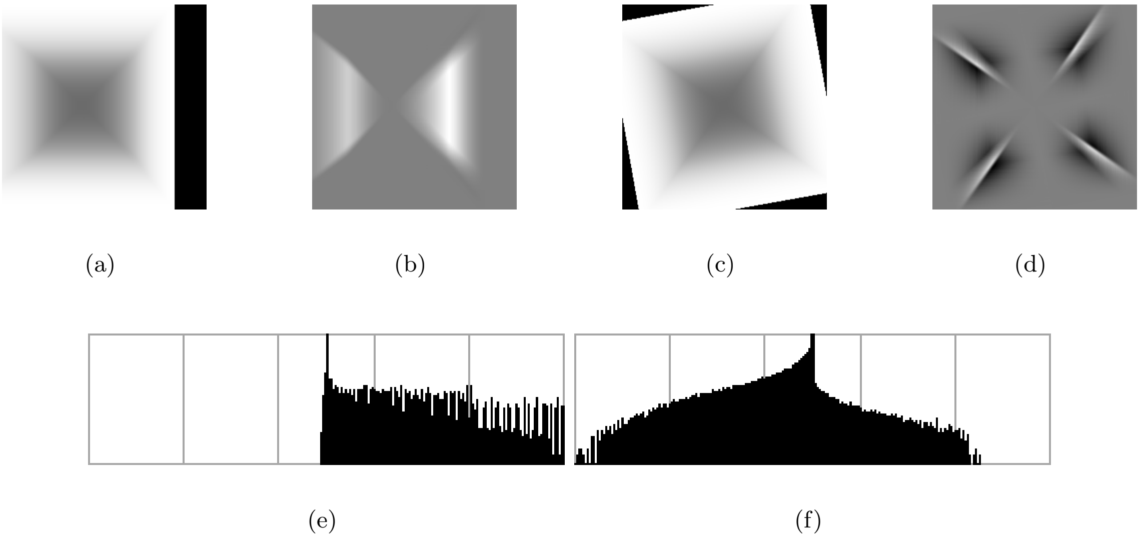


Figure 5.2: A simple object with density distribution $\mu(x,y,z) = 1 - \max(|x|,|y|,|z|)$, $x,y,z \in [-1,1]$, is translated to the left ($\Delta x < 0$) in (a), and the per-pixel contributions to the component of the parameter vector gradient ∇E corresponding to translation in x-direction are visualized in (b), where gray is zero, darker is negative (not present in the image), and brighter is positive. The histogram (e) of the gradient image (b) shows only positive entries (center is zero), indicating an overall positive value of the gradient in x-direction to compensate for the initial translation. In a similar way, the object is rotated ($\Delta\varphi > 0$) in (c), and the gradient contributions with respect to rotation around the z-axis are shown in (d). Its histogram (f) shows a prevalence of negative values, indicating an overall negative rotation gradient around the z-axis to compensate for the initial rotation. All other histograms are symmetric (not shown here), indicating a value of zero for the respective gradient components.

beads attached to a dissected bone (localization in the X-Ray and CT data with subpixel/-voxel accuracy). For real patient data, the center of the cluster of successful runs is used as ground truth since no additional markers can be attached to the patients' pelvic bones.

We define our region of interest as a cube of size $10 \times 10 \times 10$ centimeters, where we evenly distribute 1000 points on a regular grid. To determine the *capture range* of the registration algorithm, the criteria for a successful registration and the percentage of allowed failures have to be defined. We consider registrations with a mTRE below 1.5 millimeters as successful. We then define the capture range as the range of initial mTRE within which 90 percent of all registrations are successful. We use the mTRE as a measure for the initial displacement [228]. Random starting positions are uniformly divided into a set of initial mTRE intervals.

We have evaluated 70 initial mTRE intervals containing 10 random starting positions. Our approach achieves an average final mTRE of 0.6 millimeters for all successful registra-

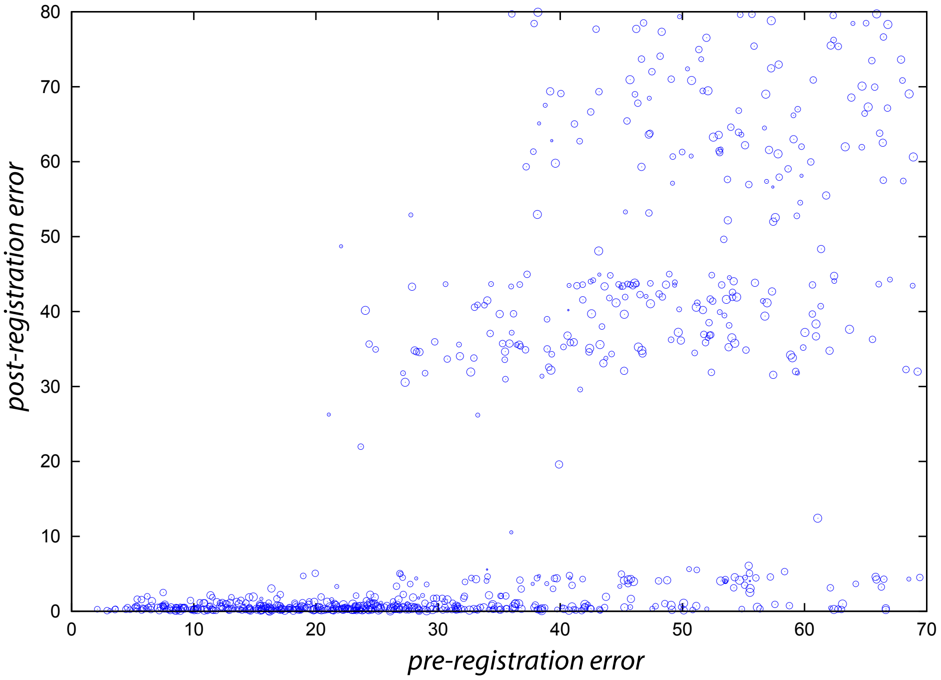
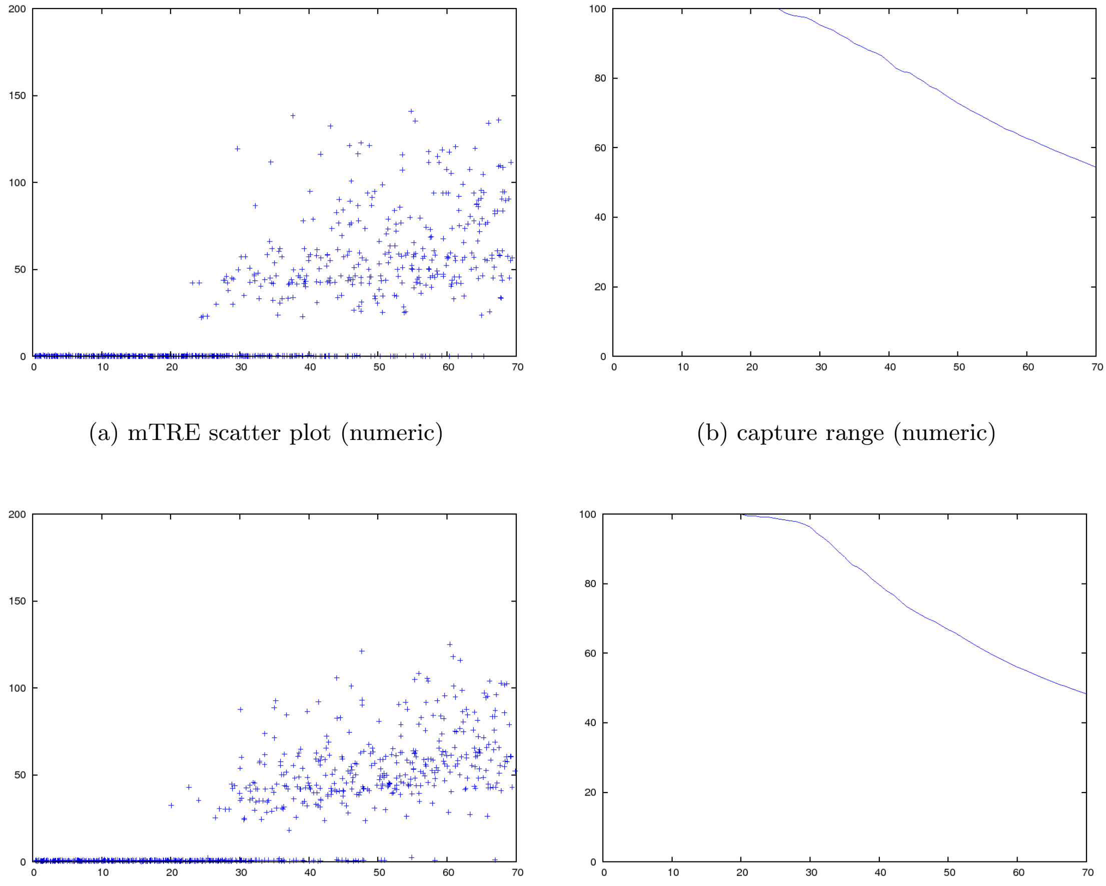


Figure 5.3: Scatter plot of the registration error [229] before (x-axis) and after (y-axis) the registration procedure, the dot size indicates the initial rotation.

	mTRE (mm)		Success (%)		C.-range (mm)	
Dataset	**NA**	**AD**	**NA**	**AD**	**NA**	**AD**
Abdomen	0.2	0.6	54.4	48.4	35	33
Pelvis	0.1	0.24	74.3	73.4	48	49

Table 5.1: The accuracy is given as the average final mTRE of successful registrations in millimeters. The capture range represents the maximum initial mTRE for which 90 percent of all registrations were successful. All values are provided for numerical approximation of the gradient (**NA**) and algorithmic differentiation (**AD**).

tions. A total of 338 (48.3%) registrations were successful. Figures 5.4(a) and 5.4(c) show the relationship between the initial and the final mTRE of single registrations with scatter plots. Each marker represents one experimental registration. Its x-coordinate specifies the initial mTRE, where the y-coordinate depicts the respective final mTRE. Figures 5.4(b) and 5.4(d) show the progression of the percentage of successful registrations. Each discrete position on the x-axis represents an upper bound for the initial mTRE. The corresponding y-coordinate states the percentage of registrations within the initial mTRE range $[0,x]$ which converged successfully.



(a) mTRE scatter plot (numeric)

(b) capture range (numeric)

(c) mTRE scatter plot (analytic)

(d) capture range (analytic)

Figure 5.4: The mTRE scatter plot and capture range for the experiments conducted with the human abdomen data using numerical approximation of the gradient and our analytically computed gradient based on automatic differentiation.

Computing the gradient requires a mean of 0.70 seconds when approximating it numerically, and 0.29 seconds when using algorithmic differentiation. An average registration takes 68.7 seconds to converge for the former, and 28.5 seconds for the latter gradient computation approach. Similar experiments are conducted with a dissected human pelvic bone from which any soft tissue was removed. Table 5.1 summarizes the results of the convergence and accuracy measurements, the time measurements for our approach on our test system are given in Table 5.2.

All results reported so far were obtained with "on-the-fly" computation of the volume gradient. Due to the high memory requirements of the precomputed gradient method, we were not able to directly compare both methods at full resolution of the volume data with our available hardware. However, to assess the benefits of gradient precomputation, we

	X-ray size	Gradient (sec)				Registration (sec)		
Dataset	**(pixels)**	**NA**	**AD**	**NA/AD**	**AD/*f***	**NA**	**AD**	**NA/AD**
Abdomen	512×512	0.70	0.29	2.41	4.98	68.72	28.52	2.41
Pelvis	1024×1024	1.87	0.88	2.13	5.63	199.35	96.35	2.07

Table 5.2: Mean time (in seconds) required to calculate a gradient and carry out a registration using both numerical approximation (**NA**) and algorithmic differentiation (**AD**). The **NA/AD** columns depict the ratio between the time required for numerical approximation and algorithmic differentiation. The column **AD/*f*** states the ratio between the runtime of evaluating the gradient using AD and evaluating the similarity measure f.

	numerical approximation	analytic on-the-fly	analytic precomputed	cost function evaluation
average time (sec)	60.0	23.0	11.8	5.0
relative to function	12.0	4.6	2.36	1.0

Table 5.3: Average execution time of numerical gradient approximation by central differences, analytic gradient computation with on-the-fly volume gradient, and analytic gradient computation with precomputed volume gradient. The numbers in the column "cost function evaluation" are 1/12 of the values for the numerical approximation (by the definition of the central differences approach in the six-dimensional parameter space). The execution times are given as absolute numbers in seconds and relative to the runtime of the cost function evaluation.

downsample the volume by a factor of two, while leaving the sampling locations unchanged. The average registration times are given in Table 5.3. Note that the numbers are slightly smaller than the corresponding ones in Table 5.2 due to increased texture cache efficiency caused by the downsampling.

5.6 Discussion

We are able to provide an accurate, cheap, and fast way to estimate gradients analytically and directly from a specialized ray-casting kernel.

We have discussed an approach to apply AD techniques to the 2D/3D registration problem which frequently appears in medical applications. We have demonstrated how to work around the limitations of current graphics hardware and software, therefore being able to benefit from the tremendous computing capabilities of GPUs.

Our method is 5.1 times faster than its counterpart with numeric approximation of the cost function's gradient by means of central differences. Its performance and accuracy are sufficient for clinical applications such as surgery navigation. The registrations performed with dataset containing predominately bones and a low amount of soft tissue, are generally more accurate and robust compared to standard registration methods which substantiate *TH3*.

In our present work we accepted a certain degree of manual work to make the code produced by our source code transformation tool suitable for a hybrid CPU/GPU setup. It remains an open question whether this step can be done fully automatically. We need to formalize the conditions under which parts of the derivative code can run on the CPU or on the GPU. During first experiments it seemed that 2D/3D registration based on algorithmic differentiation suffers a loss of accuracy compared to numerical approximation of the gradient. This is partly due to the sensitivity of an exact gradient computation to noise, which misleads the descent towards the minimum. Filtering the volume with a Gaussian kernel has solved this problem. However, the final mean target registration error for successful registrations generally amounts to 0.6 millimeters and below, which is satisfactory for targeting locations within the human body. Both approaches are similar with respect to robustness (i.e., the extent of the initial displacement for which registrations still converge reliably). An initial displacement of the patient of about two centimeters is feasible with our approach for a successful automatic pose calculation in almost 100% of our experiments.

Our approach is comparable and even superior to similar approaches for 2D/3D registration problems like [77, 230, 252] in terms of the capture range. Optimization with a hardware accelerated L-BFGS-B algorithm is sufficient to enable a feasible rough patient localization on an operating table and a subsequent automatic pose calculation in reasonable time. We are not aware of any full volume traversal approach faster than ours, which we think is due to two reasons. First, only DRR generation had been implemented on GPUs and further computations (gradient approximation) have been performed classically on the CPU. Second, the ever increasing performance of graphics hardware puts us in the position to utilize computational resources which far surpass those available only a few years ago.

The method of direct specialization of the single-volume ray-casting kernel is well suited for well defined problems as presented in the motivating 2D/3D registration task of this Chapter. However, this method is not directly applicable to visualization applications or to applications which require more than one input volume. A multi-volume ray-casting kernel can be derived with automatic differentiation to support high dimensional registration tasks but the basic algorithm itself is too slow to perform efficiently enough. The kernel which is presented in Chapter 3 can *theoretically* also be automatically differentiated for multi-variate (registration) tasks, but this approach would require operator overloading which is still not fully possible because of the missing full C++ feature-set of even the most recent versions of GPU programming languages like Nvidia CUDA.

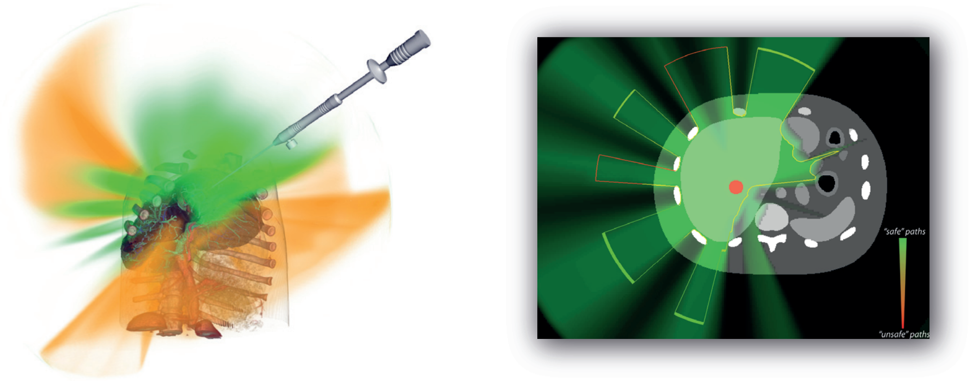
PART III - APPLICATIONS

Chapter 6

Tumor accessibility planning using multi-volume rendering

In modern clinical practice there are many ways to collect the data that might help in planning an intervention. However only a few visualization techniques can take advantage of all imaging modalities that are available for a specific task. For tumor resection and ablation (e.g. minimal invasive radio frequency ablation (RFA)) we propose a novel visualization approach based on a natural phenomenon of light ray attenuation in a scattering medium (for an example see Figure 6.2). We consider the tumor to be a bright light source emitting rays from every surface point in every direction. These rays are either attenuated or extinguished by important anatomical structures and thus shine out of the volume if nothing important is in their way. The rays can be visualized volumetrically together with other volumetric scans and/or geometric objects with our polyhedral DVR system. As a further simultaneous processing step, the rays can also be clustered into bundles and evaluated on the tumor surface. The *safest* way to the tumor is then defined by the area and direction of the largest ray bundle.

In modern clinical practice, the treating physician makes a decision about the trajectories of medical tools or the areas of resection for a particular patient. This decision is usually based solely on the physician's experience with similar interventions and general knowledge of the vulnerable anatomical structures within the human body. This empirical approach leads to a strong dependence of the treatment result on the experience of the clinician. Hildebrand *et al.* [98] have shown that operator experience has a *significant* influence on the treatment outcome of minimally invasive radio frequency ablations of malignant liver tumors. Mueller *et al.* [165] and McDonald *et al.* [154] showed similar effects for coronary interventions and spine surgery. Hence, in certain cases, this unsupervised access-path planning approach may be harmful or even *deadly* for the patient. Even when a navigation system is used for guidance during an intervention (e.g., in neurosurgery), the access path is still chosen empirically by the performing physician in advance, and thus the treatment outcome depends on the physician's experience.



(a) (b)

Figure 6.1: Regarding a tumor as a volumetric light source, we estimate the safety of *all* straight access paths in the region of interest based on the light intensity reaching every point. We display the safety information in 3D using direct volume rendering (a) and in 2D slice views that allow millimeter-accurate intervention planning, which are widespread in medicine (b). The 3D view (a) shows the safe (green) and the medium-safe paths (yellow). The 2D slice view (b), shows only the safe paths from the volumetric representation and provides additional information about the extent of safe areas using color-coded thick lines. Thin red lines signify that the available leeway is very small and that the physician must be very precise to use such access areas. In contrast, thick green lines mean that less precision is required, as the corresponding safe area is large.

Trajectory planning is a very complex procedure because of the huge amount of available data that must be taken into account. These data are produced using modern imaging modalities such as computed tomography (CT), magnetic resonance imaging (MRI) and its derivatives such as diffusion-tensor imaging (DTI), and many others. Furthermore new imaging modalities (e.g., dual-energy CT) are still being developed, and thus the information load on the physician planning the operation will continue to increase. Therefore, it is crucial to create systems that can integrate these data sources and assist the physician in choosing the access path. A fully automatic determination of the *best* access path is certainly desirable, but it is currently not possible in every case because of the enormous number of degrees of freedom; it is also not well accepted by physicians and even less accepted by patients. For this reason, we propose using visualization to assist the physician's decision-making. This visualization must be able to display all of the available medical data simultaneously to let the physician concentrate on the planning itself. Furthermore, it is beneficial to visualize information about the accessibility of the target structure along with the medical data. However, this visualization should only provide auxiliary information to the physician, leaving the final decision to the human operator.

In this work, we propose a novel multi-stage tumor-accessibility visualization approach that makes it possible to evaluate the safety of *all* possible straight access paths and to display this information in an intuitive way, thus allowing the performing physician to make a more informed decision without limiting treatment options. The intuitiveness of our visualization originates from a basic metaphor: we think of a tumor as an omnidirectional volumetric light source that is placed in an isotropic scattering medium. The vulnerable structures act as completely opaque or semi-transparent obstacles that block the light 'emitted' by the target structure. This model produces ray bundles in the safe regions and 'shadows' in the unsafe regions. These ray bundles are similar to the widespread and thus easily comprehensible natural phenomenon usually referred to as '*god rays*' or '*crepuscular rays*' (e.g., see Figure 6.2). We have refined this basic metaphor to provide valuable information to a physician and have implemented our method as a complete accessibility visualization system. We present the following technical contributions in this work:

- We propose an algorithm for the evaluation of *all* access paths to a tumor with respect to their safety. It takes into account the potential risks extracted from all of the available volumetric datasets (Section 6.2.2). We show that our method is flexible and can be adapted for various uses without changing the core algorithm. However, in this Section, we mostly refer to the medical use because of the available datasets and the given evaluation possibilities.

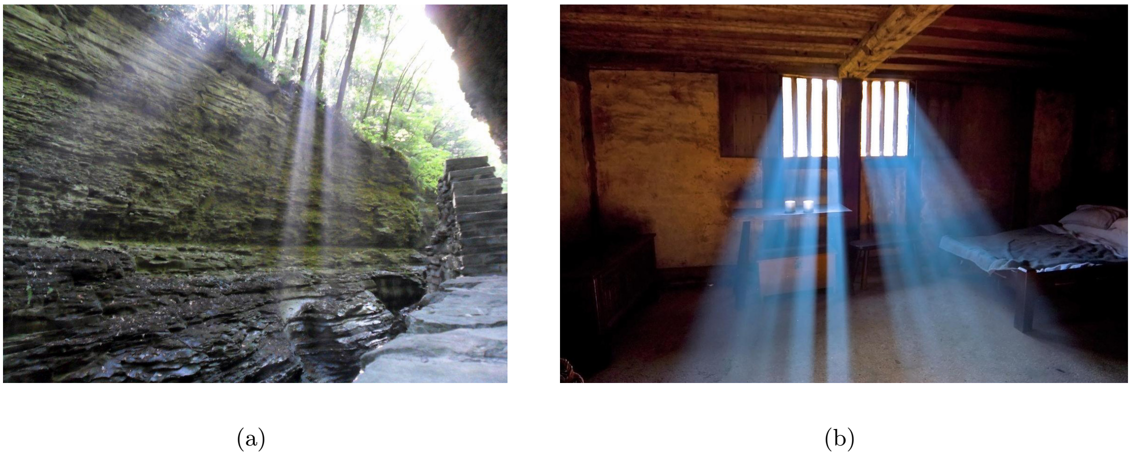
- We propose an algorithm for computing the amount of available leeway in the *safe* areas and show how this information can be saved in a form suitable for visualization (Section 6.2.3).

- We show how the information regarding the safety of access paths and the amount of leeway can be presented to a physician to assist in decision-making. Our interactive visualization system combines 3D representation for a good overview of all access paths with widely used 2D slice views for millimeter-accurate planning (Section 6.2.4).

We evaluate the applicability of our method for clinicians using real and artificial datasets for various medical tasks. We also compare the paths extracted by our algorithms for 19 real interventions with actual paths chosen by a highly skilled physician in an unsupervised environment. The evaluation results show strong evidence that our method not only reflects the possible access path choices of an experienced doctor but also has a good chance for a high acceptance rate in clinical practice.

6.1 Medical accessibility planning

Whereas navigation systems for intra-operative assistance are quite common in neurosurgery [54], preoperative planning, besides the proper integration of multiple modalities into navigation systems, remains an open problem. For example, Brunenberg *et al.* [30]



(a)

(b)

Figure 6.2: The basic idea: crepuscular rays formed by the shape of the trees or by the frame of a window. The sunlight is scattered in the dusty air and thus an observer gets an impression of separate ray bundles. (Images taken at Watkin's Glenn, NY, by the authors (a) and inside Bayleaf Farmstead by 'There and back again', Antony Scott)

and Essert *et al.* [61] calculate safe paths for deep-brain stimulation, and Navkar *et al.* [173] and Shamir *et al.* [207] calculate them for general minimally invasive brain surgery. In contrast to our approach, all of these algorithms consider only a single point as a possible target or entrance position. For simpler surgery tasks, which sometimes require only one image modality, several active path safety evaluation systems exist: Villard *et al.* [234] optimize the needle position for ablating malignant liver tumors by minimizing the size of an ellipsoid, which models the necrosis zone, needed to cover a tumor plus a safety margin. The optimization process is constrained by a collision detection algorithm to avoid the vital organs and uses an 'insertion window' that is drawn by a radiologist on the scanned patient's skin. Altrogge *et al.* [6] propose a similar approach, but they predict the necrosis zone more accurately using finite element simulation. VanCamberg *et al.* [232] use FEM-based simulation of tissue and needle deformations to compute the best needle path for performing breast biopsies. The chosen path is constrained to be at a safe distance from blood vessels. Schumann *et al.* [205] generate a list of access paths using a set of 2D constraint maps. The paths are ranked for suitability using multiple criteria and empirically determined weighting factors. These paths are then displayed one-by-one in slices of the original CT volume. Whereas these approaches try to make a decision for a physician, using computation-based prediction, our work concentrates on using visualization to assist the planning of interventions, where current automatic approaches are not suitable.

Intraoperative guidance. Another approach to assisting physicians is the provision of additional information during the intervention. Viard *et al.* [233] show how tracking of an inserted needle can be achieved in an MRI-based setup. Colchester *et al.* [46] use a localizer superimposed on a video stream for the same task. Hansen *et al.* [91] propose an approach that augments real scenes in an operating room with a distance-controlled illustrative visualization of risk structures using a projector. Chentanez [43] proposes a method for simulating deformable tissue and needles that can be used to guide a rigid or steerable needle to reach the target and avoid vulnerable structures. These strategies help in performing the intervention through better visualization and control of the current situation, but they do not guide the surgeon towards the most feasible access paths and cannot show alternative corridors.

Crepuscular rays. Our approach, using crepuscular rays to visualize abstract information for accessibility planning uses existing computer graphics research in a completely new way. Computer graphic solutions for rendering crepuscular rays were introduced in research by Max [151], who first computed these rays for photorealistic rendering, and by Nishita *et al.* [177], who invented the airlight integral to compute the effects of light scattering by air particles. Instead of computing volumetric light rays, volumetric shadows can also be computed, as demonstrated by Baran *et al.* [12]. The corresponding computer graphics research question originates in the computation of exact from-region visibility as researched by Nirenstein *et al.* [176]. Due to the high complexity of this problem, it is most often approached by sampling the region, as performed, for example, by Wonka *et al.* [250].

Isovists. In the field of architecture, the formalism of isovists describes a similar scenario. It was first introduced by Tandy [219] and is explained in detail by Davis and Benedikt [50]. A single isovist is the volume of space that is visible from a given point in space. This mathematical formalism nicely describes situations similar to our visualization metaphor.

Anatomy rendering. 3D rendering of human anatomy is usually performed with direct-volume rendering (DVR), whereas vulnerable structures, such as vessels, are often rendered as opaque geometry [71, 197] to reduce the computational load. However, rendering systems that allow correct rendering of translucent objects and multiple intersecting volumes are attracting more attention due to advances in GPU processing power and flexibility. For example, Beyer *et al.* [17] use a multi-volume rendering system to build a brain surgery planning system. In this Section, we also use such a rendering system as proposed by Kainz *et al.* [105].

Access path visualization. Rieder *et al.* [197] visualize a possible access path by cutting out all tissue in a cylindrical volume between the target and entry points. Baegert *et al.* [11] visualize safe access areas through the abdominal skin of a patient as holes in a fully opaque surface. The decision for making such holes is based on optimization of a set of candidate access paths with regard to multiple criteria. A similar technique is used by Villard *et al.* [234], where the 'insertion window' is visualized as cut-away skin.

Brunenberg *et al.* [30] compute safe paths for a neurosurgery scenario and visualize them as opaque geometry on the brain surface in a 3D rendering that displays vulnerable structures only. A selected path can be further inspected using cutting planes orthogonal to the probe axis. Vaillant *et al.* [227] evaluate a cost function along straight paths between the outer brain boundary and a target point. The computed values are then mapped to a color scale of the rendered brain surface. Although all of these access-path visualization techniques are easy to interpret, it should be noted that they make it difficult to observe the full needle trajectory or the vulnerable structures within the body, or they lack context information.

We compare the most advanced access path planning systems of Schumann *et al.* [205] and Baegert *et al.* [11] with our approach in Table 6.1. Note that the main difference is that our system aims to provide a visualization to assist in planning and leaves the freedom of decision to the performing physician instead of trying to find an optimal path in a fully automatic way.

	Baegert	**Schumann**	**Ours**
direct output	one path	one path	all paths
path visualization	3D^{a}	2D^{b}	3D+2D^{c}
target structure	point^{d}	point^{d}	full tumor
computation space	2D	2D	3D
computation time	30s^{e}	5s^{f}	30s – 300s^{g}
target application	RFA	RFA	generic
computation strategy	two-pass	multi-pass	single-pass
visualization strategy	cut outs	one path	volumetric

^{a} areas and one path ^{b} one path ^{c} all paths ^{d} volumetric constraints ^{e} depends on polygon resolution ^{f} depends on numerical constraints ^{g} depends on tumor size

Table 6.1: An overview of our accessibility evaluation system compared to the most similar systems proposed by Baegert *et al.* [11] and Schumann *et al.* [205].

6.2 Our approach

To illustrate our idea, we propose the following thought experiment. Imagine that a person needs to find and pull out a small object from a dark and dusty room with many obstacles in the way. A blind search for the best location from which to reach the object will be difficult and error-prone. However, if the object is a lamp that is switched on, then the beams of light that form in the dusty air will guide the searching person to the best path to retrieve the lamp. The best path will be easily identifiable: it is where the light beams both are strong (which signifies few obstacles in the way of the light) and cover a large area (which implies that there will be more space to operate in).

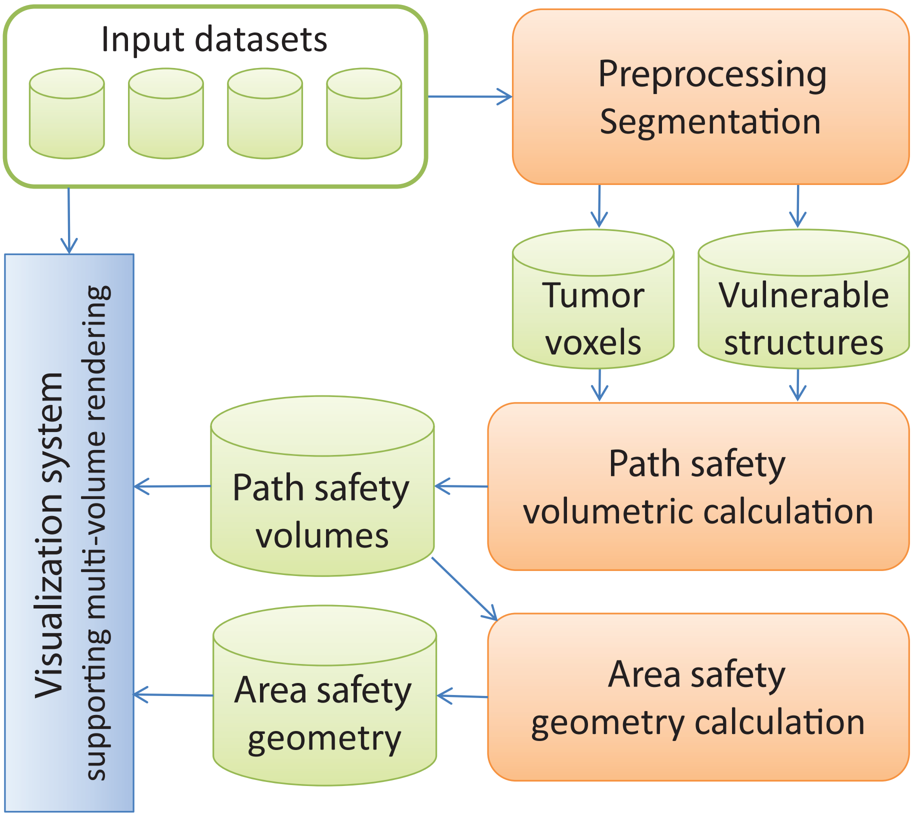


Figure 6.3: The overview of the required steps for our visualization method.

In the following subsections, we describe how we employ this metaphor for the tumor-accessibility visualization task. The description will follow the basic data flow in our application, which is outlined in Figure 6.3. We will begin by describing how to define the target and blocking structures (the ‘preprocessing’ stage, Section 6.2.1). Then, we describe how the basic metaphor can be adapted to convey useful values to a physician and provide an algorithm for computing this information (the ‘path safety’ stage, Section 6.2.2). During the development of our method, we continuously consulted with clinicians to ensure that we would fulfill their needs. They drew our attention to the insufficiency of a simple projection of 3D data for 2D slice visualization. In section 6.2.3, we propose an algorithm for evaluating the extent of *safe* areas that is designed to solve this problem (the ‘area safety’ stage). Last, in section 6.2.4, we describe the visualization system that allows a physician to explore the data interactively.

6.2.1 Preprocessing

Preprocessing of the input datasets is necessary to classify the intensity values of the scanned volumetric datasets by their vulnerability. This type of classification is usually referred to as *segmentation*. This step depends heavily on the application and is described in detail for two examples in Section 6.3.1.

Often, a rough segmentation by thresholding the intestines to find impassable structures is sufficient. This method is also used as a standard method by Schumann *et al.* [205]. Thresholding can detect bone structures, which have very high intensity values in suitable imaging modalities (e.g., CT) or lungs, which usually have very low intensity values. However, this simple method is not applicable to distinguish more complex structures such as, for example, bronchial tubes, vessel trees, or neurological structures in the brain. Segmentation of these structures is an active area of research, and increasingly accurate algorithms are becoming available. Therefore, we have decided to leave the preprocessing as a completely independent and replaceable part of our method. In the subsequent steps, we can deal with all types of binary and continuous volumetric segmentation results.

Another application-dependent variable is the definition of the vulnerability of various structures. This definition must be made once by medical experts for each intervention type. Although a continuous scale for *vulnerability* would be easy to implement, our medical partners have suggested using a few discrete levels of vulnerability. Therefore, we use four to six levels for the examples in this Section, where low values denote non-vulnerable structures and high values denote impassable structures.

After the segmentation and classification of the relevant structures is completed, we additionally apply a Gaussian smoothing filter with a kernel size of 5 mm to add a smooth safety margin around all the blocking structures. The safety margin is necessary because of the inherent uncertainty that is caused by the change of relative organ position between the planning and actual intervention stages and the inherent uncertainty of segmentation methods. This change occurs due to respiration for abdominal interventions and due to brain shift, caused by the opening of the skull, for brain surgery.

6.2.2 Path safety

We could use the light-scattering metaphor directly, but we want to convey to the user more valuable information than simply the amount of light reaching a certain point in space. Therefore, we generalize the light-scattering equation and search for a mathematical expression that maintains the basic 'beams of light' idea but also conveys information about dangerous structures on the way to the tumor. For convenience, we will use the light metaphor to describe the computed values in the rest of this section.

The amount of scattered light reaching a viewpoint from a direction $\mathbf{r}$ is obtained using the low-albedo volume rendering integral [155]:

$$I(\mathbf{r}) = \int_0^L I_s(s) \cdot e^{-\int_0^s \tau(t) \mathrm{d}t} \mathrm{d}s, \tag{6.1}$$

where L is the length of ray $\mathbf{r}$, I_s is the intensity of light scattered by particles in the air in the direction of the viewpoint, and τ is the extinction coefficient.

The definition of I_s depends on the value that is to be visualized and, in general, can be arbitrarily complex. However, for this Section, we define I_s using two functions. The first function, which we call the *ray accumulation* function, governs the computation of the amount of light I_p from a point source that reaches a certain point in space. To compute the light intensity reaching that point from the entire target structure, we represent the target structure as a union of an infinite number of point sources. The way that the intensities from different point sources are combined into the final intensity I_s is described by the *ray combination* function. The definitions of ray accumulation and ray combination functions that we found useful in the case of medical interventions are described further (see Figure 6.4 for comparison) in the following.

The **surface visibility value** shows the percentage of the target structure surface facing a certain point that can be reached from that point without any significant blocking structures on the way:

$$I_p^{perc}(\mathbf{p},\mathbf{x_0}) = \mathbf{I}_{[0,\varepsilon)}\left(\int_C b(\mathbf{x})\,\mathrm{d}s\right) \tag{6.2}$$

$$I_s^{perc}(\mathbf{x_1}) = \frac{\iint_{\partial\mathfrak{T}_+} I_p^{perc}(\mathbf{x},\mathbf{x_1})\,\mathrm{d}S}{A(\partial\mathfrak{T}_+)}\cdot 100\%, \tag{6.3}$$

where C is a line segment from the point source $\mathbf{p}$ to a point in space $\mathbf{x_0}$, $b(\mathbf{x})$ is the blocking value function, $\mathfrak{T}$ is the target structure, $\mathbf{I}(x)$ is the indicator function, ε is the desired threshold for insignificant blocking values, and $A(\partial\mathfrak{T}_+)$ signifies the area of $\partial\mathfrak{T}_+$. Also, $\partial\mathfrak{T}_+$ is the part of the target structure's surface visible from the point $\mathbf{p}$, considering only self-occlusion of the target structure (see Figure 6.5):

$$\partial\mathfrak{T}_+ = \left\{\mathbf{x}\in\partial\mathfrak{T} : \textstyle\int_C \mathbf{I}_{\mathfrak{T}}(\mathbf{p})\mathrm{d}s = 0\right\}. \tag{6.4}$$

We refer to it as the *front-facing* part of a surface. We also define the *back-facing* part of a surface, which we use further, as:

$$\partial\mathfrak{T}_- = \left\{\mathbf{x}'\in\partial\mathfrak{T} : \textstyle\int_{C'} \mathbf{I}_{\mathfrak{T}}(\mathbf{p})\mathrm{d}s = 0\right\}, \tag{6.5}$$

where C' is the ray starting at point $\mathbf{x}$ in the direction from $\mathbf{p}$ to $\mathbf{x}$.

Integral surface blocking value: The surface visibility value, which is very easy to understand, is not able to distinguish cases where several blocking structures lie on the same ray to the tumor (see Figure 6.4(b, e, II)), which may lead to failure to provide information about safe paths in complex cases where the target structure is strongly occluded. To handle such cases, we propose functions that show the total weighted volume of blocking structures on the way from a point in space to the front-facing part of the target structure's surface:

$$I_p^{surf}(\mathbf{p},\mathbf{x_0}) = \int_C b(\mathbf{x})\,\mathrm{d}s \tag{6.6}$$

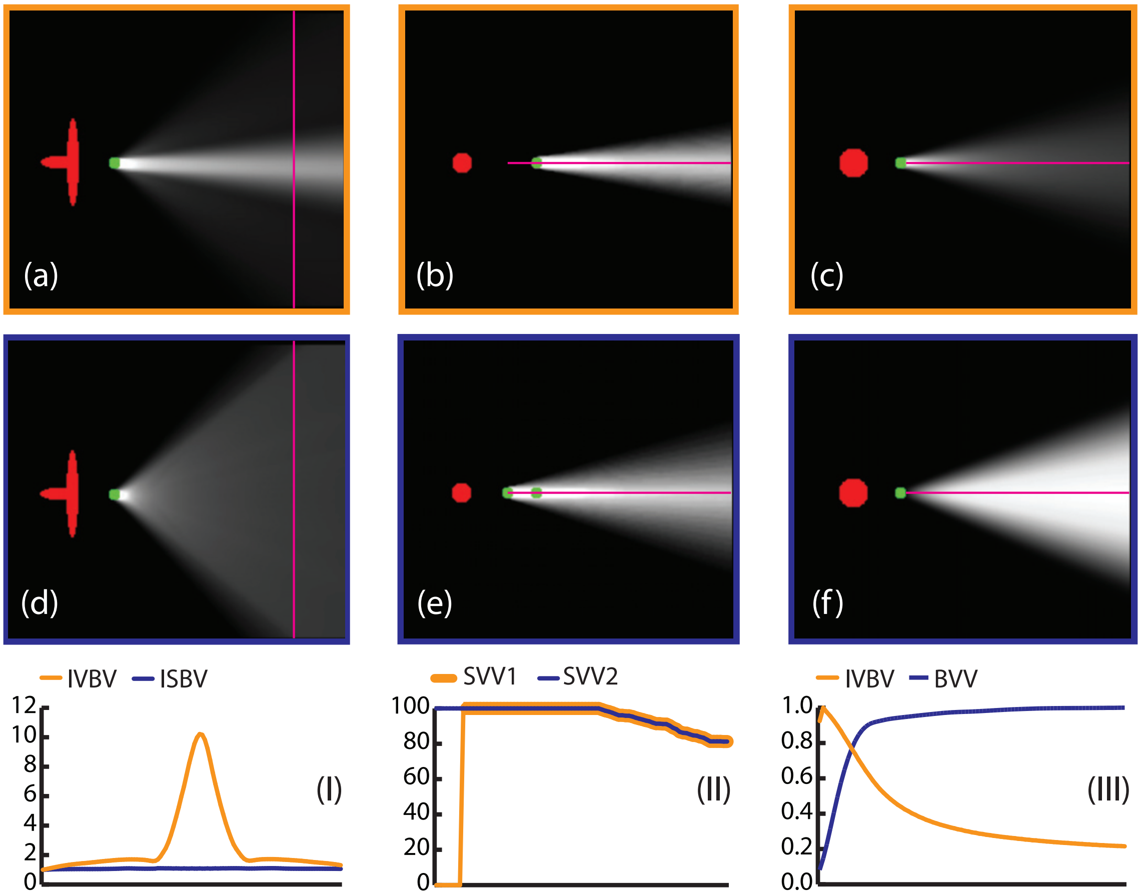


Figure 6.4: Comparison of the behavior of various value-accumulation strategies. The images show the slices of the path safety volumes computed for different configurations of the target structure (shown in red) and blocking structures (shown in green). Panels (a) and (c) show the integral volume blocking value (IVBV); (b) and (e) show the surface visibility value (inverted, for better grayscale perception) with one (SVV1) and two (SVV2) blocking structures; (d) shows the integral surface blocking value (ISBV); and (f) shows the blocker volume value (BVV). The plots (I), (II) and (III) show the comparison of profiles along the lines in corresponding columns. The ordinate axis in the plots shows the ratio of the value to the minimum value along the entire line (I), the actual computed value (II) and the ratio of the value to the maximum value along the entire line (III).

$$I_s^{surf}(\mathbf{x_1}) = \iint\limits_{\partial\mathfrak{T}_+} I_p^{surf}(\mathbf{x},\mathbf{x_1})\,\mathrm{d}S, \tag{6.7}$$

Integral volume blocking value. In some cases, such as tumor surgery, the blocked *volume* of the target structure is of interest. The functions to compute this information

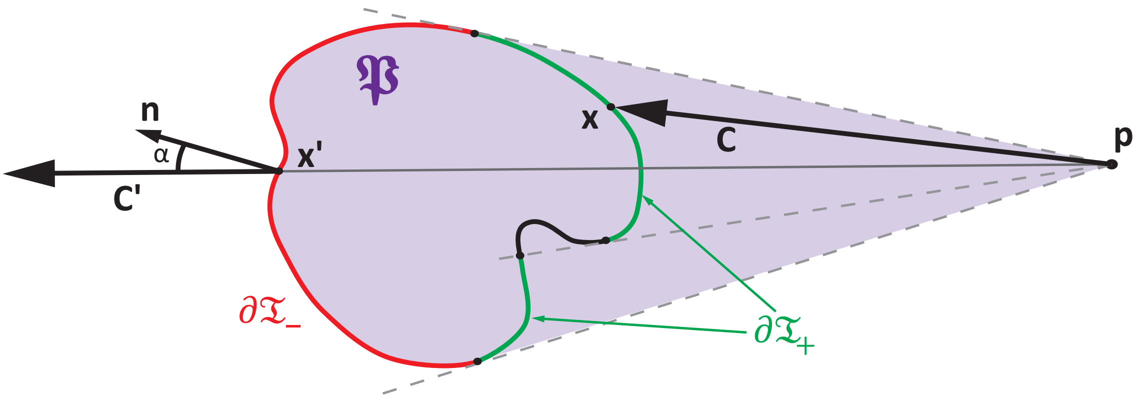


Figure 6.5: Illustration of various structures used in our method: $\partial\mathfrak{T}_+$ is the front-facing and $\partial\mathfrak{T}_-$ is the back-facing part of the surface; $\mathbf{n}$ is a surface normal; and $\mathfrak{P}$ is the pyramid, with apex at point $\mathbf{p}$ and base $\partial\mathfrak{T}_-$

.

are defined similarly to the integral surface blocking value:

$$I_s^{vol}(\mathbf{x_1}) = \iiint\limits_{\mathfrak{T}} I_p^{vol}(\mathbf{x},\mathbf{x_1})\,\mathrm{d}V, \tag{6.8}$$

where function I_p^{vol} is the same as in Eq. (6.6). Because we consider straight paths only, the integral volume blocking value can be also defined as:

$$I_p^{vol'}(\mathbf{p},\mathbf{x_0}) = \int_C b(\mathbf{x})\,\mathrm{d}s \cdot \int_C \mathbf{I}_{\mathfrak{T}}(\mathbf{x})\,\mathrm{d}s \tag{6.9}$$

$$I_s^{vol'}(\mathbf{x_1}) = \iint\limits_{\partial\mathfrak{T}_-} I_p^{vol'}(\mathbf{x},\mathbf{x_1})\,\mathrm{d}S. \tag{6.10}$$

Using this definition, the subsequent evaluation of these integrals is less computationally intensive.

The **blocker volume value** shows the total weighted volume of blocking structures on the way from a point in space to the entire target structure. It can be computed as a volume integral over the pyramid $\mathfrak{P}$, with the apex as the point and the base as the back-facing part of the target structure's surface (see Figure 6.5). After conversion from an integral in a spherical coordinate system, the blocker volume value can be defined as:

$$I_p^{block}(\mathbf{p},\mathbf{x_0}) = \int_C b(\mathbf{x}) \cdot d^2(\mathbf{x}) \cdot \cos(\alpha)\,\mathrm{d}s \tag{6.11}$$

$$I_s^{block}(\mathbf{x_1}) = \iint\limits_{\partial\mathfrak{T}_-} I_p^{block}(\mathbf{x},\mathbf{x_1})\,\mathrm{d}S, \tag{6.12}$$

where α is the angle between the normal to the surface at point $\mathbf{x_0}$ and the ray.

Discretization. We assume that the scattering is isotropic, and therefore the amount of light scattered at each point in space is viewpoint-independent. Consequently, we can precompute the values of I_s in the region of interest in an offline step and store them as a volumetric dataset (in the rest of the Section, we refer to this volume as the *path safety volume*). This volume is then used by a multi-volume rendering system [105] to evaluate equation (6.1). We choose the region of interest to be a bounding sphere of all the vulnerable structures, centered at the centroid of the target structure. We use a regular grid to cover the target domain and set its spacing to the minimum of the spacings of the input datasets.

Then, for every voxel of the output volume, we cast rays through the blocking volumes to every voxel of the target area, which changes according to the function being computed. For each ray, we accumulate the value using a discretized ray accumulation function. Figure 6.6 illustrates this process. The sampling distance is chosen to be half of the smallest input volume spacing, which will automatically satisfy the Nyquist criterion for all the input volumes. The values accumulated for different rays are then combined with a discretized version of the ray accumulation function to produce the final value of I_s that is stored in the path safety volume.

The discretization of all of the ray accumulation and ray combination functions is done in a similar way. For example, for the case of the blocker volume value, the discretized version of the ray accumulation function (Equation (6.11)) is:

$$I_p^{block} = \sum_{i=1}^{N_s} b_i \cdot V_{sample_i} \tag{6.13}$$

$$V_{sample_i} = s^2|\cos(\alpha)|\frac{i^2}{N_s^2} l_{step}, \tag{6.14}$$

where N_s is the number of samples along the ray, b_i is the i^{th} sampled blocking value, s is the spacing of the target structure volume, l_{step} is the length of a single step along the ray, and α is the angle between the normal to the surface and the ray direction. V_{sample_i} is the volume of the i^{th} sample (see Figure 6.6). Note that no blocking structures will be missed when casting the rays, because the Nyquist criterion is satisfied, and the maximum distance between cast rays does not exceed the spacing of the blocking volumes.

The discretized version of the ray combination function (Equation (6.12)) is defined as:

$$I_s^{surf} = \sum_{j=1}^{N_v} I_{p_j}, \tag{6.15}$$

where N_v is the total number of rays cast from the voxel, and I_{p_j} is the corresponding ray value.

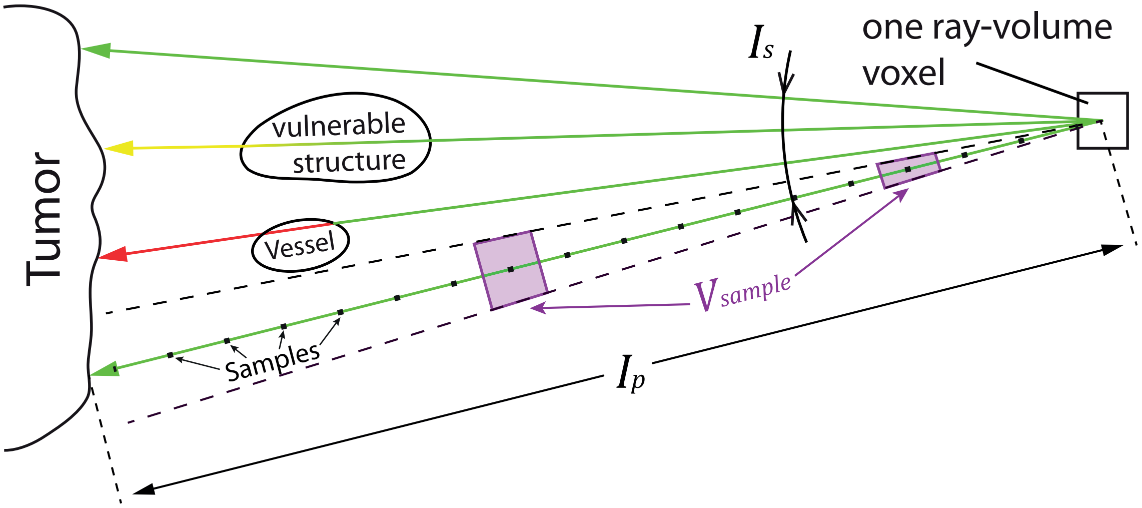


Figure 6.6: Illustration of the calculation of the integral voxel value for one voxel. Impassable or very dangerous structures raise the accumulated 'ray value' I_p quickly to high values. Other vulnerable structures gradually increase the accumulated value based on their size. Separate ray values are then combined to a voxel value I_s. V_{sample} is the approximated volume of the sample, which increases with distance from the voxel. It is used to avoid weakening of the effect of dangerous structures with distance for the blocker volume value.

6.2.3 Area safety

Many physicians often prefer using multi-planar reconstruction views instead of, or in combination with, a 3D view. To convey the safety information in 2D views, we display the relevant slices of the path safety volume. However, because the 2D slices lack depth information, it is difficult to determine the overall extent of a safe area (unlike 3D, where safe areas can be identified at a glance). To address this problem, we propose the following algorithm:

1. Approximate the target structure surface using a geometry.
2. Extrude each vertex of the geometry in the direction of its normal until a certain distance or an 'unsafe' point is reached. [*]

[*] The criterion for considering a certain point in space to be 'unsafe' can be varied based on the application. Usually, a point is considered unsafe if the path safety at the point exceeds a certain threshold. However, we can also use the blocking value datasets to determine if the point is unsafe. In this way, the described algorithm can be used as a fast preview of the path safety. However, it cannot act as a replacement for the path safety computation method, as it takes only a few paths into account and cannot distinguish between medium-safe and dangerous paths.

3. Classify vertices as ‘safe’ and ‘unsafe’, and determine the connected regions containing only ‘safe’ vertices.

4. Weight each vertex based on the size of the connected safe cluster to which it belongs.

5. Display a corresponding slice of the computed geometry to provide additional information about the overall extent of the safe regions.

We approximate the segmented tumor with an ellipsoid to avoid non-manifold faces and overlaps during the extrusion process. We align the ellipsoid’s axes with three main axes of the tumor, which are extracted using principal component analysis of the tumor voxel coordinates. We tessellate the ellipsoid using a regular grid in a spherical coordinate system, i.e., the angular distances between the vertices of the ellipsoid are equal in both angular directions. The tessellation level is chosen in such a way that after extrusion the maximum edge length does not exceed the spacing of the output volume of the path safety calculation.

Every vertex of the ellipsoid is subsequently extruded in the direction of its normal vector until a certain distance or an unsafe point is reached. We then separate the vertices of the resulting geometry into two classes: the ‘safe’ class, which contains the vertices that have been extruded up to the fixed distance, and the ‘unsafe’ class, which contains the vertices whose extrusion was stopped because the safety criterion was violated. Because of this binary separation, we can directly define disjunct connected safe regions. For each of these regions, we compute the mass and save it as a scalar property with every vertex belonging to the region. The algorithm for the mass calculation uses the discrete form of the divergence theorem with the general assumption that the surface is watertight, as described by Alyassin *et al.* [7]. In the rest of the Section, we refer to the resulting geometry with weighted vertices as the *area safety geometry*.

6.2.4 Visualization system

Our visualization system contains two significant parts that are designated for separate stages of the intervention planning procedure. The first part is intended to provide a comprehensive overview of the safety of available paths and uses a 3D representation. The second part is designed for millimeter-accurate planning of the intervention when the access area is usually already chosen and uses a 2D slice representation. We allow the physician to switch between these modes interactively or use them together in a single visualization, combining the advantages inherent in both perspectives.

3D overview stage. We use a multi-volume rendering system [105] to visualize the output of the path safety computation algorithm along with the original 3D body scans. Our rendering system also allows the addition of the segmentation results as translucent geometry if desired.

A simple one-dimensional transfer function is sufficient to control the visualization of the path safety volume. Depending on the value encoded into the path safety volume, the safe paths have either low values (in the case of integral volume or surface blocking value and blocker volume value) or high values (in the case of surface visibility value). We assign the green color to the very safe paths, yellow to the medium-safe paths, and red to extremely dangerous paths (the blue-to-yellow color scheme can be used for color-blind users, and our system also allows the definition of arbitrary color mappings). The user can remove certain safety classes from the visualization by setting their opacity value to zero to reduce the visual clutter. As the values in the path safety volume represent the intensity of 'emitted light' reaching a point in space, we do not perform additional lighting or shadowing during the ray casting of this volume. An example of the visualization of only the very safe and medium-safe paths can be seen in Figure 6.1(a).

2D millimeter-accurate planning stage. For 2D visualization, we display slices of the available datasets using cutting planes with the desired orientations. The default layout uses axial, coronal, and sagittal orientations of the cutting planes, as these orientations are widespread in medicine. However, the user can choose an arbitrary orientation during the interaction.

The slices of the original datasets, segmented datasets, and path safety volume are displayed with a contour that results from the intersection of the cutting plane and the area safety geometry. We render this contour as an OpenGL triangle strip with varying thickness and color. For the color mapping of the available leeway, we use the same color scheme as for the three-dimensional path safety rendering. The thickness of the line segments is also defined by the area safety information. After discussion with our medical partners, we fixed the upper limit for the line thickness to be one centimeter in the patient coordinate system. Figure 6.1(b) shows a sample image of an axial slice of an artificial dataset rendered using our method.

6.3 Implementation

Precomputation. The path safety computation requires the evaluation of all possible and impossible access paths. This process is very computationally intensive, and a full evaluation might take up to several years using conventional CPU-based iterative approaches. Because the evaluation of the safety level of every voxel is completely decoupled from the evaluation for other voxels, our algorithm is very well suited for parallel computation. We implemented our path safety computation algorithm using NVidia's CUDA to utilize the available processing power of the GPUs. The lookup volumes, including vulnerable structures and impassable structures, are stored in the GPU memory in the available texture units, allowing us to use the advantages of hardware trilinear interpolation and texture caching. The CUDA kernel for the fast path safety volume computation is outlined in Algorithm 4. We use 3D thread blocks with a size of 8x8x8, which results in 512 threads per block. This layout allows better thread coherency and less texture-cache misses as compared to 1D or 2D thread blocks. Each thread computes the path safety for exactly one output voxel.

The area safety computation algorithm is not as computationally intensive. Therefore, for simplicity and clarity of the implementation, we use the Visualization Toolkit (VTK[†]) for geometry extrusion and clustering.

Rendering. We use the Medical Imaging Interaction Toolkit (MITK[‡]) as the core medical visualization system. MITK provides all of the necessary basic functionality for the visualization of medical data. We have also integrated the polyhedral CUDA-based rendering system proposed by Kainz *et al.* [105] as a separate view in the provided widget system. The 2D slice visualization of the abdominal data from an arbitrary image modality, the path safety volume, and the area safety geometry use OpenGL as the core rendering system. Multi-planar reconstructions are done by 3D texture mapping. For area safety visualization, we project the geometry's normal vectors onto the desired cutting planes and use them together with information about the current cluster size to render thick lines as OpenGL quad-strips. Thus, the projected normal vectors define the direction of the quads, and the additionally stored information about the mass of an area defines their size and therefore the thickness of the lines.

Performance. Although the direct geometric area safety calculation is performed in a couple of seconds on the CPU ($O(n)$ complexity, where n is the number of vertices in the mesh approximating the tumor), the path safety volume calculation depends strongly on the number of voxels in the segmented tumor ($O(n^3 \cdot m)$ complexity, where n is number of voxels per dimension of the output volume, and m is the number of tumor voxels). We are able to split large volumes into several parts and calculate the path safety volumes on multiple GPUs simultaneously. With this approach, the evaluation of path safety takes $600ms$ per tumor voxel and approximately five minutes for the whole tumor on our test system (Intel QuadCore 3.16 GHz, 12 GB RAM, NVidia Quadro6000 and GTX580) for 512^3 input data sets and a tumor with an average diameter of 2 cm. Note that we do not need to update this volume during visualization and planning and that this computation must be performed only once per intervention.

The required time for preprocessing depends strongly on the desired application and the available segmentation algorithms. During our experiments, these steps took from a few seconds (clear structures, segmented by thresholding) up to several minutes, including user input (brain DTI, fMRI evaluation, and sophisticated vessel-segmentation algorithms).

Memory requirements. Our method requires only the resulting segmentation volumes as an input. Whereas a pure binary segmentation can be stored as a one-bit single-value

[†] www.vtk.org

[‡] www.mitk.org

Algorithm 4 The path safety volume computation kernel in pseudo-code. The sampling of the volumes is performed in the global coordinate system. Ω refers to a volume containing all information about the vulnerability of structures and $\Omega(p)$ to the volume sample at point p; $\mathfrak{T}$ defines a volume of tumor voxels and $\mathfrak{T}(p_t)$ the tumor voxel at point p_t; I_{xyz} and A_{xyz} are the virtual attenuated light intensities emitted by the tumor. Θ defines the output volume and $\Theta(p_o)$ the sample at position p_o. $\mathbf{M}_\Theta$ is the transform matrix from the local coordinate system of Θ to the global coordinate system. δ is the normalized direction of a virtual light ray scaled by the step size γ. The implementation of accumulateValue, combineValues, and finalizeValue functions as well as the values of *initialIntensity* and *initialVoxelValue* depend on the value being computed (see Section 6.2.2).

```
function castRay($p, \delta, \Omega, N_{steps}$)
    $A_{xyz} \leftarrow initialIntensity$
    for $N_{steps}$ do
        $p \leftarrow p + \delta$                                    ▷ advance along ray
        $A_{xyz} \leftarrow$ accumulateValue($A_{xyz}, \Omega(p)$)
    end for
    return $A_{xyz}$
end function
function computeAccessibility($\Omega, \mathfrak{T}$)
    setup computation grid
    for all GPU THREADS do                      ▷ is done in parallel on the GPU
        $p \leftarrow$ thread's index in the computation grid
        $p \leftarrow \mathbf{M}_\Theta \cdot (p \cdot spacing(\Theta))$                ▷ get global coordinates
        if $p \notin$ sphere inscribed in $\Theta$ then
            return
        end if
        $I_{xyz} \leftarrow initialVoxelValue$
        $\gamma \leftarrow 0.5 \cdot \min(spacing(\Omega))$                          ▷ Nyquist criterion
        for $p_t \in \mathfrak{T}$ do
            $\delta \leftarrow normalize(p_t - p) \cdot \gamma$                  ▷ calculate step direction
            $I \leftarrow$ castRay$\left(p, \delta, \Omega, \left\lceil \frac{||p_t - p||}{\gamma} \right\rceil\right)$
            $I_{xyz} \leftarrow$ combineValues($I_{xyz}, I$)
        end for
        $\Theta(p) \leftarrow$ finalizeValue($I_{xyz}$)
    end for
end function
```

scalar field that combines all segmentation results, segmentations that also include levels of vulnerability and smooth safety margins have to be stored with a bit-depth at least comparable to that of the input volumes. Nevertheless, intersecting vulnerabilities can also be summed up and subsequently saved in a single volume. Overall, the required additional memory is not more than that of the input volume with the highest resolution.

The memory requirements of the geometric representation of access areas are negligible compared to those of the path safety volume. Whereas the path safety volume might require several hundred megabytes of storage space, the area safety geometry occupies approximately 10 MB, even at a very high polyhedral resolution (D600.000 faces).

6.3.1 Selected segmentation procedures

The preprocessing of the input data varies depending on the target application area. As the main focus of this Section lies in the medical domain, we will describe the preprocessing step for the medical data used for planning the RFA of liver tumors and brain tumor resection. Note that the operation of our path safety computation system is fully decoupled from the algorithms used for the determination of vulnerable and target structures, and thus these algorithms can easily be replaced.

Preprocessing for minimally invasive liver interventions. In modern practice, both contrast-enhanced and native CT scans are used to plan an RFA intervention. We extract the following information from these scans:

- **Target structure.** The tumor is segmented using the contrast-enhanced arterial phase CT scan using the algorithm from Hame [90].
- **Safe structures, level zero.** The structures that can be safely penetrated during the RFA are the liver itself (with exceptions for the vessels, see below), which is segmented using the algorithm described by Alhonnoro *et al.* [5], and the skin, which is segmented using thresholding.
- **Vulnerable structures, level one.** In this case, the less vulnerable structures are small venous vessels of $5 - 10mm$ that are segmented from contrast-enhanced CT scans using the algorithm by Alhonnoro *et al.* [5].
- **Vulnerable structures, level two.** Arteries and large venous vessels ($> 10mm$) constitute more vulnerable structures. Organs whose penetration is possible but undesirable (such as lungs, with very low intensity values) are classified as vulnerable and segmented using thresholding. Other organs can also be defined as level three structures. This definition depends on the choice of the performing physician.
- **Impassable structures, level three.** Bones and metal implants are extracted using the native CT scan by simple threshold-based segmentation (very high intensity values). The organs that must not be penetrated during the intervention (such as heart and bowel) are also considered to be impassable and are segmented based on the liver segmentation results.

Preprocessing for brain tumor surgery. For the case of brain tumor surgery planning, various MRI and CT scans are usually taken into account. For our examples, we use perfectly registered and segmented datasets provided during the VIS-contest 2010. The datasets are therefore courtesy of Prof. B. Terwey, Klinikum Mitte, Bremen, Germany. With tools provided by MedINRIA[§], we assign six discrete levels of vulnerability for the following segmentations:

- **Target structure.** The target structure is the tumor whose resection is being planned. It has been segmented by a professional by hand.
- **Safe structures, level zero,** include the skull, skin and other non-neurological structures, such as low-tumor-lesion thickness areas (extracted from FLAIR and SWI datasets), white matter with no relevant connecting fiber bundles (fMRI, DTI), and non-stimulated gray matter areas (fMRI).
- **Vulnerable structures, level one,** are weakly stimulated gray matter areas (fMRI), areas with no fiber bundles (DTI) and no vessels (CE T1).
- **Vulnerable structures, level two,** are moderately stimulated gray matter areas (fMRI), areas with no fiber bundles (DTI) and no vessels (CE T1).
- **Vulnerable structures, level three,** are areas containing only a few fiber bundles (DTI) and no vessels (CE T1).
- **Vulnerable structures, level four,** are highly stimulated gray matter areas (fMRI) or areas containing an average count of fiber bundles (DTI) or small vessels (CE T1).
- **Impassable structures, level five,** we consider the following structures to be impassable, as damaging them would be deadly for the patient or would severely influence the brain function: highly stimulated gray matter areas (fMRI), dense fiber bundles (DTI), and thick vessels (CE T1).

Preprocessing for the engine dataset example. The preprocessing for a dataset such as this one is simple. Structures made of metal are usually not penetrable by a tool. Consequently, all areas in the dataset with a higher intensity value than that used for air can be considered impassable.

6.4 Results

We evaluated our accessibility visualization approach in several discussions with radiology experts during the implementation process. Our algorithms were successively refined using this expert knowledge. The results of this refinement process are different value-accumulation strategies for the calculation of safe and unsafe paths as they are presented in Section 6.2.2.

[§] http://www-sop.inria.fr/asclepios/software.php

Qualitative user study. To verify the suitability of the chosen path and area safety visualizations for the prospective users, we conducted an online user study among 15 physicians. Of the participants, 17% were female and 83% male, with 70% having expert knowledge in radiology, 17% in neurology, 14% in surgery, 14% in computer science and visualization, and 4% in internal medicine. Their professional experience was between five and ten years for 53% and greater than ten years for 21%. The corresponding survey is added in Appendix B.

We divided the survey into two parts. During the first part, we evaluated the overall acceptance of our method in 3D and 2D, based on an eight-value Likert scale. We chose eight values to avoid neutral answers, which can occur for odd numbered scales. Our method was presented as turns in the pitch and yaw directions for 3D visualizations and as a complete scroll through all available slices for 2D slice-based visualization. The results in Figure 6.7 show that the 2D methods were preferred by the participants in our survey. This result can be explained by the high number of participating radiologists who are specially trained to work with 2D slice visualizations. The most successful 3D visualization was the one showing only the safe paths in 3D. The addition of medium-safe paths and impassable paths decreased the acceptance by most participants. However, the wide range of results indicates that the information desired to be seen in 3D depends on personal preferences and should remain adjustable by the user, as is the case in our system. For 2D slice-based visualization, an augmentation with our proposed area safety visualization method and a combination with the projection of the path safety volume received the highest grades. The projection of the path safety volume onto 2D slices alone showed lower acceptance on average, but the acceptance also showed large variation, which provides evidence for significant differences in personal preferences. Therefore, we assume that an adjustable visualization is also the best choice for 2D views.

In the second part, we investigated the influence of our method on the choice of access paths for two cases: one liver and one brain tumor. We presented different options for access paths and observed how the decision changed with different accessibility visualizations. In the presented images, the green region shows the safest 20% of the paths, using blocker volume value as a measure. The questionnaire provided several choices for access paths, both feasible and dangerous, which were chosen in advance by a radiology expert and a neurology expert. We measured the number of feasible paths that were chosen by the participants. In the upper example in Figure 6.8, paths ‘c,d,e’ are the safest paths. For this example, the area safety calculation had the greatest impact. For example, path ‘d’ is safe but has very little leeway, which was shown with thin red lines by our algorithm. This factor motivated our survey participants to choose different paths when area safety information was shown in addition to the anatomical data. The same evidence can be seen in the lower graph of Figure 6.8, which shows an example of brain tumor surgery. Here, paths ‘a,l,k’ are the safest ones, but again, ‘k’ has very little leeway, which is visible in the area safety visualization. This factor influenced the participants’ decision to avoid the path ‘k’. Although no user chose a dangerous path using our methods, the tendency to

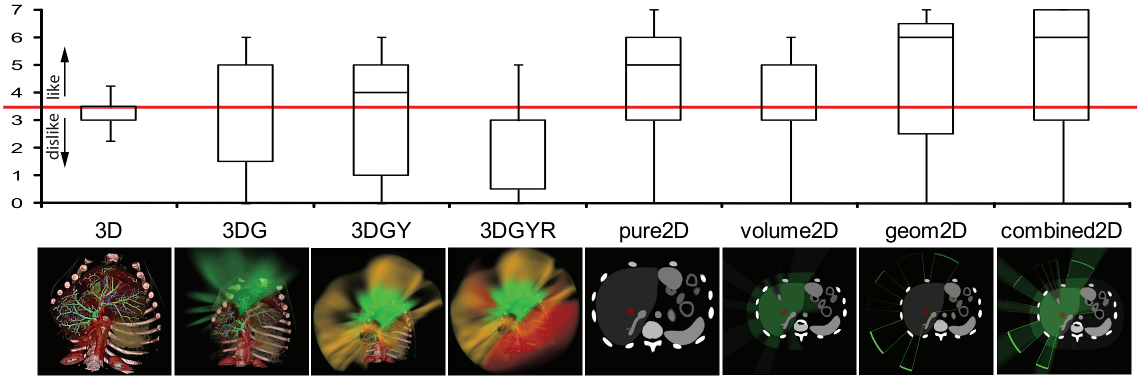


Figure 6.7: Evaluation of the acceptance of our methods for clinical practice. The participants were offered a choice in the form of an eight-value Likert scale after watching a video of our method. The 2D methods were preferred and are indicated with *pure2D*. The acceptance of 3D methods decreased with increasing amounts of displayed information. The option labeled *3DG* shows safe paths only; *3DGY* also shows medium-safe paths; and *3DGYR* further adds impassable paths. 2D multi-planar methods were also evaluated with *area safety* geometry augmentation (*lines2D*), a 2D projection of the *path safety* volume (*volume2D*), and a combination of both (*combined2D*). For this box-plot, the ends of the whisker are set at 1.5× interquartile range above the third quartile and 1.5× interquartile range below the first quartile, which corresponds to the normal convention for box-plots. The thumbnails below each box show frames from the corresponding videos.

choose a safe path increases when path safety and area safety are also displayed, as shown in Figure 6.8.

Overall, our method proved to be well-received by the participants in our survey. The results are summarized in Figure 6.7, which outlines the general expected acceptance in clinical practice, and in Figure 6.8, which shows the impact of our method on the expert decisions. Finally, we asked the participants if they would prefer a combination of the shown visualization methods. Most participants (> 80 %) would prefer a combination of a 3D volumetric representation, as shown in Figure 6.1(a), and a projection of the safe areas on 2D slices (both area safety and path safety information), as shown in Figure 6.1(b).

Gold standard comparison. In addition to the qualitative user study, which we performed to gain a subjective acceptance evaluation of the proposed accessibility-visualization method, we studied the accuracy of the displayed safety information, proposed by our algorithms, on 19 real abdominal CT-guided RFA interventions. To perform this study, we applied our visualization method to all available patient datasets and the vulnerable structures segmented from them. Subsequently, we compared the calculated path and area safety to the needle trajectory actually chosen for

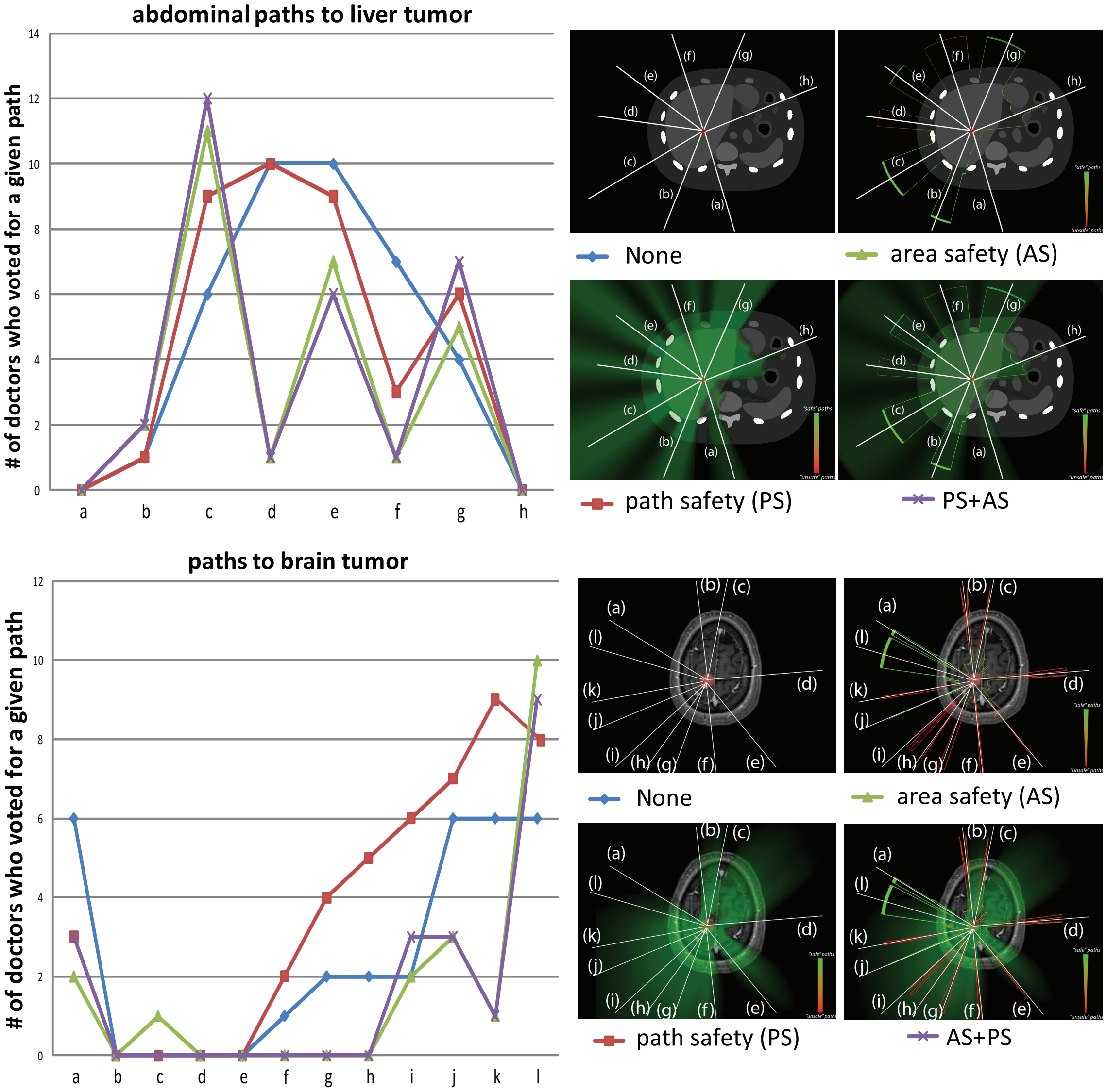


Figure 6.8: Impact of our visualization method on path choice for generic cases. **PS** refers to the path safety volume calculation and its projection onto 2D slices. **AS** refers to the geometry augmentation in 2D including the visualization of the leeway (area safety). Note that our method has a strong influence on certain path decisions.

the intervention. This straight trajectory is available in a separate registered CT scan.

We considered the chosen paths as the gold standard because all of the performing physicians had more than 10 years of experience with minimally invasive abdominal interventions. Furthermore, all patients except one have not had complications after the intervention. The only case who suffered a complication (bleeding) also recovered after a short time. Several physicians who were consulted after the intervention were not able to find an explanation for this complication, which leads us to assume that the scan may have been of insufficient resolution to show all vulnerable structures for proper planning. We consider this case as valid for our study, because all rules to find an optimal access path were followed during the actual intervention. However, we excluded two cases from the study. The first case was excluded because the patient had an implanted medical pump and several aortic stents, and the intervention had to be specially adapted. The second case was not suitable for our study because the respiratory motion in the scan showing the needle trajectory was too strong for proper registration (this patient had several partial organ resections because of multiple cancer metastases). Hence, we used 90% of the valid example interventions for this retrospective gold standard study. The results of this study are shown in Table 6.2 and further commented in the following.

For 8 cases (47%), the needle trajectory coincided with a path for which our algorithm calculated a path safety of 100%, which means that in these cases, there were no obstacles on the way to the tumor along this path. In most of these cases, the access area with the largest leeway was chosen. We also discussed these patients with our medical partners, who confirmed that the position of the tumor was suited very well for minimally invasive intervention. The remaining cases can be considered to be difficult, and they had an average gold-standard path safety of 77% using our method.

To calculate the relative area safety of the chosen paths, we rate the areas in proportion to the largest available safe area (thus, the largest area has a safety of 100%). In 7 cases (41%), the experienced physician chose the access area with the largest leeway. The average area safety of the remaining cases was 70%, which shows clear evidence that areas with the most available leeway are always chosen by the experts and that our method reflects several decision criteria for real-world intervention planning.

6.5 Discussion

We effectively applied a natural metaphor to a difficult medical-accessibility decision problem. To exploit this metaphor, we calculate an additional volumetric dataset on the GPU, that encodes the safety of all possible access paths as bright rays shining out of the body, based on various replaceable segmentation procedures. Thus, the intensity values provide additional information on how much of the target structure can be reached from every position within the region of interest. Furthermore, we evaluate the available leeway for each ray bundle and display this information in 2D slices, which are widely used in medicine. With this method, a physician can quickly and reliably determine the possible tool trajectories. A combination of area and path safety augmented 2D MPR views was implemented in a medical visualization prototype, as shown in Figure 6.9. This figure also shows that our visualization approach can be easily used to indicate dangerous and impassable areas instead of safe areas only.

case	path safety	area safety
01	100 %	75 %
02	100 %	100 %
03	100 %	100 %
04	– %	– %
05	79 %	55 %
06	100 %	100 %
07	89 %	49 %
08	100 %	100 %
09	92 %	100 %
10	95 %	82 %
11	– %	– %
12	100 %	100 %
13	100 %	100 %
14	84 %	75 %
15	91 %	85 %
16	48 %	60 %
17	43 %	32 %
18	100 %	92 %
19	70 %	94 %

Table 6.2: Proposed paths from our method retrospectively compared to 19 real-patient RFA interventions, performed by an experienced radiologist. The path safety is defined as the percentage of volume which can be reached with the chosen path, the area safety is defined as the amount of elbow space in relation to the largest available area considering the shortest path the the surface.

6.5.1 Use in medical applications

Our method shows significant improvements for the planning of various medical accessibility problems. However, we also identified possibilities for specialization and built several prototypes to overcome planning problems in special interventions. Different body parts (e.g., abdomen, brain, and prostate) require different decision criteria and introduce specific problems.

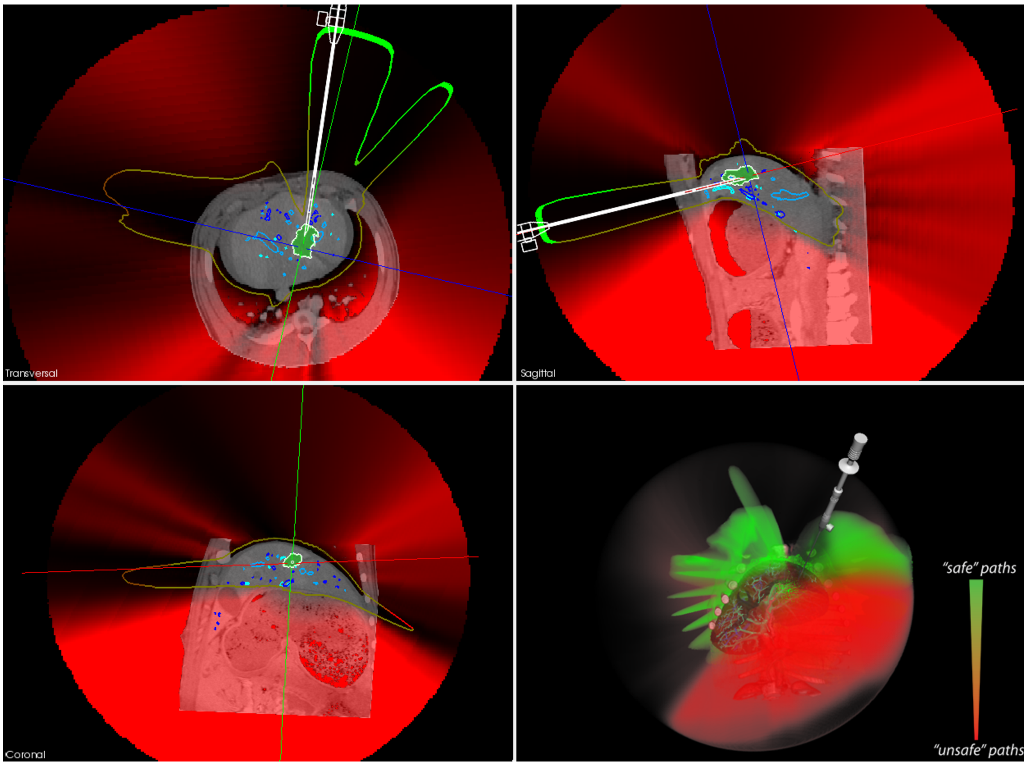


Figure 6.9: A screenshot of our medical visualization system with *area safety* and *path safety* augmentation. In this example, the path safety volume shows all dangerous and impassable paths in red. The area safety geometry shows all safe access areas. The 2D MPR views are aligned with the direction of the main axis of one safe access area. For datasets similar to this one, which show mainly large safe access areas, the area safety geometry can be smoothed and additionally displayed as translucent geometry in 3D, as shown here. The tool to be placed into the tumor (green) is a RFA needle. All vulnerable vessels are displayed in shades of blue.

Respiration compensation The most evident problem, which we discussed with experts, is abdominal and thoracic respiration compensation. During abdominal and thoracic interventions, the respiratory motion of the organs cannot be prevented, which poses a severe problem for applying offline planning results to the actual intervention. During full anesthesia of a patient, respiration can indeed be stopped for a certain time, but this time frame is not long enough to allow planning and intervention in one step. For monitored interventions with MRI guidance, our method is fast enough to be applied online or on a whole sequence of 4D motion images. However, most monitored interventions are performed with CT guidance. Constant CT monitoring is not possible due to the limited allowed radiation dose.

A feasible approach to deal with this problem is to apply our method to every stage of a patient-specific respiration simulation sequence, as is possible, for example, with the

XCat patient model [142]. During most CT-guided interventions, artificial respiration is used because of general anesthesia. Therefore, we can apply fast GPU-based registration methods between the time steps and between the patient scan and the model (e.g., Optical Flow motion fields [226]) and hence integrate the patient-specific anatomy into the model. We can thus warp a single scan of a patient to enough discrete sampling points of a whole respiration cycle (usually 10-20) and apply our method to each of the resulting volumes. The resulting path safety volumes are combined in a volume that only classifies paths as *safe* if no vulnerable structures are hit in any of the respiration states. This volume is subsequently warped back to patient space and visualized as presented in this Section. However, the overall processing time for such a sequence extends to approximately 30 minutes (excluding the necessary semi-automatic segmentation of vulnerable structures).

Our first prototype including respiration compensation shows promising results and is currently under closer investigation by our medical partners. Note that our method can always be applied directly for structures that can be fixed during intervention, such as the hip bone or skull.

Interventional Augmented Reality In addition to planning an optimal access path, finding that path during the intervention itself is also a challenging task. This stage can be defined as a third level of our method and can be configured to display either only the selected entrance segments or all paths of a selected safety level. We selected liver cancer RFA treatment for a first prototype of the system. The target clinical setup usually consists of a two-camera infrared tracking system and one or several monitors. The tracking system is small and movable to allow an optimal working area. Monitors can be provided as projections or, more commonly, as ceiling-mounted movable devices. Furthermore, additional monitors are provided outside the intervention room to control the imaging modality used. We use a small monitor mounted on a tripod to simulate the intervention room's movable monitor and an additional monitor, away from the operating field, to simulate the control monitor. The prototype of this system is shown in Figure 6.10.

6.5.2 Methodology discussion

Study results. We performed three different evaluations of our method. The first and most obvious one considered expert knowledge during the implementation process. We worked closely with medical experts and discussed all results during the various stages with them. However, in approximately 50% of the cases, our experts did not agree on a single optimal path. This disagreement was one of the main reasons we chose to visualize safe *areas* instead of direct path proposals as a single line.

Our retrospective gold-standard survey relies on the assumption of a perfect physician. As is shown by the available patient data, this assumption is of course not true. The treatment outcome depends strongly on the difficulties inherent to the tumor position and also on further circumstances such as the overall patient history. Therefore, this evaluation

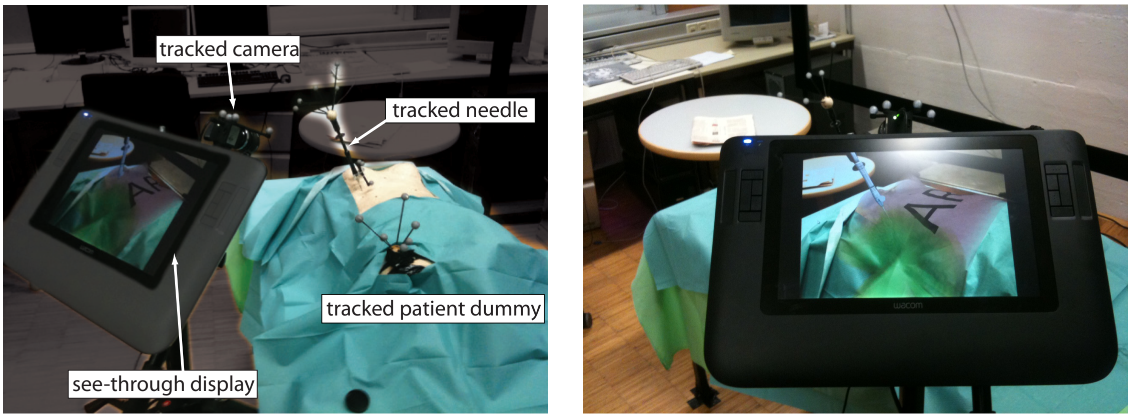


Figure 6.10: Prototype of an interventional Augmented Reality (AR) application, as a possible third level of our method. Using this system, planning decisions can be directly mapped to the real situation in the operating room.

gives an impression of the possible prospective benefit for unskilled physicians, training simulators, and complication investigation.

Other applications: Our method cannot be applied directly for architectural visibility calculations because isovist would require perspective projection, whereas medical accessibility does not. However, the projection calculation can easily be altered, and the actual visualization method might also contribute to that field of research. Our current approach can be directly applied to mechanical accessibility assistance, as shown with the engine block example data set in Figure 6.11.

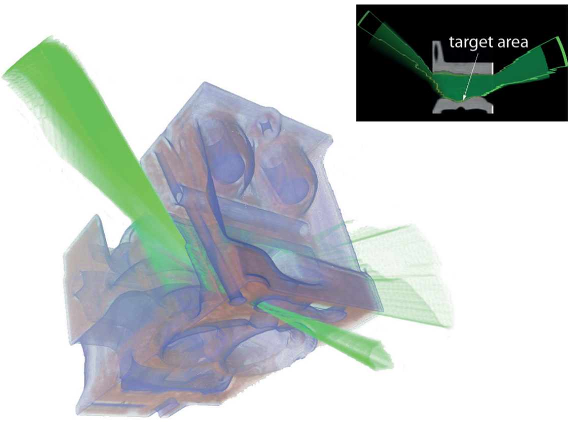


Figure 6.11: An example of the applicability of our method to other fields of research. We have chosen a spot within an engine block that is difficult to reach. Because the engine block itself is solid, only 100% *'safe'* paths are displayed. All other paths are impassable. Note the non-obvious access path at the bottom of the image.

Chapter 7

Prostate 2D/3D registration with automatic differentiation

As already introduced in Chapter 5, ray casting kernel specializations as for example done with automatic differentiation methods, can be used for specific medical registration tasks. In this section we present a practical application of this method for prostate surgery and show a method to allow a highly robust initialization to the required capture range of our AD method.

A common problem in prostate surgery is to know the exact location of the patient for tasks like rebiopsy and minimal invasive focal therapy. A possible solution is the registration of a set of X-ray images taken during an intervention to a CT data set of the same patient taken before, which is also known as 2D/3D registration.

To achieve the necessary accuracy and image generation speed, we use a specialized X-Ray target for the initialization of the registration process, which allows an initialization from a single image within one second with high accuracy.

7.1 Our approach

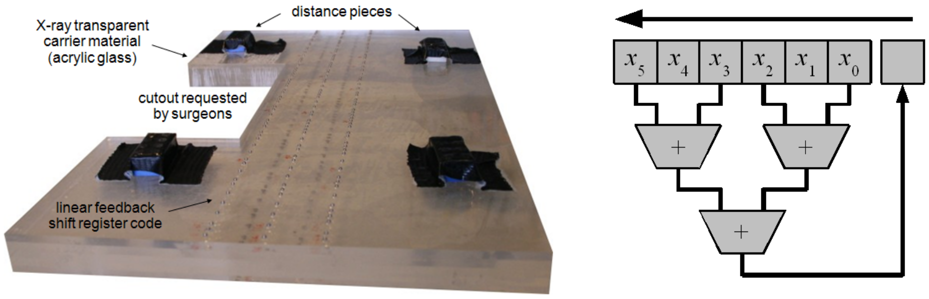
We use a plate mounted below the operating table which is prepared with a regular grid of holes on both sides. The plate is made of acrylic glass to provide maximum X-Ray transparency. The holes form a grid with 5 mm × 25 mm on one side of the plate. On the other side the same grid was used but shifted by 12.5 mm. Overall we have 78×18 available positions in our target. The challenge is to equip these holes with beads in a suitable pattern such that the reoccurring two-dimensional sub-pattern's maximum of their cross correlation was minimal in terms of coding theory. To provide unique 2D patterns of a certain size $u \times w$, the equation

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \neg(v_{(u \times w)i} \oplus v_{(u \times w)j}) = 0 \quad u > 3,\ w > 0 \tag{7.1}$$

has to be satisfied. v thereby denotes to a distinct pattern of a size $u \times w$ extracted from the complete 2D sequence. u has to be greater than three since for robust detection of the pattern we require the distance between two spheres to be either 5 mm or 10 mm (i.e., there must not be more than one "empty" position in succession). Therefore a binary "1" is represented by a single steel sphere, and a binary "0" by an empty space followed by a sphere.

One-dimensional linear feedback shift registers (LFSR) [34, 75] provide a certain code unambiguity for a given length. In our case, we are interested in the minimum size of a window such that the pattern inside the window does not appear anywhere else in the entire code. This form exactly implies the already formulated coding condition that the maximum of the in-between cross correlations is minimal. A Gold Code [73] LFSR provides this behavior for one dimension by connecting selected bits from two feedback shift registers to the last produced code part with exclusive or (XOR) gates to the input of the next sequence. Figure 7.1(b) illustrates an example of such a register. The period of the code sequence depends on the register length N and the feedback pattern, the maximum is $2^N - 1$. To produce a Gold Code from an LFSR, the XOR connected bits have to be connected according to two principal polynomials of the same order N.

With our first prototype (see Figure 7.1(a)), we have used a one-dimensional six bit LFSR code arranged in parallel lines on an acrylic glass carrier. A six bit code is restricted to 63 unique code words, and there are six different code sequences of this length witch can be produced by an LFSR.


distance pieces
X-ray transparent carrier material (acrylic glass)
cutout requested by surgeons
linear feedback shift register code
x_5
x_4
x_3
x_2
x_1
x_0
+
+
+

(a) A prototype of the X-Ray target.

(b) Linear feedback shift register.

Figure 7.1: The steel spheres were distributed on five lines related to a six bit LFSR as shown in (b) on a carrier of acrylic glass (a).

Experiments show that the prototype with five lines of six bit LFSR code works well for C-Arm pose estimation with lateral rotation. The next step is to develop a target supporting craniocaudal rotations of the C-Arm. In the craniocaudal case, the image window size in some positions is too small to provide unambiguous patterns. Therefore we developed a variant of the two-dimensional LFSR as proposed by Chen et al. [41, 42]

with an optimal binary sequence as defined by Gold [73]. Since we are not interested in minimizing the number of register stages for the 2D-LFSR as proposed by Chen et al., we can use the idea of that algorithm directly by XOR cross connection of two 1D-LFSR Gold-Codes with different principal polynomials of the same order and $N \times M$ register stages. Considering the length of the sides of our target, the next best period size is $2^6 - 1 = 63$ for the longer side. The resulting two dimensional code is shown in Figure 7.2.

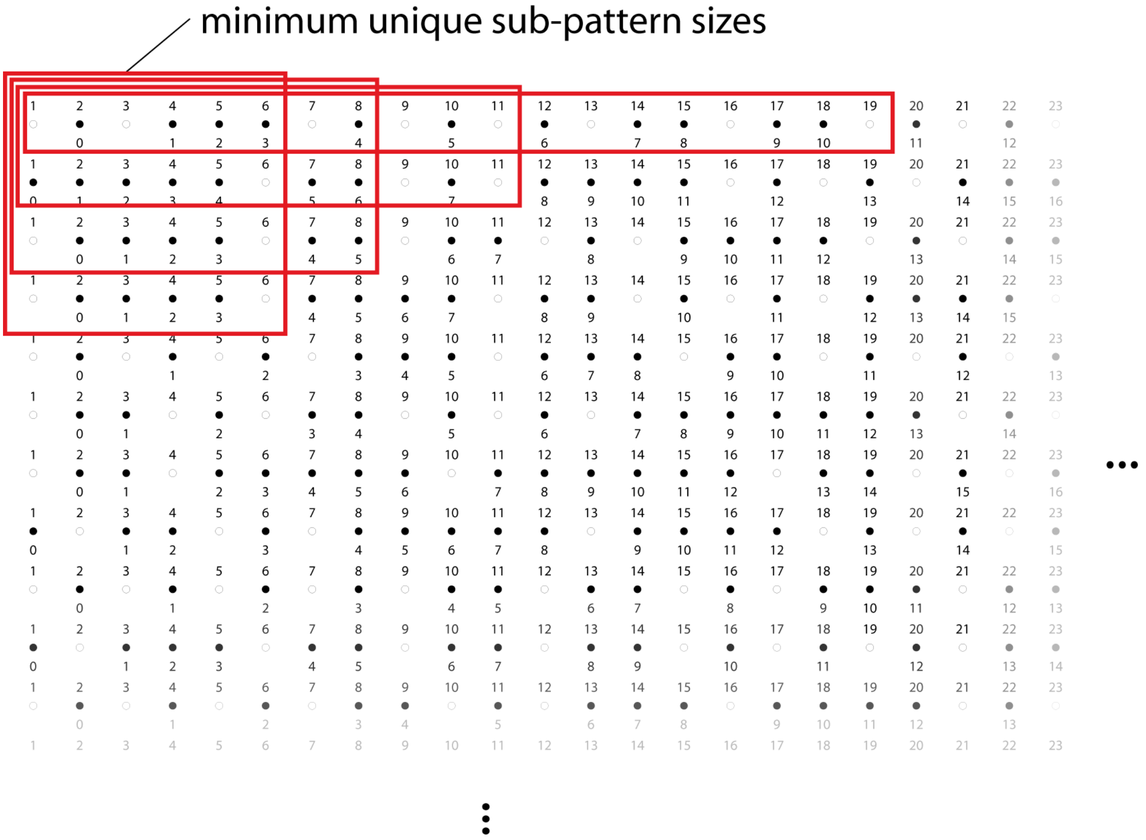


Figure 7.2: A part of our complete code pattern (front and back) generated by a 2D-LFSR with a cycle duration of 63 in one direction. A filled circle denotes a hole equipped with a steel sphere. The rectangles indicate the minimum window sizes for an unique sub-pattern. To guarantee line detection, two empty positions are not allowed after each other in one line (from left to right). Therefore a binary "0" is represented as binary "01".

The C-Arm has to be calibrated once with the acrylic glass target to determine its intrinsic parameters. Once these parameters are known, the C-Arm's orientation (rotation and translation) in 3D space can be determined.

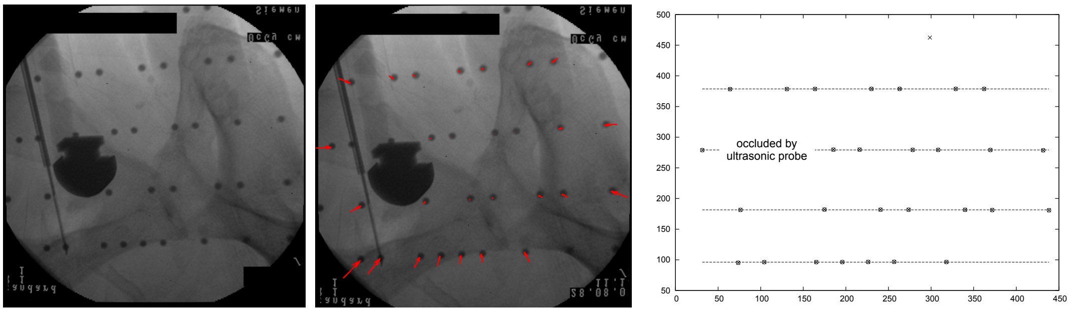
Camera calibration determins the focal length, pixel dimensions, and optical center intrinsic parameters of a camera by means of computer vision. This requires to take one picture of a large known 3D target, or several pictures of a planar calibration target (e.g., a checkerboard). Since we already have a quasi two dimensional target we have chosen the second approach. However, this step is crucial for a subsequent one-image pose estimation and it models the attributes of the system for the whole acquisition matrix.

Camera calibration is a thoroughly researched topic in computer vision and photogrammetry, and several fully automatic methods have been proposed [44]. The methods mainly differ in the underlying algebraic model of the projection process, the method to determine the model parameters, and the calibration normal used. Examples of calibration normals are plumbline targets (i.e., straight lines produced by hanging plumbline wires [25]), calibration normals with three-dimensional reference points [96], and planar normals, where all reference points are coplanar [251]. The latter can be considered as state of the art in the computer vision community. Several images of the normal are acquired by the unknown camera, each time from a different viewpoint. From this information it is possible to determine the unknown parameters of the underlying projection model. There is no definite rule as to how many calibration images are needed, since this strongly depends on the camera used. Many authors however propose heuristics for viewpoint and image selection [218]. In most cases, $20 - 30$ images are sufficient for good calibration results. The dose for the operator for the calibration steps is minimal since she or he only has to move the plate or the C-Arm for a few degrees between the activations.

Concerning the underlying projection model, in our experiments the classical camera model of central perspective projection, nonlinear radial lens distortion, and tangential distortion have been employed. This amounts to 7 intrinsic camera parameters: one for focal length, two for principal point, two radial distortion coefficients and two tangential distortion coefficients. While many alternative models have been proposed for the thorough description of C-Arms and X-Ray cameras [23, 132, 212], this standard is sufficient, since model imprecision is far below the accuracy of the "virtual" reference target which was taken by a CT-scanner.

A total of 152 calibration images were acquired with different rotation angles and inclination angles. We use a *Siemens Siremobil Compact L* for our experiments. The calibration method proposed in [251] has been adapted to the setup at hand. To gain reference information on the calibration normal, the acrylic glass target is scanned with a CT-scanner. Hence we get reference coordinates of all steel spheres with a certain measurement error. Scanning this target with the same scanner as used for the patients avoids possible inaccuracies of the CT-scanner for later image based registration tasks.

The actual C-Arm pose estimation can subsequently be performed by placing the unique pattern target below the patient. Resulting C-Arm X-Ray images are shown in Figure 7.3. Efficient post-processing is accomplished by the following steps. First the images are undistorted with the previously determined fourth order polynomial approximating the camera distortion. To segment the spheres, we make use of the fact that they appear in the image at almost constant size, so they can easily be detected and separated from the background by variational filtering techniques as proposed by Pock et al. [191]. Then a *random sample consensus* (RANSAC) algorithm is applied to find the vanishing point of the pattern lines, and subsequently the lines themselves. Correction of the perspective line distortion leads to an orthographic view of the displayed spheres. Correspondences between the displayed spheres and the complete target are established through a comparison of the observed sub-pattern with the complete known pattern. Due to the uniqueness of sub-patterns of a certain size, each steel sphere can exactly be located on the target.



(a) (b) (c)

Figure 7.3: Examples for X-Ray images taken by a *Siemens Siremobil Compact L* including our pose estimation target. The images were taken during a prostate biopsy. Therefore the biopsy needle and the head of the ultrasound probe are also visible. Figure (a) is an example for a quite narrow field of view and therefore only a small part of the target can be seen, but the C-Arm's pose can be reconstructed. (b) illustrates the distortion correction process with a 4^{th} order polynomial. (c) shows the detected beads after equalization and denoising based on variational methods. We are using a ROF-Chambolle model with a L1 norm in this case [191]. Line detection is done with a RANSAC based algorithm and subsequent perspective equalization.

The X-Ray images will not necessarily show a complete sub-pattern of the whole target. Some spheres may be occluded by medical instruments, e.g., an ultrasound sensor. Therefore we use a binary XOR comparison with all sub-patterns from the original target and choose the position which produced the smallest error. The resulting point correspondences are used to estimate position and orientation of the calibrated X-Ray camera relative to the target. To cope with the presence of outliers and occlusion, we employ the *direct linear transformation* [93] (DLT) method within a RANSAC framework to get an initial pose estimate. This pose estimate is further refined on the inlier points by nonlinear optimization with an iterative steepest descent optimization method [93].

7.2 Target accuracy results

For our experiments, X-Ray images were taken in the range of ±20° craniocaudal and ±30° lateral rotation using a Siemens Siremobil Compact L during real biopsy interventions.

Hence the C-Arm pose was not optimized in any way, and our method was tested with an uncontrolled surgical scenario. The mean reprojection error during pose estimation settled down between **0.67 pixels** and **0.89 pixels** for different datasets. This value is calculated through cross-validation, taking randomly 75% of the detected inliers to calculate the intrinsic camera parameters and the remaining 25% to calculate the error.

For a subsequent 2D/3D registration, ideally two images acquired at a relative angle of 90 degree are required. Consequently, images at angles of 180 degree are not considered. Due to mechanical restrictions during (prostate) investigations, we examined the operational range limited to ± 45°. Visible beads are masked out for subsequent registration.

To evaluate the accuracy of the C-Arm pose estimation, we add Gaussian noise to the detected positions in the undistorted X-Ray images with a mean of a quarter of the steel spheres' radius considering the calculated reprojection error. A subsequent principal components analysis results in a variance maximum of **0.33 mm** for the X-Ray source translation and **0.15**° for the X-Ray source rotation on the principal axis.

We are aware of the adverse effects of gravitation and the earth magnetic field on calibration consistency. However, since we operate the C-arm at a small number of repeatable poses, we can compute separate calibrations for each of them. Interior effects like pincushion/barrel distortion are sufficiently compensated by the radial distortion model, as the reprojection error suggests. Furthermore, Jain [103] noted that a faulty calibration has only little influence at least on relative poses.

To evaluate our target in terms of unambiguity and robustness against occlusions, we compare each sub-pattern of a given size with all other sub-patterns. An array of XOR gates indicates the diversity of two sub patterns and can therefore be used as an error measurement. To evaluate the robustness against occlusion we artificially increase the number of bit-inversions until the first detection failure occurs. A bit-inversion corresponds to the occlusion of a steel sphere in a sub-pattern (e.g., due to a medical instrument) or a false positive detection due to noise.

Our experiments show that 19 to 24 positions are sufficient to guarantee uniqueness assuming that the complete pattern are optimized for the size of the target and that a binary “0” takes two positions. However, between lines, two zero (i.e., empty) positions may follow each other. Therefore, the required unique sub-pattern size depends more on the number of visible lines than on the number of visible positions within a line.

19 positions would allow for only two occlusions according to Figure 7.4. However, in the clinical setup we detect 50 spheres and more, even for disadvantageous views (e.g., Figure 7.1(a)). This allows for 13 and more occluded positions (a US probe, for example, usually occludes 3-7 positions). Consequently, unambiguity is getting better the more lines are visible.

Figure 7.4 illustrates the growth of the required window size for an increasing number of occlusions. Above a sub-pattern size of 60 positions, half of the positions could be occluded before a detection failure occurs.

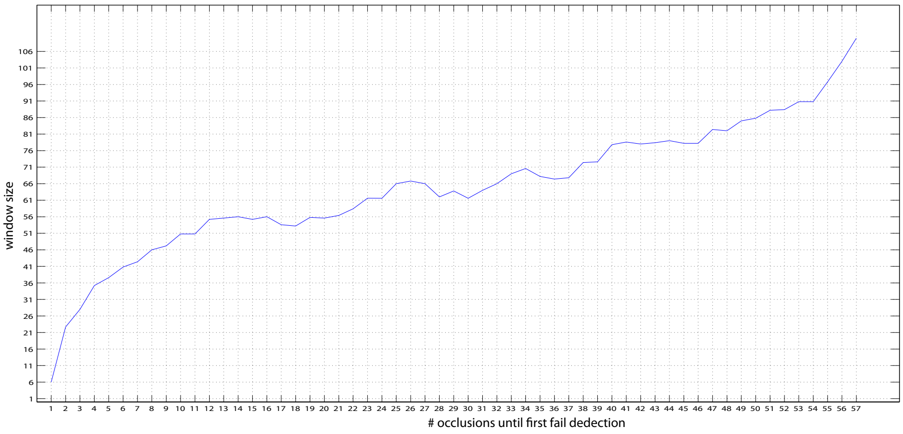


Figure 7.4: The curve shows the minimal window size containing a unique sub sub-pattern as a function of the number of occluded positions. Above 60 visible positions, half of the sub-pattern's positions can be occluded until a detection failure occurs.

7.3 Overall prostate biopsy registration results

The pelvic bone has a distinctive anatomy and is located close to the prostate, it is therefore well suited for the reconstruction of the patient's position during prostate biopsy or surgery.

The runtime for a complete pose estimation is the sum of the runtimes for image undistortion, line detection, segmentation of spheres, localization of the sub-pattern on the target, camera pose estimation from the known point correspondences and the registration refinement done by the AD method presented in Chapter 5. C-Arm camera calibration and analysis of the target's CT-scan has to be done only once and can therefore be performed offline. The RANSAC based algorithms (line detection and camera pose estimation) are the critical parts and have to be considered for runtime measurements. The variational methods algorithm for sphere detection is hardware accelerated (i.e., executed on a graphics processing unit) and hence not comparable to CPU executed tasks. Its runtime is in the range of a hundredth of a second, depending on the graphics board. On a PC (Intel QuadCore 3.16 GHz CPU, 12 GB RAM, NVidia Geforce GTX480) the mean runtime for line detection and calculation of the point correspondences is **0.24 seconds** and for subsequent pose estimation with these correspondences **0.01 seconds**. Consequently, we are able to reconstruct the pose of a calibrated C-Arm within one second from one image displaying a quite small part of our target. Together with the results for an AD based subsequent final registration from Section 5.5 the overall runtime for prostate registration is within in the range of a few seconds. The final registration accuracy is defined by the accuracy of the AD method and therefore below 0.6 mm as also shown in Section 5.5.

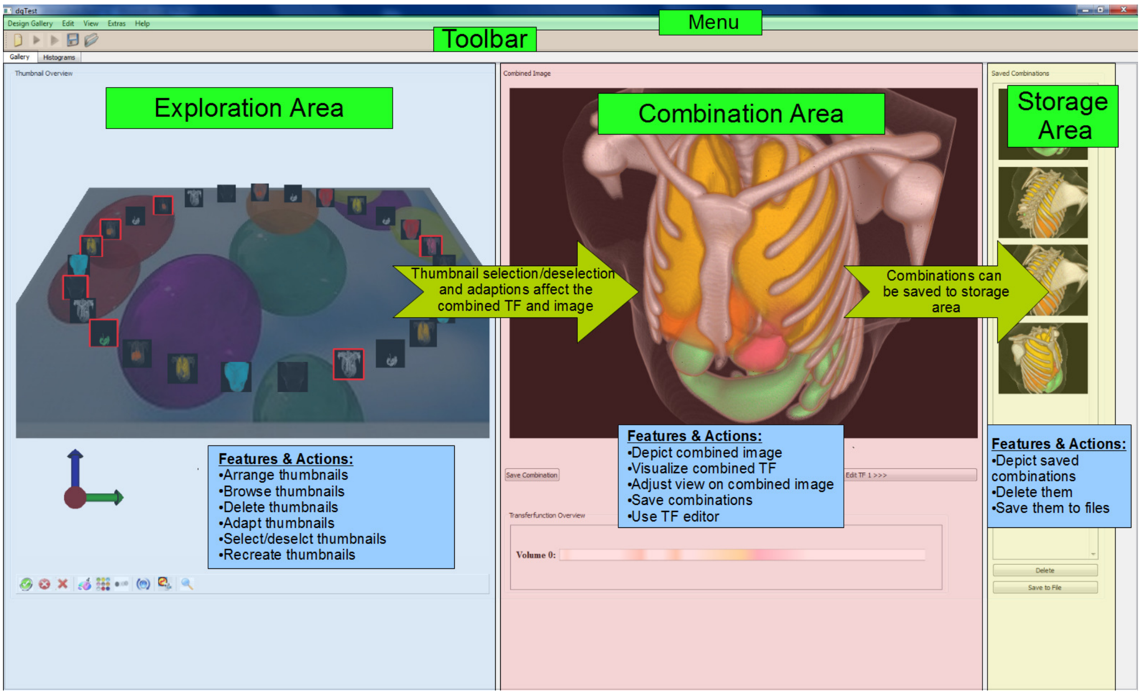
Chapter 8

Automatically generated transfer function design-galleries

As described in detail in the Master's Thesis of a collaborating student Markus Muchitsch [163], interactive multi volume rendering can also be used for automatic generation of high dimensional transfer functions.

A common problem in medical data visualization is the generation and modification of proper transfer functions for volumetric datasets. Especially untrained staff quickly decides to omit the use of 3D representations of the dataset, if the desired information cannot be shown with standard methods or predefined (device specific) transfer functions. Transfer function design is considered as cumbersome task because of the large number of degrees of freedom of transfer functions and the lack of constraints and guidance of conventional design editors [114]. In a basic editor, a polyline is used to define a simple 1D transfer function over the whole transfer function domain. It can be adjusted by moving control points between its segments. Thereby each control point has two degrees of freedom. Assuming that every region of interest is defined by approximately three control points, there exists a large number of potential transfer functions. Since the design process is neither constrained nor guided according to the used data set, a user may only obtain a desired visualization by finding appropriate control points positions using trial and error interaction. Manual design with such an editor is especially problematic, because there is no straight forward correlation between a transfer function adaptation and a resulting modified data set visualization. Small changes in the transfer function domain frequently result in strong and unexpected changes of the renderer output. Hence, transfer function design based on simple editors can be considered as an unintuitive, time consuming and tedious trial and error work.

Furthermore a potential user expects a clear distinction of data set features, regardless of the used modality. For this purpose, multivariate data sets and corresponding multidimensional transfer functions are essential. By describing voxels with multiple data values the separability of structures is enhanced. Unfortunately, the large number of degrees of



Thumbnail selection/deselection and adaptions affect the combined TF and image

Combinations can be saved to storage area

Features & Actions:
- Arrange thumbnails
- Browse thumbnails
- Delete thumbnails
- Adapt thumbnails
- Select/deselct thumbnails
- Recreate thumbnails

Features & Actions:
- Depict combined image
- Visualize combined TF
- Adjust view on combined image
- Save combinations
- Use TF editor

Features & Actions:
- Depict saved combinations
- Delete them
- Save them to files

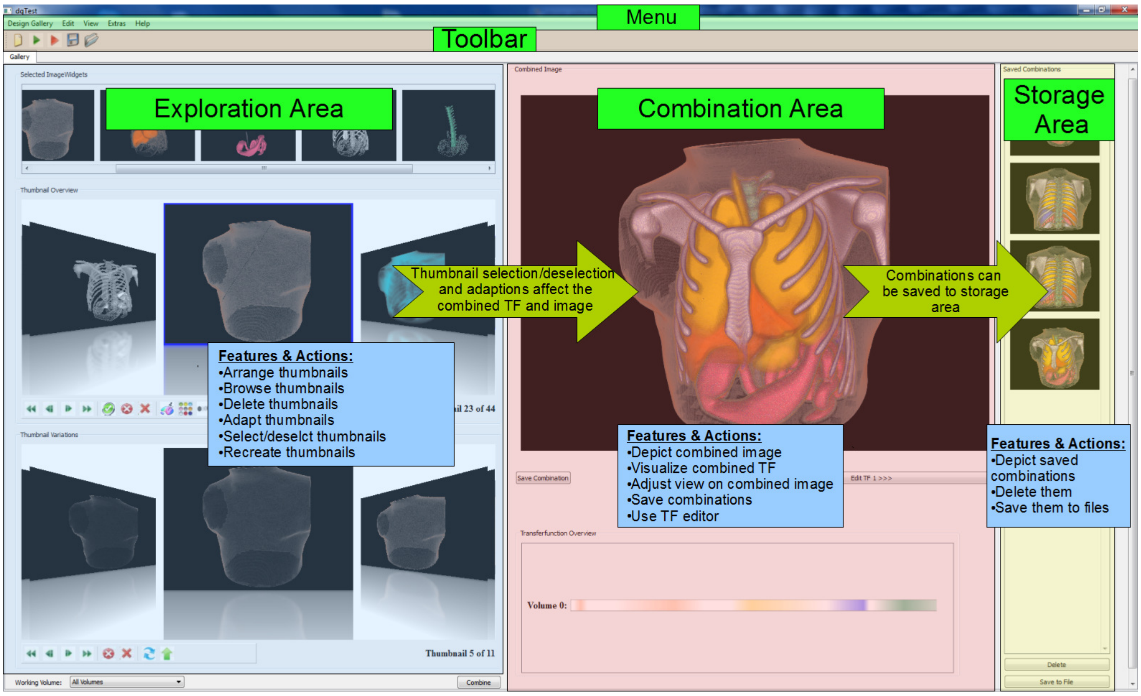
Figure 8.1: Design Gallery User Interface, using the data mountain technique. Green rectangles contain labels of the single areas. The notes in the remaining colored rectangles describe the most important features and actions of each area. The green arrows show the most important effects from actions in one area on another. All DVR widgets use the rendering system from Chapter 3.

freedom of multi-dimensional transfer functions also complicates their design. A possible approach to reduce the complexity of high-dimensional transfer functions are separable transfer functions. They represent a single multi-dimensional transfer function by a number of lower dimensional ones which are combined according to a certain scheme. However, without additional measures they have the potential to falsely classify features.

8.1 Our approach

In order to tackle the described problems this work presents a transfer function design gallery which deploys an automatic data centric approach in combination with an intuitive user interface. Moreover the render system from Chapter 3 and Chapter 4 provides important functionality to make the over all concept work.

The data centric approach identifies value ranges of features of interest in the transfer function space using the alpha histogram and partial range histogram method. Based on this method, initial transfer functions are created. Each of them defines the visualization of one data set structure. Automatic generation of initial transfer functions can be


Menu
Toolbar
Exploration Area
Combination Area
Storage Area
Thumbnail selection/deselection and adaptions affect the combined TF and image
Combinations can be saved to storage area
Thumbnail 5 of 11
Volume 0:
Delete
Save to File
Combine

Features & Actions:
- Arrange thumbnails
- Browse thumbnails
- Delete thumbnails
- Adapt thumbnails
- Select/deselct thumbnails
- Recreate thumbnails

Features & Actions:
- Depict combined image
- Visualize combined TF
- Adjust view on combined image
- Save combinations
- Use TF editor

Features & Actions:
- Depict saved combinations
- Delete them
- Save them to files

Figure 8.2: Design Gallery User Interface, using the coverflow technique. Green rectangles label the single areas. The notes in the remaining colored rectangles describe the most important features and actions of each area. The green arrows show the most important effects from actions in one area on another. All DVR widgets use the rendering system from Chapter 3.

considered as a restriction of the transfer function space to interesting regions and saves users a lot of tedious trial and error work. However, the initial transfer functions alone are not sufficient. Users must still be able to adapt and combine them according to their visualization task. This can be achieved via the design gallery user interface. It neatly arranges a set of thumbnails, representing the initial transfer functions. By interacting with them, users can compose desired visualizations without any knowledge about the concept of transfer functions. In order to allow a simple and efficient exploration of the structures depicted on the initial thumbnails, the system provides two user interface versions based on the approved coverflow [40] and data mountain [198] approach. Moreover the presented framework allows a multi-dimensional classification of data set structures to ensure their clear distinction. This is implemented using separable transfer functions. In order to eliminate false classifications resulting from them, an additional color test in the used renderer followed by a homogeneity test in the design gallery are performed.

The two user-interface approaches for a guided transfer function process are outlined in Figure 8.1 and Figure 8.2. A data centric pre-processing step, combining the alpha histogram [136] and partial range histogram [135] is performed on each input dataset

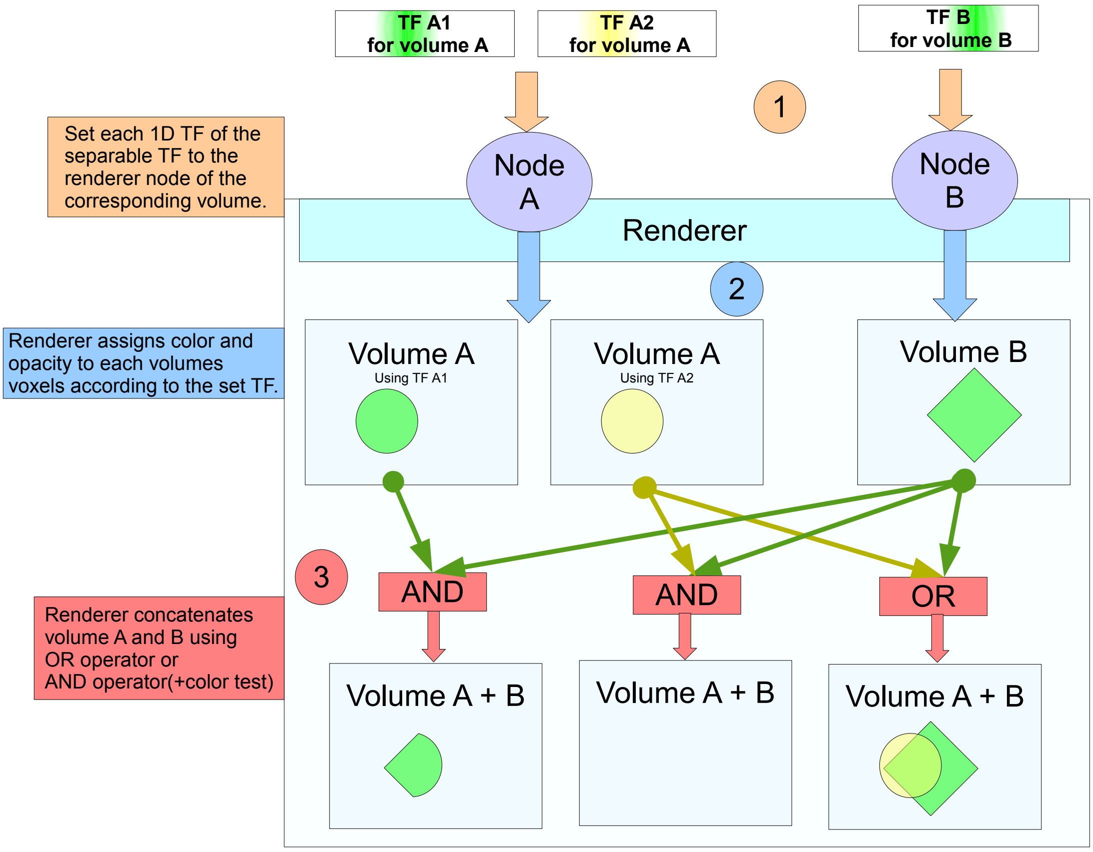

TF A1
for volume A
TF A2
for volume A
TF B
for volume B
1
Set each 1D TF of the separable TF to the renderer node of the corresponding volume.
Node A
Node B
Renderer
2
Renderer assigns color and opacity to each volumes voxels according to the set TF.
Volume A
Using TF A1
Volume A
Using TF A2
Volume B
3
AND
AND
OR
Renderer concatenates volume A and B using OR operator or AND operator(+color test)
Volume A + B
Volume A + B
Volume A + B

Figure 8.3: Abstract presentation of how the renderer may create a data set visualization for a 2D separable transfer function in dependence of the concatenation mode from [163]. The *OR* concatenation depicts the emphasized structures from both volumes (see bottom right *Volume A+B*). The color of superimposed parts is mixed. The *AND* concatenation depicts superimposed parts of the two volumes if they have a similar color and opacity (see bottom left *Volume A+B*). Otherwise the *AND* concatenation results in an empty visualization (see bottom center *Volume A+B*).

before the user can select specific combinations of identified structures.

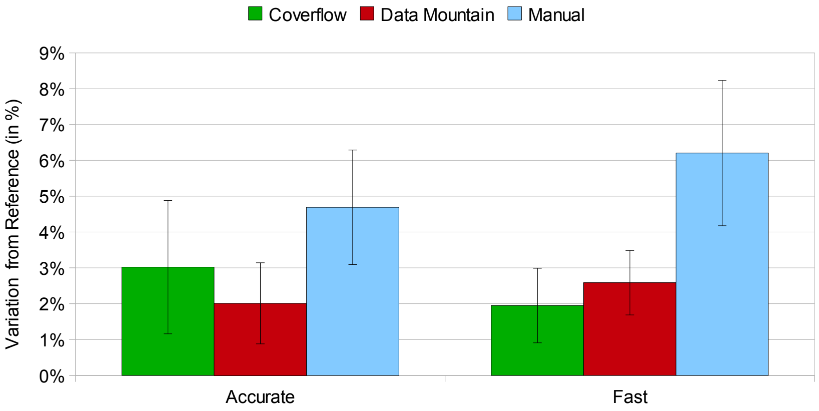
This approach can also be used to determine structural connections between registered datasets from different modalities. Therefore the rendering system from Chapter 3 was modified to allow simple logical connections of the input data sets and their transfer functions and to support separable transfer functions. Consequently, a set of 1D transfer functions is responsible for one visualization of a multivariate data set. After the automatic determination of initial transfer functions, a color test in the renderer in combination with a homogeneity test in the design gallery allow to detect and discard separable transfer functions which erroneously classify features. This approach is outlined in Figure 8.3. The rendered images corresponding to the remaining separable transfer functions are arranged in the coverflow or data mountain based exploration area. User interactions to adapt and combine the depicted structures stay as simple and efficient as for the one-dimensional application of the design gallery. The multi-volume rendering abilities of our rendering system are essential for that approach.

8.2 Results

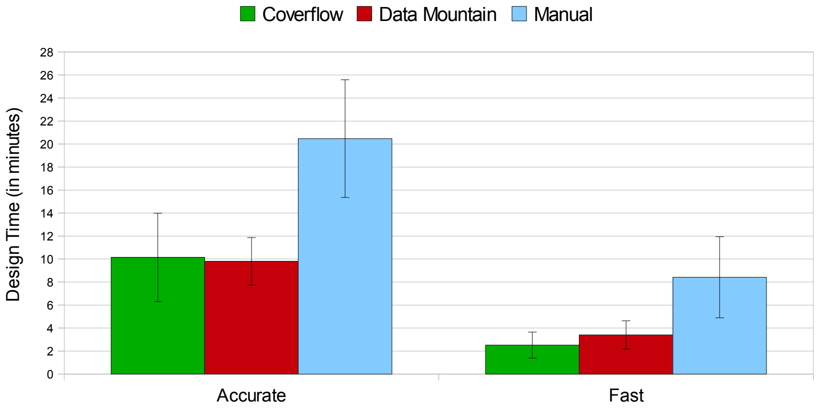
A usability study amongst 12 users with computer skills, performed by Muchitsch [163] shows the advantages and disadvantages of both approaches. From the data acquired throughout the survey, a number of objective and subjective criteria for the evaluation and comparison of the two proposed design approaches and the classical (manual) design approach can be obtained.

Objective criteria are the quality of a created visualization and the required time for this task. In order to obtain quality values, we create a visualization with all saved transfer functions of each user and compare them to the reference image. All visualizations must be considered from the same point of view. The variation between reference and user visualization describes the quality of the latter. The subjective criteria are derived from the quantifiable interview questions. They include the subjective design speed, intuitivity and clearness.

All survey results are summarized in Figure 8.4 and Figure 8.5. All results show clear evidence that the *data mountain* and the *cover flow* approach are better than manual transfer function editing in terms if design speed and accuracy. Manual transfer function editing has clearly the longest design time, and shows the worst results. The *cover flow* approach tends to be the best solution. However, this might be founded in the fact that current multimedia hardware makes extensive use of this concept and that the participates are already used to this kind of representation.



(a)



(b)

Figure 8.4: Quantitative Evaluation. We have asked 12 users to perform two different tasks. A given rendered image from volume data had to be re-designed with the *data mountain* and the *cover flow* approach in one task as accurate as possible and in a subsequent task as fast as possible. (a) shows the absolute average pixel difference to the reference image in percent, hence the accuracy of the approaches and (b) shows the time which was required to until the the subject could not see any more difference between his result and the given reference image. The results show evidence that *cover flow* provides the most accurate and fastest interface.

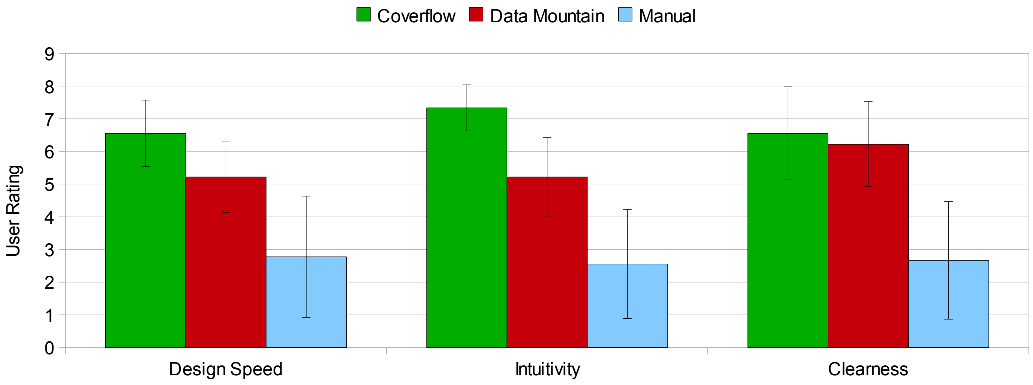


Figure 8.5: Qualitative Evaluation. After the given tasks have been accomplished the users have been asked to rate the presented methods. The *cover flow* approach tends to be the most liked interface, while a good but manual transfer function editor has been sensed as cumbersome and unclear.

PART IV - CONCLUSION

Chapter 9

Summary and conclusion

This thesis is concerned with the algorithmic shortcomings of ray-based 3D image synthesis methods which still prevents a wide use of these methods in the clinical practice. We have investigated the basic ray-based image generation kernels and have found several novel ways to improve and/or specialize them for certain applications. Before investigating these methods, we have tried to answer the sword-of-Damocles question of Computer Graphics in Medicine: Is 3D necessary for the clinical practice at all? This question is not necessarily obvious, but arises invariably when working closely together with experts from Medicine. 2D slice view investigation is the most common way for standard diagnostics and can also be efficiently done by trained experts. Furthermore, this method provides the highest level of accuracy and detail which makes it hard to compete with. However, we could show clear evidence by doing a survey amongst 24 experts from radiology and surgery (54% radiologists, 29% surgeons, 16% other clinictians of whom 50% have more than five years of professional experience) that 3D image synthesis is indeed necessary and welcome in the clinic, but that it strongly depends on the kind of diagnostic or interventional application. We have summarized our main motivation for this thesis in four observations (*cf.*, *AOH1-4*), which have also been assesed by the survey participants. A large majority agrees with our hypothesis and give also indication of the main medical areas of 3D image synthesis. The result can be summarized as follows: In most cases 3D image synthesis is essential if the data input space gets too large for human cognitive abilities and for medical inter-disciplinary communication (e.g., intervention planning between radiologists and surgeons). Furthermore, we investigated the level of satisfaction with state-of-the-art 3D image synthesis algorithms (mostly DVR) which indicated the necessity for improvements of 3D algorithms in terms of accuracy and rendering speed. Both problems have been attacked in this thesis.

To overcome the shortcomings of conventional volume processing attempts we discuss how to optimize a standard ray-casting kernel (Algorithm 2 in Section 2) and its ray setup parts. Therefore, we present in Chapter 3 a ray-based image generation system which solves the problem of multi-modality in direct volume rendering. This rendering system forms one of the major contributions of this thesis and the basis for the following optimizations and improvements which are presented in Chapter 4. A condensed version of this Chapter appeared in [105]. Chapter 4 shows how object features which are especially prominent for the human visual system can be exploited to guarantee a minimum rendering speed. This system is finally used amongst others in Chapter 6 to allow tool trajectory guidance during interventions and further applications, which are performed with an assisting medical scanner and multiple modalities. In Chapter 5 we show also how to specialize ray-based image generation and show its essential use for a 2D/3D registration problems.

For each of the presented algorithms, we discuss an application in part three of this thesis. We show how our novel algorithms can be applied to practical applications. Therefore we present multi-volume rendering for medical tumor accessibility planning, ray-casting kernel specialization for a real 2D-3D image registration problem during prostate surgery and automatic transfer-function generation for unskilled users. All parts of this thesis have been fully evaluated by experts from the medical domain and computer science. These parts show furthermore strong evidence, that our hypothesis in this thesis are not only valid but also directly applicable in many ray-based rendering environments.

9.1 Lessons learned

Research on novel methods for 3D image synthesis respectively direct volume rendering has sometimes a tough act to follow in all target communities. For some Computer Science researchers (or paper reviewers) the problem of real-time DVR has been solved with the first shader-based implementations on GPUs. In medicine simple but effective methods like texture slicing and MIP are sometimes considered as sufficient. Therefore the need for novel methods which are at first glance not ground-braking is often called into question. Furthermore, medicine has a very long – if not the longest – acceptance and approval time for new methods. The time until a new drug or a new diagnostic method is approved by the US Food and Drug Administration (FDA) is often cited as the gold standard in this context. In 2008 this time was 113 days [99], not including the time until a new method was made ready for the market. The whole process from the first publication of a new method until the final use in the clinical practice can take up to ten years overall. From our own further work [105, 194], we know, that such periods are far beyond the expected time for a PhD thesis. The cited work was developed in 2004, published first in 2007 with impressing and very clear results for the future pre-diagnostics of pulmonary hypertension but is still in the phase of making ready for the market. A possible FDA approval might happen within the next five years and after that it might take another few years until this method gets standard.

Of course, besides these general problems we had to fight a huge number of practical and specific technological problems when developing our methods. The following list gives only the most severe but also the at least obvious examples:

- **Building a X-Ray registration target.** Besides the fact that a device for the operating theater has to be sterilized without taking damage before every use, a X-Ray device also has to show homogeneous attenuation values and preferably no refractions. The first problem was solved easily. We wrapped a temperature and chemical (basically C_2H_5OH) insensitive adhesive foil around the target. The second problem required more experiments. Since we had to drill holes in an acrylic class target to equip it with steel markers the emerging acrylic-glass air border was a source for X-Ray refraction and inflection. We could solve that problem by filling the wholes with agarose gel. This gel is commonly used for genomics and agarose gel electrophoresis. Experiments and research in tables showed that this gel has nearly the same X-Ray attenuation coefficient as acrylic glass when using it in the same concentration as used for agarose gel electrophoresis. Furthermore agarose is easy to use, non-toxic and keeps its properties for a long time if it is conserved beneath a foil like in our case.

- **Using tools from Automatic Differentiation (AD).** AD is a very experimental research field and so are the provided tools. Automatically differentiating the ray-casting kernel required a lot of manual refinement and produced overall unreadable and hardly comprehensible output. Furthermore neither Cg, GLSL nor CUDA provide full C++ support, which restricts the use of AD to only one method (source transformation). We expect that operator overloading would produce better output and probably also faster algorithms, but not yet on a GPU.

- **CUDA vs. GLSL.** In Chapter 3 we re-implemented and improved a lot of parts from a rendering pipeline completely in software which are indeed already implemented as fixed function parts directly in hardware on every standard GPU. While CUDA supported a full flexible access to all memory units one the GPU and supported data scattering, GLSL did not at that time. GLSL is able to use a lot of fixed function commands but there is still no way to outsource critical parts in low-latency memory units. In the meantime, GLSL at least supports data scattering and an early version of CUDA interoperability. This is the reason why the author and his colleagues started to replace for example the software rasterization in our polyhedral rendering systems with GLSL interoperability, using the hardware rasterization unit itself. These hard-wired units are of course unbeatable fast and have shown already very promising results in combination with our rendering kernel.

- **VTK vs. Coin3D.** While Scene Graph implementations like Coin3D [*] or OpenSG [†] are very common for the Augmented Reality community, these libraries are quite uncommon for medical applications. *The Visualization Toolkit (VTK)* [‡] is much more

[*] http://www.coin3d.org/
[†] http://www.opensg.org/
[‡] http://www.vtk.org/

common and in combination with the *Segmentation & Registration Toolkit (ITK)* [§] it provides a much more powerful environment for medical applications. This is mainly because of the vast amount of already implemented specialized medical operations, transformations and data formats. Hence, with 2010, we changed our basic development environment slowly from Scene Graph-based libraries to a combined library of ITK, VTK and many interoperability additions which is called *The Medical Imaging Interaction Toolkit (MITK)* [¶]. This system was made public available end of 2009, which prevented an earlier use of it. For AR we just had to define an interface to *Opentracker* [‖] which is part of the in-house AR/VR software environment *Studierstube*[**].

- **New potential from Nvidia OptiX.** In 2010, a few months after we published our polyhedral DVR system, Nvidia published their novel ray-based rendering system OptiX. This system is mainly intended for ray-tracing of photorealistic scenes. Nevertheless, its early version provided already enough flexibility, and our aim for a small and comprehensible core component of our DVR system allowed us to quickly integrate DVR possibilities in OptiX. We used the resulting new system, to steer the ray setup in Chapter 4.

- **Designing an interventional AR prototype.** Even tough our institute provides a vast selection of tracking devices and display devices, we were restricted to the clinical preconditions during the design process. The infrastructure available in an operating theater is simple but well considered. Nothing non-sterile must get into the sterile safety volume around the operation field, the tracking possibilities are very restricted and the performing doctor must not be disturbed by additional devices. We had to rebuild these conditions in our lab, and we used therefore devices like a two-camera optical tracking setup and a movable monitor with attached camera as see-through augmentation. In the clinical practice only two-camera optical tracking systems are currently available (for example NDI) and sometimes also a ceiling-mounted freely movable monitors. The only addition we made was to attach an outside-in tracked camera on the backside of the monitor to facilitate the impression of a see-through device.

Finally a personally learned lesson: Getting a PhD in Computer Science is definitely not easy, completely underpaid and often very disappointing. It requires more than reading and being able to reproduce the work of others or attending lectures. Like in leading position, one has to overcome one's own self-doubts and find one's own source of motivation, sometimes also against the will of others. However, perhaps no other *"vocation"*

[§] http://www.itk.org/

[¶] http://www.mitk.org/

[‖] http://studierstube.icg.tu-graz.ac.at/opentracker/

[**] http://studierstube.icg.tu-graz.ac.at/

provides so many opportunities for personal and professional development and the possibility to be worldwide one of the best in a certain field of expertise. Even though this field might be extremely narrow, it is a life-changing experience.

9.2 Directions for future work

Our methods give also room for improvements and further applications. The following paragraphs give a short outline of possible future additions and changes. Most are motivated by conceivable technical improvements of hardware and its driver APIs.

Multi-volume DVR and perceptual ray setup. GLSL and CUDA improved their mutual interoperability and their feature set. GLSL's new features with OpenGL 4.2 concern mainly the ability to write into arbitrary memory positions. Earlier versions allowed only to write in the same fragment position as it has been read from. Additionally it is possible with CUDA to access texture memory directly and also to write directly in OpenGl memory space. CUDA and OpenGL memory space have been separated so far. These new options give completely new opportunities for certain parts of our multi volume rendering system. Especially the optimized multi-level single-step rasterization module can benefit a lot from switching (back) the hardware built-in rasterization units. However, some of the CUDA low-latency high speed memory access options would get lost that way. Definitely, an appropriate trade-off between fixed-function GLSL hardware features and CUDA register units will have to be found.

The volume texture storage is furthermore only a simple container with fixed size slots in the presented implementation. A bricking approach as proposed by Weiskopf *et al.* [248] and binary space partitioning examination like done by Lindholm *et al.* [130] would definitely improve our systems abilities. However, such considerations get more interesting when trying to make a product out of a research prototype.

Full utilization of the GPU is extremely difficult to achieve for complex graphics algorithms with real-time demands. Building a toolset, similar to common CPU scheduling and memory management strategies that allows harvesting of the full GPU power for a general class of real-time volume graphics algorithms, would greatly improve the abilities and rendering speed of our DVR approach. The "mega kernel" approach as presented with Nvidia's OptiX [183] ray tracing engine is already one first simple example system targeting in that direction. For polyhedral multi-volume rendering a more complex scheduling strategy would be required, but OptiX can define the right starting point. However, to allow these options a foreseeable interface between CUDA and OptiX will have to be released by Nvidia first.

For the presented perceptual ray setup, we plan to perform a larger user study with different ray-tracing materials and ray-casted volumetric objects. From such a work we expect a qualitatively founded classification of salient object features to answer the question which feature works best for a particular type of object by means of human perception.

In this work we show evidence that contours are the most valuable feature of an object in terms of mathematically estimated image error and quality. However, to better qualify the remaining, less distinctive features, a deeper analysis will have to be performed with human subjects. Furthermore, we want to test our algorithm with a giga-pixel display for real-time interaction with tremendously huge volumetric data sets.

Single kernel specialization. As soon as GPU programming languages will support a full C++ instruction set we can try to extend our algorithms with the "Automatic Differentiation by Overloading in C++" technique (ADOL-C) as presented by [84]. This technique will likely be able to produce more comprehensible output and probably also gain additional execution speed. The registration initialization target can additionally be improved by its size and the removal of the steel markers could be improved by incorporating the 16 Bit value range of X-Ray images. At least regions of a high density, which are covered by the markers could be still noticeable within the signal-noise ratio.

The original implementation was based on Nvidia's Cg, which is in the meantime more or less outdated. A newer implementation in CUDA could solve some of the issues we had with manual code derivative refinement, since CUDA is more related to a standard C interface as it is the case with Cg. However, it should not be expected, that CUDA will provide more performance or higher accuracy, because of the higher memory transfer requirements for an actual render pass. First, when CUDA provides a full C++ feature set, other AD methods might introduce new possibilities for this method.

Further applications. Our approaches are applicable to various medical fields. The possibilities of algorithms which guarantee a certain frame-rate without loosing too much image quality are virtually endless for Augmented Reality applications involving volumetric objects or aiming for photorealism with ray-based algorithms. Also normal volumetric data set investigation can benefit from our approaches since many modern diagnostic procedures rely on multiple modalities which are still investigated side by side. Furthermore most off-the-shelf visualization systems omit detail during interaction, which highly influences the diagnostic value of those systems. To preserve important features like vessels or other vulnerable structures, also during interactive investigation, our graceful degradation approaches can be directly applied. Of course, one has to consider, that our methods are intended for a medical environment, which implies an extensive testing and medical approval procedure before they come into operation. The registration approach can be used in every localization problem involving a C-Arm. However, currently we concentrate on stiff or rigid structures. A further improvement could be a permanent registration approach to allow also interventions on non rigid structures like vessels.

We could also apply our rendering algorithms to non-medical environments, like game engines and hand held devices which provide only reduced computational power. However, this is far beyond the scope of this thesis and would require much more further research and funding.

Bibliography

[1] (2007). *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*. NVIDIA Corporation.

[2] Aila, T., Miettinen, V., and Nordlund, P. (2003). Delay streams for graphics hardware. *ACM Transactions on Graphics (TOG)*, 22(3):792–800.

[3] Akeley, K. (1993). Reality engine graphics. In *Proceedings of 20th annual conference on Computer graphics and interactive techniques SIGGRAPH '93*, pages 109–116, New York, NY, USA. ACM.

[4] Akenine-Möller, T., Haines, E., and Hoffman, N. (2008). *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA.

[5] Alhonnoro, T., Pollari, M., Lilja, M., Flanagan, R., Kainz, B., Muehl, J., Mayrhauser, U., Portugaller, H., Stiegler, P., and Tscheliessnigg, K. (2010). Vessel segmentation for ablation treatment planning and simulation. In *MICCAI (1)*, pages 45–52.

[6] Altrogge, I., Kröger, T., Preusser, T., Büskens, C., Pereira, P. L., Schmidt, D., Weihusen, A., and Peitgen, H. O. (2006). Towards optimization of probe placement for radio-frequency ablation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 4190 of *LNCS*, pages 486–493. Springer.

[7] Alyassin, A. M., Lancaster, J. L., Downs, J. H., and Fox, P. T. (1994). Evaluation of new algorithms for the interactive measurement of surface area and volume. *Med. Phys.*, 6:741–52.

[8] Amidror, I. (2002). Scattered data interpolation methods for electronic imaging systems: a survey. *J. Electronic Imaging*, 11(2):157–76.

[9] Anxionnat, R., Bracard, S., Ducrocq, X., Trousset, Y., Launay, L., Kerrien, E., Braun, M., Vaillant, R., Scomazzoni, F., Lebedinsky, A., and Picard, L. (2001). Intracranial Aneurysms: Clinical Value of 3D Digital Subtraction Angiography in the Therapeutic Decision and Endovascular Treatment1. *Radiology*, 218(3):799–808.

[10] Avila, R. S., Sobierajski, L. M., and Kaufman, A. E. (1992). Towards a comprehensive volume visualization system. In *Proceedings of IEEE Visualization '92*, pages 13–20, Boston, Massachusetts.

[11] Baegert, C., Villard, C., Schreck, P., and Soler, L. (2007). Multi-criteria trajectory planning for hepatic radiofrequency ablation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 4792/200 of *LNCS*, pages 676–684.

[12] Baran, I., Chen, J., Ragan-Kelley, J., Durand, F., and Lehtinen, J. (2010). A hierarchical volumetric shadow algorithm for single scattering. *ACM Trans. Graph.*, 29:178:1–178:10.

[13] Bauer, J., Schindler, K., and Karner, K. (2002). Plane parameter estimation by edge set matching. In *Proceedings Annual Conference of the Austrian Association for Pattern Recognition (AAPR)*, pages 29–36, Graz, Austria.

[14] Baur, W. and Strassen, V. (1983). The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330.

[15] Bavoil, L., Callahan, S. P., Lefohn, A., Comba, J. L. D., and Silva, C. T. (2007). Multi-fragment effects on the GPU using the k-buffer. In *Proceedings of ACM symposium on interactive 3D graphics and games*, pages 97–104.

[16] Bavoil, L. and Myers, K. (2008). Order independent transparency with dual depth peeling. Tech. rep., NVIDIA.

[17] Beyer, J., Hadwiger, M., Wolfsberger, S., and Buhler, K. (2007). High-quality multi-modal volume rendering for preoperative planning of neurosurgical interventions. *IEEE Trans. on Visualization and Computer Graphics*, 13:1696–1703.

[18] Bischof, C., Khademi, P., Mauer, A., and Carle, A. (1996). Adifor 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Computational Science & Engineering*, 3(3):18–32.

[19] Bischof, C. H. and Bücker, H. M. (2000). Computing derivatives of computer programs. In *Modern Methods and Algorithms of Quantum Chemistry: Proceedings, Second Edition*, volume 3 of *NIC Series*, pages 315–327. NIC-Directors, Jülich.

[20] Blaas, J., Botha, C., Majoie, C., Nederveen, A., and Post, F. (2007). Interactive Visualization of Fused fMRI and DTI for Planning Brain Tumor Resections. In *Proc. SPIE Medical Imaging 2007*. paper nr 6509-60, accepted, to appear.

[21] Boada, I., Navazo, I., and Scopigno, R. (2001). Multiresolution volume visualization with a texture-based octree. *The Visual Computer*, 17:185–197.

[22] Borland, D., Clarke, J., Fielding, J., and Taylor, R. (2006). Volumetric depth peeling for medical image display. In *Proceedings of SPIE Visualization and Data Analysis*, pages 1–11.

[23] Brack, C., Gosse, F., Moctezuma, J., Roth, M., and Schweikard, A. (1997). Towards accurate x-ray-camera calibration in computer-assisted robotic surgery. *Proc. Computer-Aided Radiology*, pages 721 – 728.

[24] Brecheisen, R., Platel, B., Vilanova, A., and ter Haar Romenij, B. (2008). Flexible GPU-based multi-volume ray-casting. In *Proceedings of Vision, Modelling and Visualization*, pages 1–6.

[25] Brown, D. C. (1971). Close range camera calibration. *Photogrammetric Engineering*.

[26] Bruckner, S. (2008a). *Interactive Illustrative Volume Visualization.* PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

[27] Bruckner, S. (2008b). *Interactive Illustrative Volume Visualization.* PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology.

[28] Bruckner, S., Grimm, S., and Kanitsar, A. (2006). Illustrative Context-Preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569.

[29] Bruckner, S., Gröller, M. E., Mueller, K., Preim, B., and Silver, D. (2010). Illustrative focus+context approaches in interactive volume visualization. In Hagen, H., editor, *Scientific Visualization: Advanced Concepts*, Dagstuhl Follow-Ups, chapter 10. The article was originally written in 2005 after the Dagstuhl Seminar on Scientific Visualization and reflects the state-of-the-art at that time.

[30] Brunenberg, E. J. L., Vilanova, A., Visser-Vandewalle, V., Temel, Y., Ackermans, L., Platel, B., and Romeny, B. M. T. H. (2007). Automatic trajectory planning for deep brain stimulation: a feasibility study. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, LNCS, pages 584–592. Springer.

[31] Burns, M., Klawe, J., Rusinkiewicz, S., Finkelstein, A., and DeCarlo, D. (2005). Line drawings from volume data. *ACM Trans. Graph.*, 24(3):512–518.

[32] Cabral, B., Cam, N., and Foran, J. (1994). Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 symposium on Volume visualization*, VVS '94, pages 91–98, New York, NY, USA. ACM.

[33] Callahan, S. P. and Comba, J. L. D. (2005). Hardware-assisted visibility sorting for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):285–295.

[34] Canteaut, A. (2005). Linear feedback shift register. In van Tilborg, H. C. A., editor, *Encyclopedia of Cryptography and Security*. Springer.

[35] Carpenter, L. (1984). The A-buffer, an antialiased hidden surface method. *ACM SIGGRAPH Computer Graphics*, 18(3):103–108.

[36] Carr, N., Mech, R., and Miller, G. (2008). Coherent layer peeling for transparent high-depth-complexity scenes. In *Proceedings of SIGGRAPH/EUROGRAPHICS symposium on graphics hardware*, pages 33–40.

[37] Catalano, C. and Passariello, R. (2005). *Multidetector-row CT angiography*. Springer.

[38] Chan, T. and Shen, J. (2005). *Image Processing and Analysis – Variational, PDE, Wavelet, and Stochastic Methods.* SIAM, Philadelphia.

[39] Chan, Y.-N., Dandamudi, S. P., and Majumdar, S. (1997). Performance comparison of processor scheduling strategies in a distributed-memory multicomputer system. In *IPPS '97: Proceedings of the 11th International Symposium on Parallel Processing*, pages 139–145, Washington, DC, USA. IEEE Computer Society.

[40] Chaudhri, I. (2010).

[41] Chen, C.-I. H. (2001). Synthesis of configurable linear feedback shifter registers for detecting random-pattern-resistant faults. In *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*, pages 203–208, New York, NY, USA. ACM.

[42] Chen, C.-I. H. and George, K. (2003). Automated synthesis of configurable two-dimensional linear feedback shifter registers for random/embedded test patterns. In *ISQED '03: Proceedings of the 4th International Symposium on Quality Electronic Design*, page 111, Washington, DC, USA. IEEE Computer Society.

[43] Chentanez, N., Alterovitz, R., Ritchie, D., Cho, L., Hauser, K. K., Goldberg, K., Shewchuk, J. R., and O'Brien, J. F. (2009). Interactive simulation of surgical needle insertion and steering. In *ACM Trans. Graph.*, pages 88:1–88:10. ACM.

[44] Clarke, T. and Fryer, J. (1998). The development of camera calibration methods and models. *Photogrammetric Record*, 16(91):51–66.

[45] Cohen, M. F., Wallace, J., and Hanrahan, P. (1993). *Radiosity and realistic image synthesis.* Academic Press Professional, Inc., San Diego, CA, USA.

[46] Colchester, A. C., Zhao, J., Holton-Tainter, K. S., Henri, C. J., Maitland, N., Roberts, P. T., Harris, C. G., and Evans, R. J. (1996). Development and preliminary evaluation of VISLAN, a surgical planning and guidance system using intra-operative video imaging. *Med. Image Analysis*, 1(1):73–90.

[47] Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H. S., Finkelstein, A., Funkhouser, T., and Rusinkiewicz, S. (2008). Where do people draw lines? *ACM Trans. Graph.*, 27.

[48] Corliss, G. F. and Griewank, A. (1993). Operator overloading as an enabling technology for automatic differentiation. Technical Report MCS-P358-0493, Mathematics and Computer Science Division, Argonne National Laboratory.

[49] Danskin, J. and Hanrahan, P. (1992). Fast algorithms for volume ray tracing. In *VVS '92*, pages 91–98.

[50] Davis, L. S. and Benedikt, M. L. (1979). Computational models of space: Isovists and isovist fields. *Computer Graphics and Image Processing*, 11(1):49–72.

[51] Dayal, A., Woolley, C., Watson, B., and Luebke, D. (2005). Adaptive frameless rendering. In *ACM SIGGRAPH 2005 Courses*, page 24.

[52] DeCarlo, D. and Rusinkiewicz, S. (2007). Highlight lines for conveying shape. In *NPAR'07*, pages 63–70.

[53] DeFanti, T. A., Brown, M. D., and McCormick, B. H. (1989). Visualization: Expanding scientific and engineering research opportunities. *Computer*, 22:12–25.

[54] DiMaio, S., Archip, N., Hata, N., Talos, I.-F., Warfield, S., Majumdar, A., Mcdannold, N., Hynynen, K., Morrison, P., III, W. W., Kacher, D., Ellis, R., Golby, A., Black, P., Jolesz, F., and Kikinis, R. (2006). Image-guided Neurosurgery at Brigham and Women's Hospital. 25(5):67–73.

[55] Dippé, M. A. Z. and Wold, E. H. (1985). Antialiasing through stochastic sampling. *SIGGRAPH CG*, 19:69–78.

[56] Drebin, R. A., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. *SIGGRAPH Comput. Graph.*, 22:65–74.

[57] Edmondson, A. C., Bohmer, R. M., and Pisano, G. P. (2001). Disrupted Routines: Team Learning and New Technology Implementation in Hospitals. *Administrative Science Quarterly*, 46(4).

[58] Engel, Kraus, and Ertl (2001). High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, New York, NY, USA. ACM.

[59] Engel, K., Hadwiger, M., Kniss, J., and Rezk-Salama, C. (2002). High-quality volume graphics on consumer pc hardware. Course Notes for Course No. 42 at SIGGRAPH 2002, ACM SIGGRAPH.

[60] Engel, K., Hadwiger, M., Kniss, J., Rezk-Salama, C., and Weiskopf, D. (2006). *Realtime Volume Graphics*. A. K. Peters.

[61] Essert, C., Haegelen, C., and Jannin, P. (2010). Automatic computation of electrodes trajectory for deep brain stimulation. In *Proceedings of the 5th international conference on Medical imaging and augmented reality*, MIAR'10, pages 149–158, Berlin, Heidelberg. Springer-Verlag.

[62] Everitt, C. (2001). Interactive order-independent transparency. Tech. rep., NVIDIA.

[63] Eyles, J., Austin, J., Fuchs, H., Greer, T., and Poulton, J. (1988). Pixel-planes 4: A summary. In *Advances in Computer Graphics Hardware II (Eurographics'87 Workshop)*, pages 183–207, London, UK. Springer-Verlag.

[64] Falsetti, H., Kiser, K., Francis, G., and Belmore, E. (1972). Sequential velocity development in the ascending and descending aorta of the dog. *Circulation Research*, 31:328–338.

[65] Fernando, R. (2004). *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education.

[66] Filler, A. and Filler, A. G. (2009). MR Neurography and Diffusion Tensor Imaging: Origins, History & Clinical Impact . *Nature Precedings*, (713).

[67] Fiume, E., Fournier, A., and Rudolph, L. (1983). A parallel scan conversion algorithm with anti-aliasing for a general-purpose ultracomputer. In *ACM SIGGRAPH Comp. Graph.*, pages 141–150.

[68] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1996). *Computer Graphics: Principles and Practice in C*. Addison-Wesley Professional, second edition.

[69] Freund, J. and Sloan, K. (1997). Accelerated volume rendering using homogeneous region encoding. In *VIS '97*, pages 191–ff.

[70] Fung, W. W. L., Sham, I., Yuan, G., and Aamodt, T. M. (2007). Dynamic warp formation and scheduling for efficient gpu control flow. In *MICRO 40: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 407–420, Washington, DC, USA. IEEE Computer Society.

[71] Geveci, B., Ayachit, U., Baumes, J., Bostock, M., Ogievetsky, V., Wylie, B., Shead, T. M., Santos, E., Ropinski, T., and Praßni, J.-S. (2010). DIY vis applications. In *Tutorial at the IEEE Visualization Conf.*

[72] Giering, R. and Kaminski, T. (2002). Recomputations in reverse mode AD. In *Automatic differentiation of algorithms: from simulation to optimization*, pages 283–291. Springer-Verlag New York, Inc., New York, NY, USA.

[73] Gold, R. (1967). Optimal binary sequences for spread spectrum multiplexing. *IEEE Trans. Inform. Theory*, 13(4):619–621.

[74] Goldstein, R. A. and Nagel, R. (1971). 3-d visual simulation. *Simulation*, 1:25–31.

[75] Golomb, S. W. (1981). *Shift Register Sequences*. Aegean Park Press, Laguna Hills, CA, USA.

[76] Gong, R. H., Abolmaesumi, P., and Stewart, J. (2006a). A Robust Technique for 2D-3D Registration. In *Engineering in Medicine and Biology Society*, pages 1433–1436.

[77] Gong, R. H., Stewart, A. J., and Abolmaesumi, P. (2006b). A new method for ct to fluoroscope registration based on unscented kalman filter. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2006*, pages 891–898.

[78] Gonzalez, E. and Schoephoerster, R. (1996). A simulation of three-dimensional systolic flow dynamics in a spherical ventricle: effects of abnormal wall motion. *Annals of Biomedical Engineering*, 24:48–57.

[79] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The lumigraph. In *Computer graphics and interactive techniques*, SIGGRAPH'96, pages 43–54.

[80] Greene (1996). Hierarchical polygon tiling with coverage masks. In *ACM SIGGRAPH Computer Graphics*, pages 65–74.

[81] Griewank, A. (1989). On Automatic Differentiation. In Iri, M. and Tanabe, K., editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers.

[82] Griewank, A. (1991). The chain rule revisited in scientific computing. *SINEWS: SIAM News*, 24(4):8–24.

[83] Griewank, A. (2000). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[84] Griewank, A., Juedes, D., and Utke, J. (1996). ADOL–C, A package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, 22(2):131–167.

[85] Grimm, S., Bruckner, S., Kanitsar, A., and Gröller, M. E. (2004). Flexible direct multi-volume rendering in interactive scenes. In *Proceedings of Vision, Modeling, and Visualization*, pages 386–379.

[86] Grossman, J. P. and Dally, W. J. (1998). Point sample rendering. In *Rendering Techniques 98*, pages 181–192.

[87] Guthe, S. and Strasser, W. (2004). Advanced techniques for high quality multiresolution volume rendering. In *In Computers & Graphics (2004*, pages 51–58. Elsevier Science.

[88] Hadwiger, M., Kniss, J. M., Rezk-salama, C., Weiskopf, D., and Engel, K. (2006a). *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA.

[89] Hadwiger, M., Kratz, A., Sigg, C., and Bühler, K. (2006b). GPU-accelerated deep shadow maps for direct volume rendering. In Harris, M., Meissner, M., Olano, M., and Slusallek, P., editors, *Graphics Hardware*, pages 49–52, Vienna, Austria. Eurographics Association.

[90] Häme, Y. (2008). Liver tumor segmentation using implicit surface evolution. *The Midas Journal*.

[91] Hansen, C., Wieferich, J., Ritter, F., Rieder, C., and Peitgen, H.-O. (2010). Illustrative visualization of 3d planning models for augmented reality in liver surgery. *Int. Journal of Computer Assisted Radiology and Surgery*, 5:133–141.

[92] Hartley, R. and Zisserman, A. (2004a). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

[93] Hartley, R. and Zisserman, A. (2004b). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

[94] Hascoët, L., Greborio, R.-M., and Pascual, V. (2005). Computing adjoints by automatic differentiation with TAPENADE. In Sportisse, B. and Dimet, F.-X. L., editors, *École INRIA-CEA-EDF "Problèmes non-linéaires appliqués"*. Springer.

[95] Hauswiesner, S., Kalkofen, D., and Schmalstieg, D. (2010). Multi-frame rate volume rendering. In *EGPGV'10*.

[96] Heikkila, J. and Silvén, O. (1997). A four-step camera calibration procedure with implicit image correction. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1106, Washington, DC, USA. IEEE Computer Society.

[97] Henyey, L. and Greenstein, J. (1941). Diffuse radiation in the galaxy. *Astrophysics Journal*, 93:70–83.

[98] Hildebrand, P., Leibecke, T., Kleemann, M., Mirow, L., Birth, M., Bruch, H., and Burk, C. (2006). Influence of operator experience in radiofrequency ablation of malignant liver tumours on treatment outcome. *European Journal of Surgical Oncology*, 32(4):430–434.

[99] Hogan, J. and Simmons, G. (2008). Standards for clearance of 510(k) premarket notifications in the us. RAJ Devices, Informa UK Ltd.

[100] Hou, Q., Zhou, K., and Guo, B. (2008). BSGP: bulk-synchronous GPU programming. *ACM Transactions on Graphics (TOG)*, 27(3):19.

[101] Huang, W. and He, K. (2009). A new heuristic algorithm for cuboids packing with no orientation constraints. *Computers & Operations Research*, 36(2):425–432.

[102] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20:1254–1259.

[103] Jain, A. and Fichtinger, G. (2006). C-arm tracking and reconstruction without an external tracker. In Larsen, R., Nielsen, M., and Sporring, J., editors, *Proceedings Medical Image Computing and Computer-Assisted Intervention - MICCAI, Part I*, volume 4190 of *Lecture Notes in Computer Science*, pages 494–502.

[104] Jeschke, S., Wimmer, M., Schumann, H., and Purgathofer, W. (2005). Automatic impostor placement for guaranteed frame rates and low memory requirements. In *ACM SIGGRAPH I3D*, pages 103–110.

[105] Kainz, B., Grabner, M., Bornik, A., Hauswiesner, S., Muehl, J., and Schmalstieg, D. (2009). Ray casting of multiple volumetric datasets with polyhedral boundaries on manycore gpus. *ACM Trans. Graph.*, 28(5):Article No. 152.

[106] Kainz, B., Khlebnikov, R., Bornik, A., and Schmalstieg, D. (2010). Multivariate beam ray obstacle visualization for brain tumor resection. IEEE Visualization Contest.

[107] Kajiya, J. T. (1986). The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150.

[108] Khamene, A., Bloch, P., Wein, W., Svatos, M., and Sauer, F. (2006a). Automatic registration of portal images and volumetric ct for patient positioning in radiation therapy. *Medical Image Analysis*, 10:96–112.

[109] Khamene, A., Chisu, R., Wein, W., Navab, N., and Sauer, F. (2006b). A Novel Projection Based Approach for Medical Image Registration. In *WBIR*, pages 247–256.

[110] Köhn, A., Drexl, J., Ritter, F., König, M., and Peitgen, H. O. (2006). GPU Accelerated Image Registration in Two and Three Dimensions. In *Bildverarbeitung für die Medizin*, pages 261–265. Springer.

[111] Kühnapfel, U., Çakmak, H. K., and Maaß, H. (2000). Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics*, 24(5):671 – 682.

[112] Khronos OpenCL Working Group (2008). *The OpenCL Specification, version 1.0.29.*

[113] Kniss, J., Kindlmann, G., and Hansen, C. (2001). Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *VIS '01*, pages 255–262.

[114] Kniss, J., Premoze, S., Ikits, M., Lefohn, A., Hansen, C., and Praun, E. (2003). Gaussian transfer functions for multi-field volume visualization. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 65. IEEE Computer Society.

[115] Kraus, M. and Ertl, T. (2001). Topology-Guided Downsampling. In *VG 2001*, Springer Computer Science, pages 223–234.

[116] Krüger, J. and Westermann, R. (2003). Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings of IEEE Visualization 2003*, pages 287–292.

[117] Krüger, J. and Westermann, R. (2003). Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization*, pages 287–292.

[118] Kubias, A., Deinzer, F., Feldmann, T., and Paulus, D. (2007a). Extended Global Optimization Strategy for Rigid 2D/3D Image Registration. In *Computer Analysis of Images and Patterns*, pages 759–767.

[119] Kubias, A., Deinzer, F., Feldmann, T., Paulus, S., Paulus, D., Schreiber, B., and Brunner, T. (2007b). 2D/3D Image Registration on the GPU. In *Proceedings of the OGRW*.

[120] La Mar, E. C., Hamann, B., and Joy, K. I. (1999). Multiresolution techniques for interactive texture-based volume visualization. In *VIS'99*.

[121] LaRose, D. (2001). *Iterative X-ray/CT Registration Using Accelerated Volume Rendering*. PhD thesis, Carnegie Mellon University.

[122] Le Bihan, Denis, Mangin, J.-F., Poupon, Cyril, Clark, Chris A., Pappata, Sabina, Molko, Nicolas, and Chabriat, Hughes (2001). Diffusion tensor imaging: Concepts and applications. *Journal of Magnetic Resonance Imaging*, 13(4):534–546.

[123] Ledergerber, C., Guennebaud, G., Meyer, M., Baecher, M., and Pfister, H. (2008). Volume MLS ray casting. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1372–1379.

[124] Lee, M. E., Redner, R. A., and Uselton, S. P. (1985). Statistically optimized sampling for distributed ray tracing. *SIGGRAPH CG*, 19:61–68.

[125] Lell, M., Anders, K., Klotz, E., Ditt, H., Bautz, W., and Tomandl, B. (2006). Clinical evaluation of bone-subtraction ct angiography (bscta) in head and neck imaging. *European Radiology*, 16:889–897. 10.1007/s0330-005-0032-1.

[126] Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics and Applications*.

[127] Levoy, M. (1990a). Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9:245–261.

[128] Levoy, M. (1990b). Volume rendering by adaptive refinement. *The Visual Computer*, 6(1):2–7.

[129] Li, W., Mueller, K., and Kaufman, A. (2003). Empty space skipping and occlusion clipping for texture-based volume rendering. In *Proceedings of IEEE Visualization*, pages 317–324.

[130] Lindholm, S., Ljung, P., Hadwiger, M., and Ynnerman, A. (2009). Fused multi-volume dvr using binary space partitioning. *Comput. Graph. Forum*, 28(3):847–854.

[131] Liu, B., Wei, L.-Y., and Xu, Y.-Q. (2006). Multi-layer depth peeling via fragment sort. Technical Report MSR-TR-2006-81, Microsoft Research Asia.

[132] Livyatan, H., Yaniv, Z., and Joskowicz, L. (2003). Gradient-Based 2D/3D Rigid Registration of Fluoroscopic X-ray to CT. *IEEE Transactions on Medical Imaging*, 22(11):1395–1406.

[133] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algoritm. *Computer graphics*, 21(4):163–168.

[134] Luebke, D. and Erikson, C. (1997). View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH '97*, pages 199–208.

[135] Lundström, C., Ljung, P., and Ynnerman, A. (2006a). Local histograms for design of transfer functions in direct volume rendering. In *IEEE Transactions on Visualization and Computer Graphics*, volume 12, pages 1570 – 1579, Piscataway, NJ, USA. IEEE Educational Activities Department.

[136] Lundström, C., Ynnerman, A., Ljung, P., Persson, A., and Knutsson, H. (2006b). The alpha-histogram: Using spatial coherence to enhance histograms and transfer function design. In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization 2006*, Lisbon, Portugal.

[137] Maes, F., Vandermeulen, D., and Suetens, P. (1999). Comparative Evaluation of Multiresolution Optimization Strategies for Multimodality Image Registration by Maximization of Mutual Information. *Medical Image Analysis*, 3(4):373–386.

[138] Maintz, J. B. A. and Viergever, M. A. (1998a). A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36.

[139] Maintz, J. B. A. and Viergever, M. A. (1998b). A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36.

[140] Manenti, G., Carlani, M., Mancino, S., Colangelo, V., Di Roma, M., Squillaci, E., and Simonetti, G. (2007). Diffusion tensor magnetic resonance imaging of prostate cancer. *Invest Radiol*, 42(6):412–9.

[141] MARCHESIN, S., MONGENET, C., and DISCHLER, J.-M., editors (2006). *Dynamic load balancing for parallel volume rendering*, EGPGV'06: symposium on parallel graphics and visualization, Braga, Portugal. Eurographics.

[142] Marin, T., Wernick, M. N., Yang, Y., and Brankov, J. G. (2010). Motion-compensated reconstruction of gated cardiac spect images using a deformable mesh model. In *IEEE Int. Conf. on biomedical imaging: from nano to macro*, pages 520–523. IEEE Press.

[143] Mark, W. R., Glanville, R. S., Akeley, K., and Kilgard, M. J. (2003a). Cg: a system for programming graphics hardware in a C-like language. In Hodgins, J., editor,

SIGGRAPH 2003 Conference Proceedings, Annual Conference Series, pages 896–907, New York, NY, USA. ACM SIGGRAPH, ACM Press.

[144] Mark, W. R., Glanville, R. S., Akeley, K., and Kilgard, M. J. (2003b). Cg: a system for programming graphics hardware in a C-like language. In Hodgins, J., editor, *SIGGRAPH 2003 Conference Proceedings*, pages 896–907. ACM SIGGRAPH, ACM Press.

[145] Mark, W. R. and Proudfoot, K. (2001). The F-buffer: a rasterization-order FIFO buffer for multi-pass rendering. In *Proceedings of SIGGRAPH/EUROGRAPHICS workshop on graphics hardware*, pages 57–64.

[146] Marroquim, R., Kraus, M., and Cavalcanti, P. R. (2008). Special section: Point-based graphics: Efficient image reconstruction for point-based and line-based rendering. *SIGGRAPH CG*, 32:189–203.

[147] Marsalek, L., Hauber, A., and Slusallek, P. (2008). High-speed volume ray casting with CUDA. In *Proceedings of IEEE Symposium on Interactive Ray Tracing*, page 185.

[148] Marsh, A., Simistira, F., and Robb, R. (1998). Vr in medicine: Virtual colonoscopy. *Future Generation Computer Systems*, 14(3-4):253 – 264. Virtual Reality in Industy and Research.

[149] Matsui, M., Ino, F., and Hagihara, K. (2005). Parallel volume rendering with early ray termination for visualizing large-scale datasets. In Cao, J., Yang, L., Guo, M., and Lau, F., editors, *Parallel and Distributed Processing and Applications*, volume 3358 of *Lecture Notes in Computer Science*, pages 245–256. Springer Berlin Heidelberg.

[150] Max, N. (1995). Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1:99–108.

[151] Max, N. L. (1986). Atmospheric illumination and shadows. In *ACM SIGGRAPH Computer Graphics*, pages 117–124. ACM.

[152] Maximo, A., Ribeiro, S., Bentes, C., Oliveira, A., and Farias, R. (2008). Memory Efficient GPU-Based Ray Casting for Unstructured Volume Rendering. pages 155–162, Los Angeles, California, USA. Eurographics Association.

[153] McCormick, B. H. (1988). Visualization in scientific computing. *SIGBIO Newsl.*, 10:15–21.

[154] McDonald, R. J., Gray, L. A., Cloft, H. J., Thielen, K. R., and Kallmes, D. F. (2009). The Effect of Operator Variability and Experience in Vertebroplasty Outcomes. *Radiology*, 253(2):478–485.

[155] Meißner, M., Huang, J., Bartz, D., Mueller, K., and Crawfis, R. (2000). A practical evaluation of popular volume rendering algorithms. In *IEEE Symp. on Volume visualization*, pages 81–90. ACM.

[156] Messenger, J. C., Chen, S. J., Carroll, J. D., Burchenal, J., Kioussopoulos, K., and Groves, B. M. (2000). 3d coronary reconstruction from routine single-plane coronary angiograms: Clinical validation and quantitative analysis of the right coronary artery in 100 patients. *The International Journal of Cardiac Imaging*, 16:413–427. 10.1023/A:1010643426720.

[157] Mitchell, D. P. (1987). Generating antialiased images at low sampling densities. *SIGGRAPH CG*, 21:65–72.

[158] Müller, C., Strengert, M., and Ertl, T. (2006). Optimized Volume Raycasting for Graphics-Hardware-based Cluster Systems. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV06)*, pages 59–66. Eurographics Association.

[159] Molnar, S., Cox, M., Ellsworth, D., and Fuchs, H. (1994). A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32.

[160] Moloney, B., Weiskopf, D., Möller, T., and Strengert, M. (2007). Scalable Sort-First Parallel Direct Volume Rendering with Dynamic Load Balancing. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV07)*, pages 45–52. Eurographics Association.

[161] Morris, L., Delassus, P., Callanan, A., Walsh, M., Wallis, F., Grace, P., and McGloughlin, T. (2005). 3-d numerical simulation of blood flow through models of the human aorta. *Journal of Biomechanical Engineering*, 127:767–775.

[162] Moseley, M. E., Cohen, Y., Kucharczyk, J., Mintorovitch, J., Asgari, H. S., Wendland, M. F., Tsuruda, J., and Norman, D. (1990). Diffusion-weighted MR imaging of anisotropic water diffusion in cat central nervous system. *Radiology*, 176(2):439–445.

[163] Muchitsch, M. (2010). Automatically generated transfer function galleries for high dimensional multi variate data. Master's thesis, Graz University of Technology.

[164] Muehl, J., Kainz, B., Portugaller, H., Stiegler, P., and Bauer, C. (2009). Computer oriented image acquisition of the liver: Toward a better numerical model for radiofrequency ablation. *Conf Proc IEEE Eng Med Biol Soc*, 1.

[165] Mueller, C., Hodgson, J. M., Brutsche, M., Bestehorn, H.-P., Marsch, S., Perruchoud, A. P., Roskamm, H., and Buettner, H. J. (2003). Operator experience and long term outcome after percutaneous coronary intervention. *Can J Cardiol*, 19:1047–51.

[166] Müller, C., Strengert, M., and Ertl, T. (2007). Adaptive load balancing for raycasting of non-uniformly bricked volumes. *Parallel Comput.*, 33(6):406–419.

[167] Nadeau, D. R. (2000). Volume scene graphs. In *Proceedings of IEEE symposium on volume visualization*, pages 49–56.

[168] Napel, S., Marks, M. P., Rubin, G. D., Dake, M. D., McDonnell, C. H., Song, S. M., Enzmann, D. R., and Jeffrey, R. B. (1992). CT angiography with spiral CT and maximum intensity projection. *Radiology*, 185(2):607–610.

[169] Natterer, F. and Ritman, E. (2002). Past and future directions in x-ray computed tomography (ct). 12(4):175–187.

[170] Naumann, U. (2002a). *Automatic Differentiation of Algorithms: From Simulation to Optimization*. Springer.

[171] Naumann, U. (2002b). Reducing the memory requirement in reverse mode automatic differentiation by solving tbr flow equations. *Lecture Notes in Computer Science*, 2330/2002:1039.

[172] Navab, N., Bani-Hashemi, A., Mitschke, M., Fox, A., Holdsworth, D., Fahrig, R., and Graumann, R. (1996). Dynamic geometrical calibration for 3-d cerebral angiography. *SPIE Medical Imaging*, 2708 / 361:361–370.

[173] Navkar, N. V., Tsekos, N. V., Stafford, J. R., Weinberg, J. S., and Deng, Z. (2010). Visualization and planning of neurosurgical interventions with straight access. In *Proceedings of the First international conference on Information processing in computer-assisted interventions*, IPCAI'10, pages 1–11. Springer.

[174] Nerem, R., Jr., J. R., Gross, D., Hamlin, R., and Geiger, G. (1974). Hot-film anemometer velocity measurements of arterial blood flow in horses. *Circulation Research*, 34:193–203.

[175] Nickolls, J., Buck, I., and Garland, M. (2008). Scalable parallel programming with CUDA. *ACM Queue*, 6(2):40–53.

[176] Nirenstein, S., Blake, E., and Gain, J. (2002). Exact from-region visibility culling. In *Eurographics Workshop on Rendering*, EGRW, pages 191–202. Eurographics Association. ACM ID: 581921.

[177] Nishita, T., Miyawaki, Y., and Nakamae, E. (1987). A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 303–310. ACM.

[178] Notkin, I. and Gotsman, C. (1997). Parallel progressive ray-tracing. *SIGGRAPH CG*, 16(1):43–55.

[179] NVIDIA (2008). *NVIDIA CUDA Programming Guide 2.0*. NVIDIA Corporation.

[180] NVIDIA (2009). Nvidia's next generation cuda compute architecture: Fermi. White paper. Available online.

[181] Painter, J. and Sloan, K. (1989). Antialiased ray tracing by adaptive progressive refinement. *SIGGRAPH CG*, 23:281–288.

[182] Parker, S., Martin, W., pike J. Sloan, P., Shirley, P., Smits, B., and Hansen, C. (1999). Interactive ray tracing. In *Interactive 3D Graphics*, pages 119–126.

[183] Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., and Stich, M. (2010a). Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*.

[184] Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., and Stich, M. (2010b). Optix: A general purpose ray tracing engine. *ACM Trans. Graph.*

[185] Patidar, S. and Narayanan, P. J. (2008). Ray casting deformable models on the GPU. In *6th Indian Conference on Computer Vision, Graphics & Image Processing*, pages 481–488.

[186] Pfister, H., Zwicker, M., van Baar, J., and Gross, M. (2000). Surfels: surface elements as rendering primitives. In *Computer graphics and interactive techniques*, SIGGRAPH'00, pages 335–342.

[187] Pharr, M., editor (2005). *GPU Gems 2*. Addison Wesley.

[188] Plate, J., Holtkaemper, T., and Froehlich, B. (2007). A flexible multi-volume shader framework for arbitrarily intersecting multi-resolution datasets. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1584–1591.

[189] Pock, T., Grabner, M., and Bischof, H. (2007a). Real-time computation of variational methods on graphics hardware. In *12th Computer Vision Winter Workshop (CVWW 2007)*, St. Lamprecht.

[190] Pock, T., Pock, M., and Bischof, H. (2007b). Algorithmic differentiation: Application to variational problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1180–1193.

[191] Pock, T., Pock, M., and Bischof, H. (2007c). Algorithmic differentiation: Application to variational problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1180–1193.

[192] Pomi, A. and Slusallek, P. (2005). Interactive Ray Tracing for Virtual TV Studio Applications. *J. Virtual Reality and Broadcasting*, 2(1).

[193] Popper, K. R. (1934). *The Logic of Scientific Discovery.* Hutchinson, London.

[194] Reiter, G., Reiter, U., Kovacs, G., Kainz, B., Schmidt, K., Maier, R., Olschewski, H., and Rienmueller, R. (2008a). Magnetic Resonance-Derived 3-Dimensional Blood Flow Patterns in the Main Pulmonary Artery as a Marker of Pulmonary Hypertension and a Measure of Elevated Mean Pulmonary Arterial Pressure. *Circ Cardiovasc Imaging*, 1(1):23–30.

[195] Reiter, G., Reiter, U., Kovacs, G., Kainz, B., Schmidt, K., Meier, R., Olschewski, H., and Rienmueller, R. (2008b). Three-dimensional blood flow patterns in the pulmonary outflow tract as marker of pulmonary hypertension. *to appear in Circulations, American Heart Association.*

[196] Rezk-Salama, C., Todt, S., and Kolb, A. (2008). Raycasting of light field galleries from volumetric data. *Computer Graphics Forum*, 27(3):839–846.

[197] Rieder, C., Ritter, F., Raspe, M., and Peitgen, H.-O. (2008). Interactive visualization of multimodal volume data for neurosurgical tumor treatment. *Comput. Graph. Forum*, 27(3):1055–1062.

[198] Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D., and Van Dantzich, M. (1998). Data mountain: Using spatial memory for document management.

[199] Roessler, F., Botchen, R. P., and Ertl, T. (2008). Dynamic shader generation for GPU-based multi-volume ray casting. *IEEE Computer Graphics and Appllications*, 28(5):66–77.

[200] Roettger, S., Guthe, S., Weiskopf, D., and Ertl, T. (2003). Smart Hardware-Accelerated Volume Rendering. In *Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '03*, pages 231–238.

[201] Ropinski, T., Döring, C., and Rezk-Salama, C. (2010). Interactive volumetric lighting simulating scattering and shadowing. In *IEEE Pacific Visualization Symposium (PacificVis 2010)*, pages 169–176.

[202] Rovaris, M., Gass, A., Bammer, R., Hickman, S. J., Ciccarelli, O., Miller, D. H., and Filippi, M. (2005). Diffusion mri in multiple sclerosis. *Neurology*, 65(10):1526–32.

[203] Rubin, G. D., Dake, M. D., Napel, S. A., McDonnell, C. H., and Jeffrey, R. B. (1993). Three-dimensional spiral CT angiography of the abdomen: initial clinical experience. *Radiology*, 186(1):147–152.

[204] Schroeder, W., Martin, K. M., and Lorensen, W. E. (1998). *The visualization toolkit (2nd ed.): an object-oriented approach to 3D graphics.* Prentice-Hall, Inc.

[205] Schumann, C., Bieberstein, J., Trumm, C., Schmidt, D., Bruners, P., Niethammer, M., Hoffmann, R. T., Mahnken, A. H., Pereira, P. L., and Peitgen, H.-O. (2010). Fast automatic path proposal computation for hepatic needle placement. volume 7625 of *Proceedings of the SPIE*, pages 76251J–1 – 76251J–10.

[206] Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., and Hanrahan, P. (2008). Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics (TOG)*, 27(3):18.

[207] Shamir, R. R., Tamir, I., Dabool, E., Joskowicz, L., and Shoshan, Y. (2010). A method for planning safe trajectories in image-guided keyhole neurosurgery. In *Proceedings of the 13th international conference on Medical image computing and computer-assisted intervention: Part III*, MICCAI'10, pages 457–464. Springer.

[208] Shareef, N., Lee, T.-Y., Shen, H.-W., and Mueller, K. (2006). An image-based modelling approach to gpu-based rendering of unstructured grids. In *Volume Graphics*, pages 31–38.

[209] Smit, F. A., van Liere, R., Beck, S., and Fröhlich, B. (2009). An image-warping architecture for VR: Low latency versus image quality. In *VR*, pages 27–34. IEEE.

[210] Smit, F. A., van Liere, R., and Fröhlich, B. (2007). The design and implementation of a VR-architecture for smooth motion. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 153–156, New York, NY, USA. ACM.

[211] Smit, F. A., van Liere, R., and Fröhlich, B. (2008). An image-warping VR-architecture: design, implementation and applications. In *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 115–122, New York, NY, USA. ACM.

[212] Soimu, D., Badea, C., and Pallikarakis, N. (2003). A novel approach for distortion correction for x-ray image intensifiers. *Computerized Medical Imaging and Graphics*, 27(7):79–85.

[213] Song, W., Hua, S., Ou, Z., An, H., and Song, K. (2008). Octree based representation and volume rendering of three-dimensional medical data sets. In *Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics - Volume 01*, pages 316–320, Washington, DC, USA. IEEE Computer Society.

[214] Springer, J. P., Beck, S., Weiszig, F., Reiners, D., and Froehlich, B. (2007). Multi-frame rate rendering and display. In Sherman, W. R., Lin, M., and Steed, A., editors, *VR*, pages 195–202. IEEE Computer Society.

[215] Springer, J. P., Lux, C., Reiners, D., and Froehlich, B. (2008). Advanced multi-frame rate rendering techniques. In *VR*, pages 177–184. IEEE.

[216] Stegmaier, S., Strengert, M., Klein, T., and Ertl, T. (2005). A simple and flexible volume rendering framework for graphics-hardware-based raycasting. *Int. Workshop on Volume Graphics*, 0:187–241.

[217] Sugerman, J., Fatahalian, K., Boulos, S., Akeley, K., and Hanrahan, P. (2009). Gramps: A programming model for graphics pipelines. *ACM Trans. Graph.*, 28(1):1–11.

[218] Sun, W. and Cooperstock, R. (2006). An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques. *Machine Vision and Applications*, 17(1):51 – 67.

[219] Tandy, C. R. V. (1967). The Isovist Method of Landscape Survey. *Methods of Landscape Analysis*, pages 9–10.

[220] Taubin, G. (1995). A signal processing approach to fair surface design. In *SIGGRAPH '95*, pages 351–358.

[221] Termeer, M., Bescós, J. O., and Telea, A. (2006). Preserving sharp edges with volume clipping. In *Proceedings of Vision, Modeling and Visualization*, pages 341–348.

[222] Tomaževič, D., Likar, B., and Pernuš, F. (2006). 3-D/2-D registration by integrating 2-D information in 3-D. *IEEE Transactions on Medical Imaging*, 25(1):17–27.

[223] Torrey, H. C. (1956). Bloch Equations with Diffusion Terms. *Physical Review*, 104:563–565.

[224] Tummala, R. P., Chu, R. M., Liu, H., Truwit, C. L., and Hall, W. A. (2003). Application of diffusion tensor imaging to magnetic-resonance-guided brain tumor resection. *Pediatr Neurosurg*, 39(1):39–43.

[225] Udupa, J. K. and Herman, G. T., editors (1991). *3D imaging in medicine*. CRC Press, Inc., Boca Raton, FL, USA.

[226] Urschler, M., Werlberger, M., Scheurer, E., and Bischof, H. (2010). Robust optical flow based deformable registration of thoracic ct images. In *MICCAI Workshop Medical Image Analysis in the Clinic: A Grand Challenge*, LNCS. Springer.

[227] Vaillant, M., Davatzikos, C., Taylor, R., and Bryan, R. (1997). A path-planning algorithm for image-guided neurosurgery. In *Joint Conf. Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, volume 1205 of *LNCS*, pages 467–476. Springer.

[228] Van de Kraats, E., Penney, G., Tomazevic, D., Van Walsum, T., and Niessen, W. (2004). Standardized Evaluation of 2D-3D Registration. In *Proceedings of the MICCAI*, pages 574–581.

[229] van de Kraats, E. B., Penney, G. P., Tomazevic, D., van Walsum, T., and Niessen, W. J. (2004). Standardized evaluation of 2D-3D registration. In Barillot, C., Haynor, D. R., and Hellier, P., editors, *Proceedings Medical Image Computing and Computer-Assisted Intervention–MICCAI*, volume 3216 of *Lecture Notes in Computer Science*, pages 574–581. Springer.

[230] van de Kraats, E. B., Penney, G. P., van Walsum, T., and Niessen, W. J. (2005). Multispectral MR to X-Ray Registration of Vertebral Bodies by Generating CT-Like Data. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2005*, pages 911–918.

[231] van Rossum, A. B., Pattynama, P. M., Ton, E. R., Treurniet, F. E., Arndt, J. W., van Eck, B., and Kieft, G. J. (1996). Pulmonary embolism: validation of spiral CT angiography in 149 patients. *Radiology*, 201(2):467–470.

[232] Vancamberg, L., Sahbani, A., Muller, S., and Morel, G. (2010). Needle path planning for digital breast tomosynthesis biopsy. In *Robotics and Automation (ICRA)*, pages 2062 –2067.

[233] Viard, R., Betrouni, N., Rousseau, J., Mordon, S., Ernst, O., and Maouche, S. (2007). Needle positioning in interventional mri procedure: real time optical localisation and accordance with the roadmap. In *Engineering in Medicine and Biology Society (EMBS)*, pages 2748–2751.

[234] Villard, C., Soler, L., and Gangi, A. (2005). Radiofrequency ablation of hepatic tumors: simulation, planning, and contribution of virtual reality and haptics. *Comput. Methods Biomech Biomed. Engin.*, 8(4):215–227.

[235] Wald, I., Benthin, C., and Slusallek, P. (2002). OpenRT – A Flexible and Scalable Rendering Engine for Interactive 3D Graphics. Technical report, CG Group, Saarland University.

[236] Wald, I., Friedrich, H., Marmitt, G., Slusallek, P., and Seidel, H.-P. (2005). Faster isosurface ray tracing using implicit kd-trees. *IEEE Vis and CG*, 11:562 –572.

[237] Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., Parker, S. G., and Shirley, P. (2007). State of the art in ray tracing animated scenes. In *STAR Proceedings of EG 2007*, pages 89–116.

[238] Wang, Q. and JaJa, J. (2008). Interactive high-resolution isosurface ray casting on multicore processors. *IEEE Vis and CG*, 14(3):603 –614.

[239] Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117.

[240] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Image Processing*, 13(4):600–612.

[241] Weiler, M., Westermann, R., Hansen, C., Zimmerman, K., and Ertl, T. (2000a). Level–of–detail volume rendering via 3D textures.

[242] Weiler, M., Westermann, R., Hansen, C., Zimmerman, K., and Ertl, T. (2000b). Level-of-detail volume rendering via 3d textures. pages 7–13.

[243] Wein, W. (2003). Intensity Based Rigid 2D-3D Registration Algorithms for Radiation Therapy. Master's thesis, Technische Universität München, Fakultät für Informatik.

[244] Wein, W., Röper, B., and Navab, N. (2005a). 2D/3D registration based on volume gradients. In *SPIE Medical Imaging 2005, San Diego*.

[245] Wein, W., Röper, B., and Navab, N. (2005b). 2D/3D Registration Based on Volume Gradients. In *Proceedings of SPIE: Medical Imaging 2005: Image Processing*, volume 5747.

[246] Weiskopf, D. (2006). *GPU-Based Interactive Visualization Techniques (Mathematics and Visualization)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

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

[248] Weiskopf, D., Weiler, M., and Ertl, T. (2004). Maintaining constant frame rates in 3d texture-based volume rendering. In *Proceedings of the Computer Graphics International*, pages 604–607, Washington, DC, USA. IEEE Computer Society.

[249] Westermann, R. and Sevenich, B. (2001). Accelerated volume ray-casting using texture mapping. In *Proceedings of the conference on Visualization '01*, VIS '01, pages 271–278, Washington, DC, USA. IEEE Computer Society.

[250] Wonka, P., Wimmer, M., Zhou, K., Maierhofer, S., Hesina, G., and Reshetov, A. (2006). Guided visibility sampling. In *ACM Transactions on Graphics*, volume 25, pages 494–502. ACM.

[251] Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. ICCV*.

[252] Zheng, G. (2007). Unifying Energy Minimization and Mutual Information Maximization for Robust 2D/3D Registration of X-Ray and CT Images. In *LNCS: Pattern Recognition*, volume 4713, pages 547–557. Springer.

[253] Zhou, K., Hou, Q., Ren, Z., Gong, M., Sun, X., and Guo, B. (2009). Renderants: interactive reyes rendering on gpus. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–11, New York, NY, USA. ACM.

[254] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560.

[255] Zitova, B. and Flusser, J. (2003). Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000.

Appendix A

Online survey to evaluate the observations stated in Chapter 1

A.1 Questionnaire

Informal survey on the usefulness of 3D visualization in the clinical practice

It would be important for my dissertation if you could kindly provide your input through this survey. This survey is informal, which means that its purpose is to measure the degree of satisfaction of medical personell with current state-of-the-art visualization methods for clinical volumetric data. The survey will not take longer than 10 minutes.
Please feel free to email me if you do not understand a question and do not hesitate to ask also your medical colleagues to participate.

Thank you for participating in this informal survey.

There are 31 questions in this survey

General

In this section you will be asked some general questions about your actual profession.

1 [Profession] Which is your main field of medical expertise?

*

Please choose **only one** of the following:

○ Radiology

○ Surgery

○ Internal medicine

○ Physicist

○ General practitioner

○ Medical student

○ Visualization expert

○ Biomedical engineer

○ Other

2 [Experiense]How many years have you been working in this field?

*

Please choose **only one** of the following:

○ < 1 year

○ 2-5 years

○ 5-10 years

○ > 10 years

3 [Modalities] With which medical imaging modalities are you familiar?

*

Please choose **all** that apply:

□ CT

□ 4D-CT

□ MRI (standard, e.g. T1, T2...)

□ Ultrasound

□ MRI (advanced sequences, e.g. fMRI, DTI, 4D...)

□ SPECT

□ PET

□ Scintigraphy

□ Thermography (e.g. mamma)

□ X-ray (C-Arm, film...)

□ Other:

2D vs. 3D representation

In this section you will be asked for your personal opinion and experience with medical volumetric data visualization.

4 [Overall]3D image synthesis from medical volumetric data is most important for:

*

Please choose **only one** of the following:

○ Interdisciplinary communication and interdisciplinary intervention planning

○ Radiology and radiologic treatment planning

○ Surgery and intervention planning

○ Internal medicine

○ Physicists

○ General practitioners and patient communication/explanation

○ Medical students and teaching

○ Biomedical engineers

○ Other

5 [Preference]Do you personally prefer scrolling through 2D slices instead of interacting with a 3D representation of the volumetric data set?

*

Please choose **only one** of the following:

○ Always

○ It depends on the application

○ Never

6 [plausibility] Are you able to interpret high dimensional and derived datasets (PC-MRI, DTI..) from 2D slices? *

Only answer this question if the following conditions are met:
° Answer was 'Always' at question '5 [Preference]' (Do you personally prefer scrolling through 2D slices instead of interacting with a 3D representation of the volumetric data set?)

Please choose **only one** of the following:

○ Yes

○ No

7 [Whythat] For which kinds of applications do you prefer 2D slice views? *

Only answer this question if the following conditions are met:

° Answer was 'It depends on the application' at question '5 [Preference]' (Do you personally prefer scrolling through 2D slices instead of interacting with a 3D representation of the volumetric data set?)

Please choose all that apply and provide a comment:

□ general diagnosis

□ millimeter accurate planning of radiological interventions

□ to get an overview over the whole scan

□ to communicate certain conditions to colleques from the same field of expertise

□ to communicate certain conditions to colleques from other fields of expertise

□ planning of surgery interventions

□ investigation of functional imaging (e.g. DTI, PC-MRI..)

□ surgery navigation

□ minimal invasive surgery navigation

□ angiography analysis

Other:

You are welcome add comments in the boxes at the right hand side.

8 [whythat2] For which kind of applications do you prefer 3D views? *

Only answer this question if the following conditions are met:
° Answer was 'It depends on the application' at question '5 [Preference]' (Do you personally prefer scrolling through 2D slices instead of interacting with a 3D representation of the volumetric data set?)

Please choose all that apply and provide a comment:

□ general diagnosis

□ millimeter accurate planning of radiological interventions

□ to get an overview over the whole scan

□ to communicate certain

conditions to colleques from the same field of expertise

□ to communicate certain conditions to colleques from other fields of expertise

□ planning of surgery interventions

□ investigation of functional imaging (e.g. DTI, PC-MRI..)

□ surgery navigation

□ minimal invasive surgery navigation

□ angiography analysis

Other: □

You are welcome add comments in the boxes at the right hand side.

9 [whaythatfplaus] For which kind of applications do you prefer 2D slice views? *

Only answer this question if the following conditions are met:
° Answer was 'No' at question '6 [plausibility]' (Are you able to interpret high dimensional and derived datasets (PC-MRI, DTI..) from 2D slices?)

Please choose all that apply and provide a comment:

□ General diagnosis

□ millimeter accurate planning of radiological interventions

□ to get an overview over the whole scan

□ to communicate certain conditions to colleques from the same field of expertise

□ to communicate certain conditions to colleques from other fields of expertise

□ planning of surgery interventions

□ investigation of functional imaging (e.g. DTI, PC-MRI..)

□ surgery navigation

□ minimal invasive surgery navigation

□ angiography analysis

Other:

You are welcome add comments in the boxes at the right hand side.

10 [whaythatfplaus2] For which kind of applications do you prefer 3D views? *

Only answer this question if the following conditions are met:
° Answer was 'No' at question '6 [plausibility]' (Are you able to interpret high dimensional and derived datasets (PC-MRI, DTI..) from 2D slices?)

Please choose all that apply and provide a comment:

□ general diagnosis

□ millimeter accurate planning of radiological interventions

□ to get an overview over the whole scan

□ to communicate certain conditions to colleques from the same field of expertise

□ to communicate certain conditions to colleques from other fields of expertise

□ planning of surgery interventions

□ investigation of functional imaging (e.g. DTI, PC-MRI..)

□ surgery navigation

□ minimal invasive surgery navigation

□ angiography analysis

Other:

You are welcome add comments in the boxes at the right hand side.

State-of-the-art medical volume visualization feasibility

In this section you will be asked for your personal experience with 3D image synthesis in the clinical practice.

11 [Feasible1]

Are you currently using 3D image synthesis in your diagnostic or interventional practice?
3D image synthesis describes every visualization method which goes beyond 2D slice views. Every 3D object which you can rotate in a virtual 3D space and which is based on a medical volumetric scan is valid for this question.

*

Please choose **only one** of the following:

○ Yes

○ No

12 [feasible2] In which context and for which application are you using 3D image synthesis currently?

*

Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '11 [Feasible1]' (Are you currently using 3D image synthesis in your diagnostic or interventional practice? 3D image synthesis describes every visualization method which goes beyond 2D slice views. Every 3D object which you can rotate in a virtual 3D space and which is based on a medical volumetric scan is valid for this question.)

Please write your answer here:

13 [feasible3]Are you satisfied with the*image quality*, which is provided by current 3D image synthesis algorithms? (1 means absolutely unsatisfied; 10 means absolutely satisfied. Use 5 if you have no expirence with that technique.)

*

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Direct Volume Rendering Techniques (DVR) [Example]	○	○	○	○	○	○	○	○	○	○
Segmentation approximations by geometric										

objects [Example]	○	○	○	○	○	○	○	○	○	○
Non-Photorealistic Rendering (NPR) [Example]	○	○	○	○	○	○	○	○	○	○
Photorealistic representations [Example]	○	○	○	○	○	○	○	○	○	○
Maximum Intesity Projection (MIP) [Example]	○	○	○	○	○	○	○	○	○	○

1 means absolutely unsatisfied; 10 means absolutely satisfied.

14 [feasible4]Are you satisfied with the *image generation frequency* and interactivity, which is provided by current 3D image synthesis algorithms? (1 means absolutely unsatisfied; 10 means absolutely satisfied.Use 5 if you have no expirence with that technique.)

*****

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Direct Volume Rendering Techniques (DVR) [Example]	○	○	○	○	○	○	○	○	○	○
Segmentation approximations by geometric objects [Example]	○	○	○	○	○	○	○	○	○	○
Non-Photorealistic Rendering (NPR) [Example]	○	○	○	○	○	○	○	○	○	○
Photorealistic representations [Example]	○	○	○	○	○	○	○	○	○	○
Maximum Intesity Projection (MIP) [Example]	○	○	○	○	○	○	○	○	○	○

1 means absolutely unsatisfied; 10 means absolutely satisfied

15 [feasible5]
Are you satisfied with the*image accuracy*, which is provided by current 3D image synthesis algorithms?
(1 means absolutely unsatisfied; 10 means absolutely satisfied.Use 5 if you have no expirence with that technique.)

*****

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Direct Volume Rendering Techniques (DVR) [Example]	○	○	○	○	○	○	○	○	○	○
Segmentation approximations by geometric objects [Example]	○	○	○	○	○	○	○	○	○	○
Non-Photorealistic Rendering (NPR) [Example]	○	○	○	○	○	○	○	○	○	○
Photorealistic representations [Example]	○	○	○	○	○	○	○	○	○	○
Maximum Intesity Projection (MIP) [Example]	○	○	○	○	○	○	○	○	○	○

1 means absolutely unsatisfied; 10 means absolutely satisfied.Use 5 if you have no expirence with that technique.n

16 [feasible6]

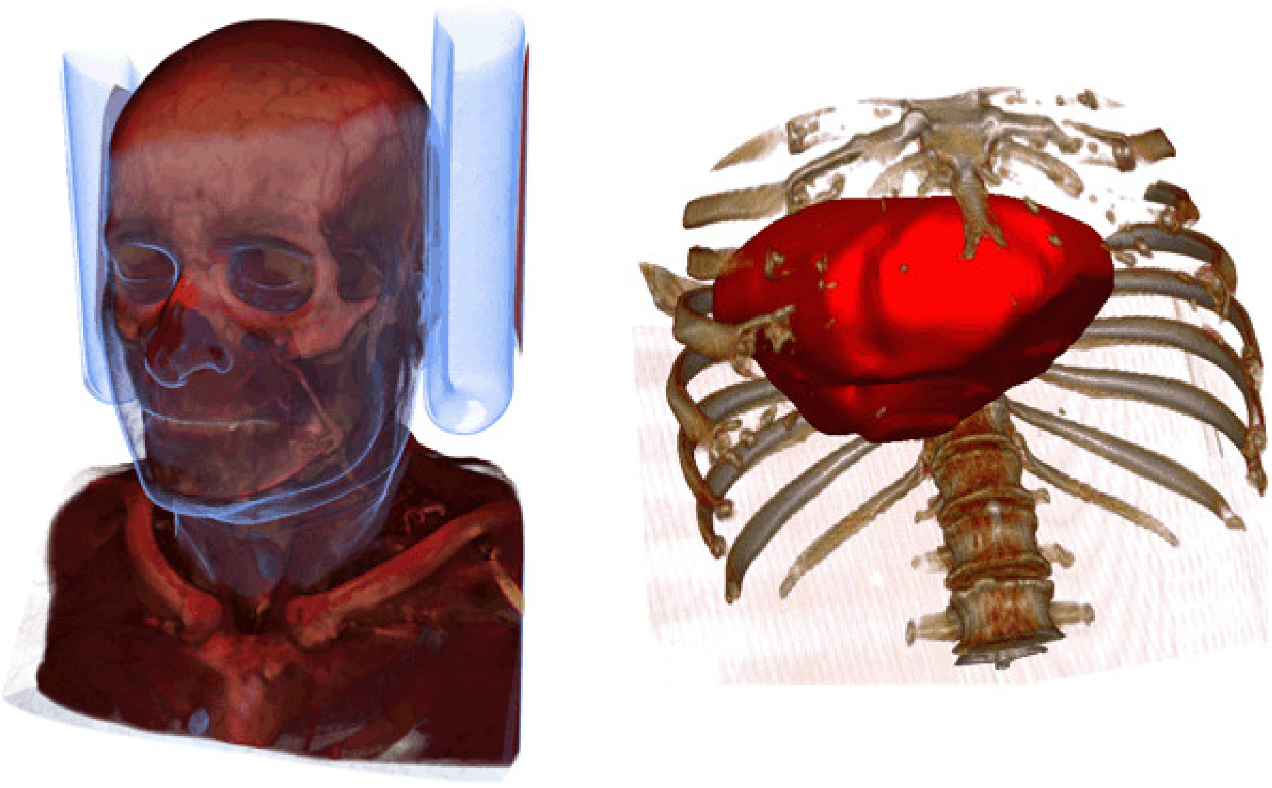
What is the most troublesome shortcoming of current direct volume rendering (DVR) techniques?

*

Please choose **only one** of the following:

- ○ image generation speed and reduced interactivity
- ○ image quality (e.g. display resolution)
- ○ missing important information
- ○ missing photorealism
- ○ image accuracy
- ○ reduced quality during interaction
- ○ Other □

Examples for DVR:



Observation rating

Please rate the following observations due to their validity.

17 [observation1]Medical 3D image synthesis algorithms must speed up the information finding process to be accepted by medical doctors. For standard diagnostic procedures, 3D representations do not provide additional information to radiologists but they are useful to illustrate pathological findings to other medical specialists who use that information for opinion making and intervention planning. *

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Validity (1...completely wrong, 10...fully true)	○	○	○	○	○	○	○	○	○	○

18 [observation2]If a 3D image synthesis algorithm is comprehensible and if it is related to a familiar physical principle, 3D image synthesis is accepted as diagnostic valuable tool and integrated into the clinical workflow. *

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Validity (1...completely wrong, 10...fully true)	○	○	○	○	○	○	○	○	○	○

19 [observation3]3D image synthesis gets crucial if the data input dimensionality exceeds normal human experience. (e.g. fMRI, DTI, PC-MRI,...) *

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Validity (1...completely wrong, 10...fully true)	○	○	○	○	○	○	○	○	○	○

20 [observation4]Sate-of-the-art 3D image synthesis algorithms are either not able to provide the necessary image quality or the necessary rendering speed or they are restricted by the amount of input data. This prevents a common use of these techniques in the clinical practice and for applications where the overall result must be available within reasonable time. *

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Validity (1...completely wrong, 10...fully true)	○	○	○	○	○	○	○	○	○	○

21 [additional comments] Do you have any additional comments on the four observations from above?

Please write your answer here:

Observation falsification

Please answer the following questions to the best to your knowlege.

22 [false1]It exists a 3D image synthesis algorithms, which does not speed up clinical communication or the information finding process, but which is overall accepted by medical doctors. *

Please choose **only one** of the following:

○ Yes

○ No

○ I definitely don't know

Make a comment on your choice here:

23 [which1] Which one? *

Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '22 [false1]' (It exists a 3D image synthesis algorithms, which does not speed up clinical communication or the information finding process, but which is overall accepted by medical doctors.)

Please write your answer here:

24 [false2]It exists a by no one comprehensible (random) 3D image synthesis algorithm which has been integrated into a clinical workflow. *

Please choose **only one** of the following:

○ Yes

○ No

○ I definitely don't know

Make a comment on your choice here:

25 [which2]Which one? *

Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '24 [false2]' (It exists a by no one comprehensible (random) 3D image synthesis algorithm which has been integrated into a clinical workflow.)

Please write your answer here:

26 [false3]It exists a human being who is able to interpret more than three dimensional multi-variate raw data directly (e.g. DTI or PC-MRI data). *

Please choose **only one** of the following:

○ Yes

○ No

○ I definitely don't know

Make a comment on your choice here:

27 [which3] Which one? *

Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '26 [false3]' (It exists a human being who is able to interpret more than three dimensional multi-variate raw data directly (e.g. DTI or PC-MRI data).)

Please write your answer here:

28 [false4]It exists a 3D image synthesis system which is able to process dozens of input data sets, to render the result at interactive frame rates at high quality and which is already standard in clinical visualization systems or which is used for intermediate data generation in other algorithms. *

Please choose **only one** of the following:

○ Yes

○ No

○ I definitely don't know

Make a comment on your choice here:

29 [which4] Which one? *

Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '28 [false4]' (It exists a 3D image synthesis system which is able to process dozens of input data sets, to render the result at interactive frame rates at high quality and which is already standard in clinical visualization systems or which is used for intermediate data generation in other algorithms.)

Please write your answer here:

Final Comments

30 [future] Do you think that 3D image synthesis will get more important for future medical applications? *

Please choose the appropriate response for each item:

Yes	Uncertain	No
○	○	○

31 [final] Do you have any final comments on the content of this survey?

Please write your answer here:

Appendix B

Online survey on a novel tumor accessibility visualization method used in Chapter 6

B.1 Questionnaire

Informal survey on a novel tumor accessibility visualization method

We want to evaluate the usefulness of a novel visualization method which we have invented at Graz University of Technology. It would be important for our work if you could kindly provide your input through this survey. This survey is informal, which means that its purpose is to measure the degree of satisfaction of medical personell with our evaluated method.
Thank you for participating in this survey. The survey will not take longer than 10 minutes. Please feel free to email us if you do not understand a question and do not hesitate to ask also your medical colleagues to participate.

There are 28 questions in this survey

General questions

In this section you will be asked some general questions about your actual profession.

1 [maleFemale] Please tell us your gender *

Please choose **only one** of the following:

○ Female
○ Male

2 [profession] What is your current profession? *

Please choose **all** that apply:

□ Radiology
□ Surgery
□ Internal medicine
□ Physicist
□ Neurology
□ General practitioner
□ Medical student
□ Computer Scientist
□ Visualization expert
□ Biomedical engineer
□ Other: ____________

3 [experience] How many years have you been working in your primary area of expertise? *

Please choose **only one** of the following:

○ < 1 year
○ 2 years
○ 3 years
○ 4 years
○ 5 years
○ 5-10 years
○ > 10 years

4 [colorblind]

Do you have any kind of color blindness? *

Please choose **only one** of the following:

○ Yes
○ No
○ I don't know

5 [colorblindtype]Which type of color blindness do you have? *

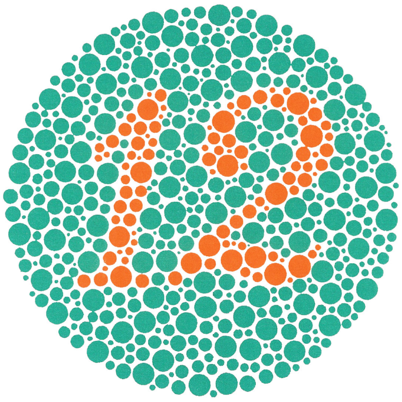
Only answer this question if the following conditions are met:
° Answer was 'Yes' at question '4 [colorblind]' (Do you have any kind of color blindness?)

Please choose **only one** of the following:

○ Red–green dichromacy (protanopia and deuteranopia)
○ Red–green anomalous trichromacy (protanomaly and deuteranomaly)
○ Blue–yellow dichromacy (tritanopia)
○ Blue–yellow anomalous trichromacy (tritanomaly)

○ Total color blindness

○ I don't know



6 [color1]

Which number do you see? *

Only answer this question if the following conditions are met:
°

-------- Scenario 1 --------

Answer was 'I don't know' at question '4 [colorblind]' (Do you have any kind of color blindness?)

-------- or Scenario 2 --------

Answer was 'Yes' at question '4 [colorblind]' (Do you have any kind of color blindness?) *and* Answer was 'I don't know' at question '5 [colorblindtype]' (Which type of color blindness do you have?)

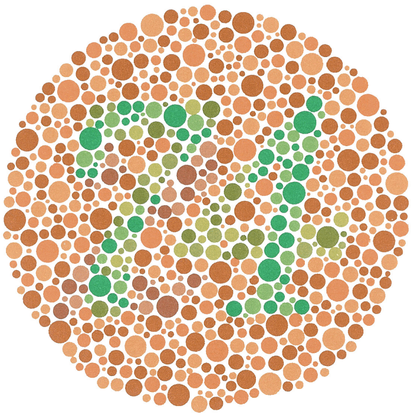
Please choose **all** that apply:

□ 71

□ 12

□ 78

□ just spots



7 [color2]

Which number do you see? *

Only answer this question if the following conditions are met:
°

-------- Scenario 1 --------

Answer was 'I don't know' at question '4 [colorblind]' (Do you have any kind of color blindness?)

-------- or Scenario 2 --------

Answer was 'Yes' at question '4 [colorblind]' (Do you have any kind of color blindness?) *and* Answer was 'I don't know' at question '5 [colorblindtype]' (Which type of color blindness do you have?)

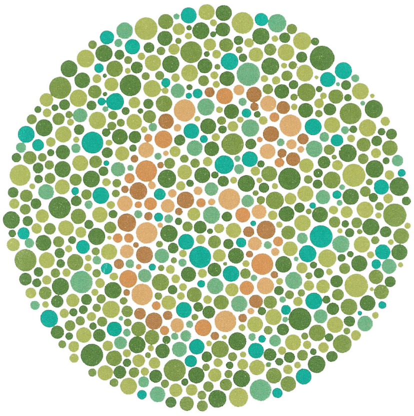
Please choose **all** that apply:

□ 71

□ 11

□ 74

□ just spots



8 [color3]

Which number do you see? *

Only answer this question if the following conditions are met:
°

-------- Scenario 1 --------

Answer was 'I don't know' at question '4 [colorblind]' (Do you have any kind of color blindness?)

-------- or Scenario 2 --------

Answer was 'Yes' at question '4 [colorblind]' (Do you have any kind of color blindness?) *and* Answer was 'I don't know' at question '5 [colorblindtype]' (Which type of color blindness do you have?)

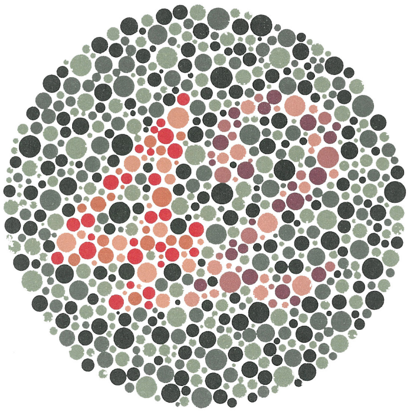
Please choose **all** that apply:

□ 6

□ 8

□ 2

□ just spots



9 [color4]

Which number do you see? *

Only answer this question if the following conditions are met:
°

-------- Scenario 1 --------

Answer was 'I don't know' at question '4 [colorblind]' (Do you have any kind of color blindness?)

-------- or Scenario 2 --------

Answer was 'Yes' at question '4 [colorblind]' (Do you have any kind of color blindness?) *and* Answer was 'I don't know' at question '5 [colorblindtype]' (Which type of color blindness do you have?)

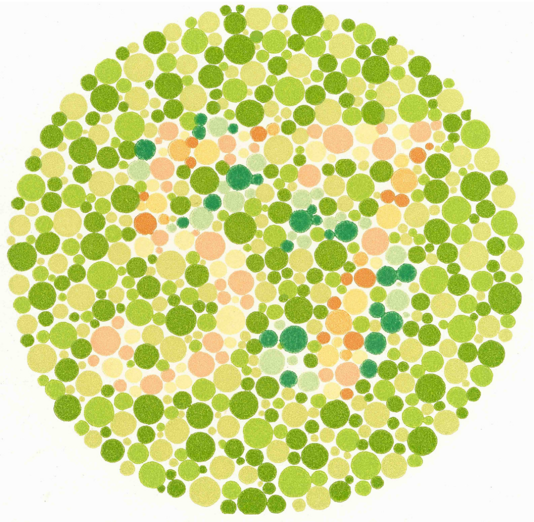
Please choose **all** that apply:

□ 42

□ 4

□ 2

□ just spots



10 [color5]

Which number do you see? *

Only answer this question if the following conditions are met:
°

-------- Scenario 1 --------

Answer was 'I don't know' at question '4 [colorblind]' (Do you have any kind of color blindness?)

-------- or Scenario 2 --------

Answer was 'Yes' at question '4 [colorblind]' (Do you have any kind of color blindness?) *and* Answer was 'I don't know' at question '5 [colorblindtype]' (Which type of color blindness do you have?)

Please choose **all** that apply:

□ 35

□ 57

□ 51

□ just spots

The new method for tumor accessability

This section shows several videos with and without our visualization method. The videos show a medical dataset with an enhances tumerous structure. In 2D the videos show a complete scroll though all slices, in 3D a full rotation of the volume.

11 [like3D]

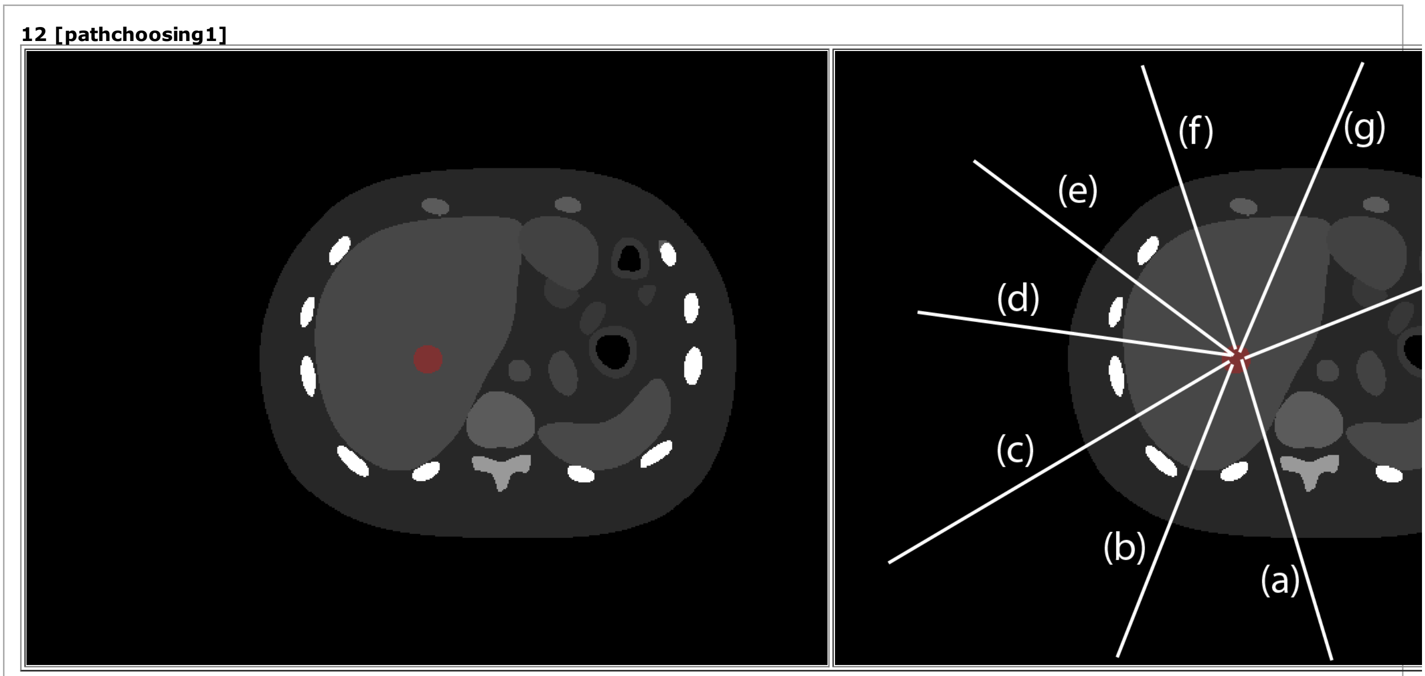
Would you use this view to choose the best way to access a tumor?

(link if embedded video is broken: http://www.youtube.com/embed/2HWWTUzx92k) *

Please choose **only one** of the following:

○ never
○ 1
○ 2
○ 3
○ 4
○ 5
○ 6
○ 7
○ always

12 [pathchoosing1]

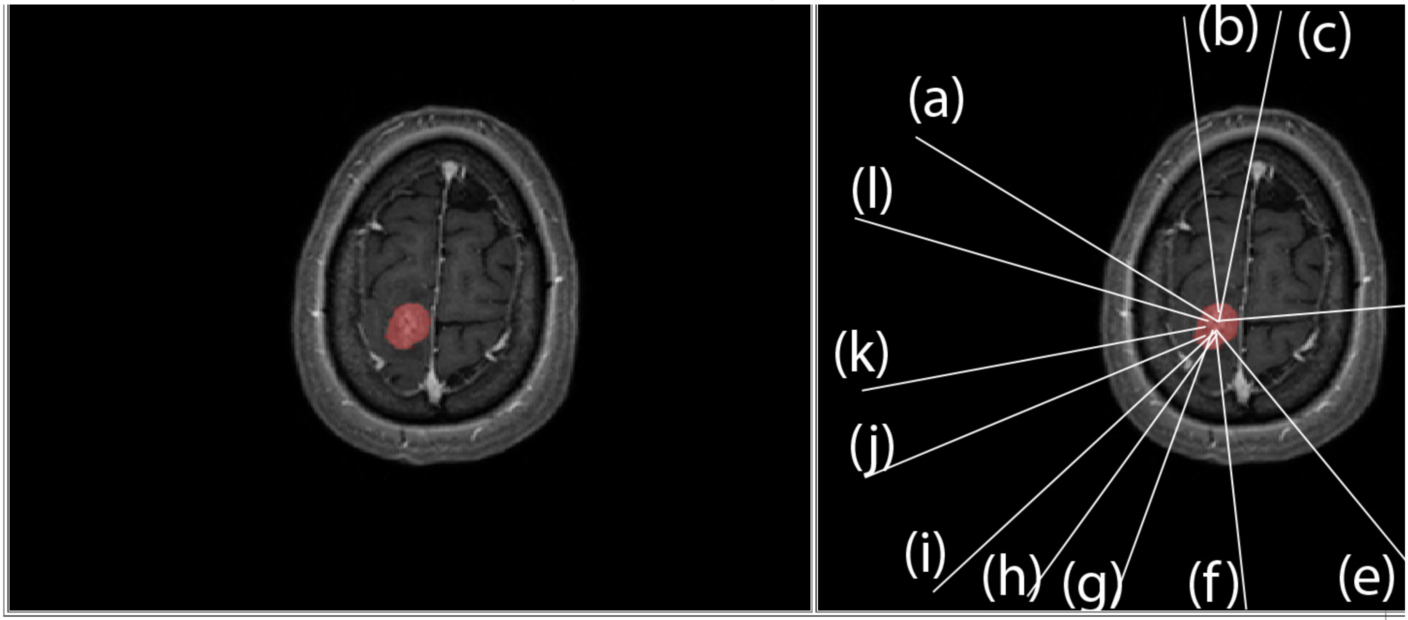


Which path would you choose to access the tumor shown in red in the center? (multiple answers possible) *

Please choose **all** that apply:

□ a
□ b
□ c
□ d
□ e
□ f
□ g
□ h
□ Other: ▭

13 [pathchoosing5]



Which path would you choose to access the tumor shown in red in the center? (multiple answers possible)

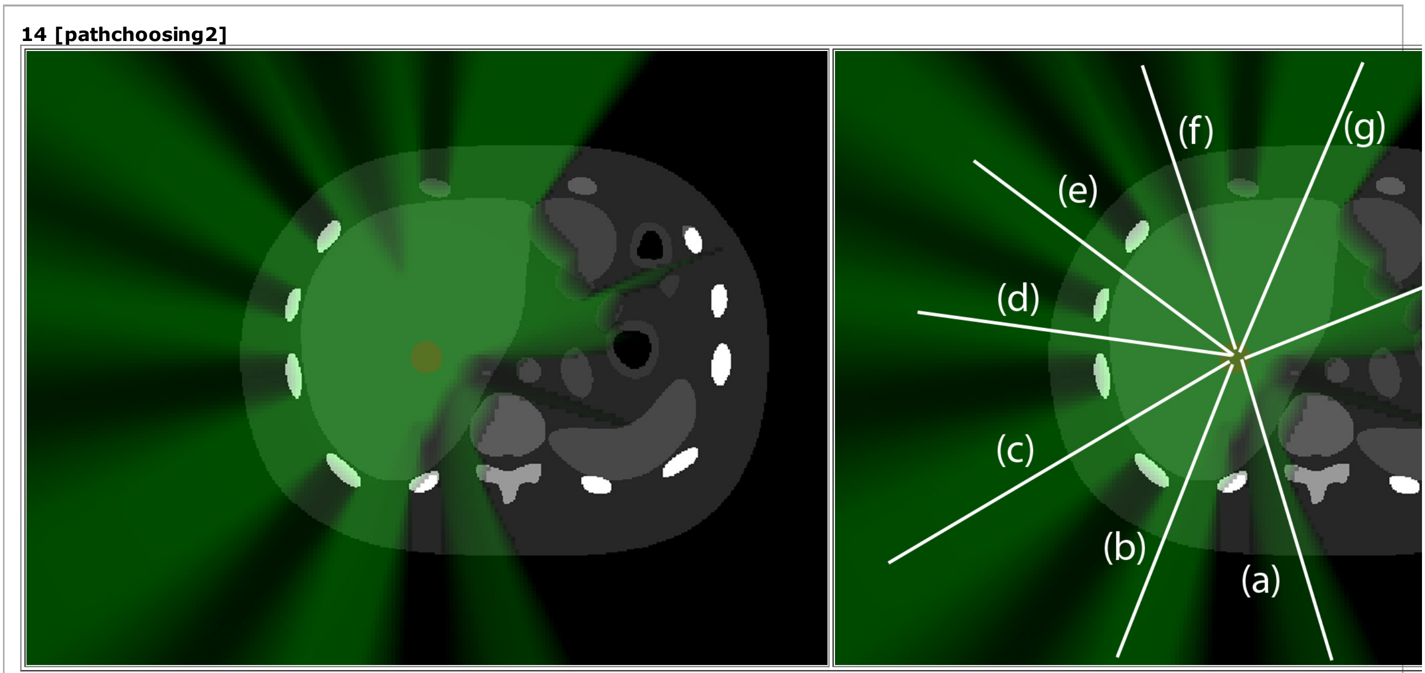
*

Please choose **all** that apply:

□ a
□ b
□ c
□ d
□ e
□ f
□ g
□ h
□ i
□ j
□ k
□ l
□ Other:

Which path would you choose?

14 [pathchoosing2]



Which path would you choose to access the tumor shown in red in the center?

The green areas indicate all paths from which at least 80 % of the tumor volume is reachable.

Vulnerable and impassable structures have been determined from CE-CT (vessels).

(multiple answers possible) *

Please choose **all** that apply:

☐ a

☐ b

☐ c

☐ d

☐ e

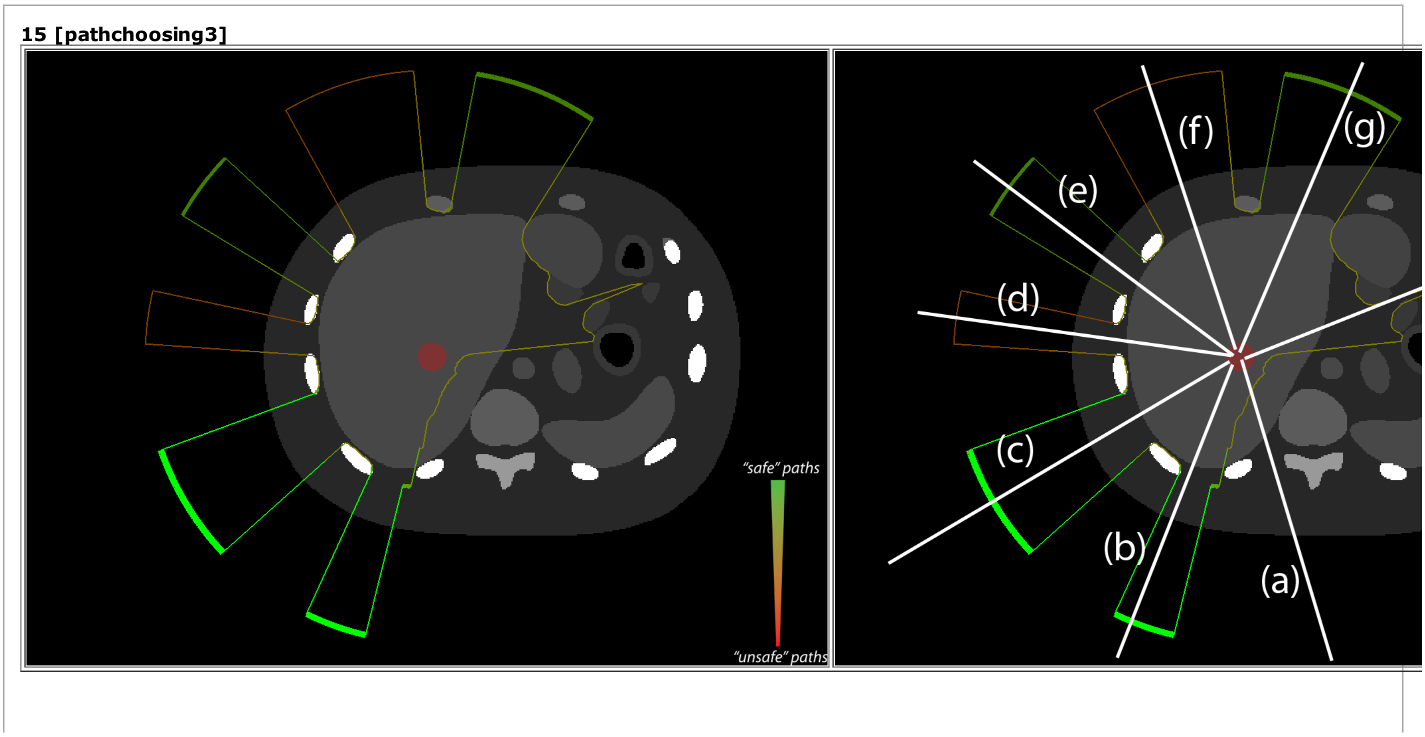
☐ f

☐ g

☐ h

☐ Other: ▭

15 [pathchoosing3]



Which path would you choose to access the tumor shown in red in the center?

The geometric lines indicate areas from which a certain part of the surface of the tumor is reachable without hitting an obstacle. The thickness and color of the lines indicates the overall size of these areas.

Vulnerable and impassable structures have been determined from CE-CT (vessels).

(multiple answers possible) *

Please choose **all** that apply:

□ a

□ b

□ c

□ d

□ e

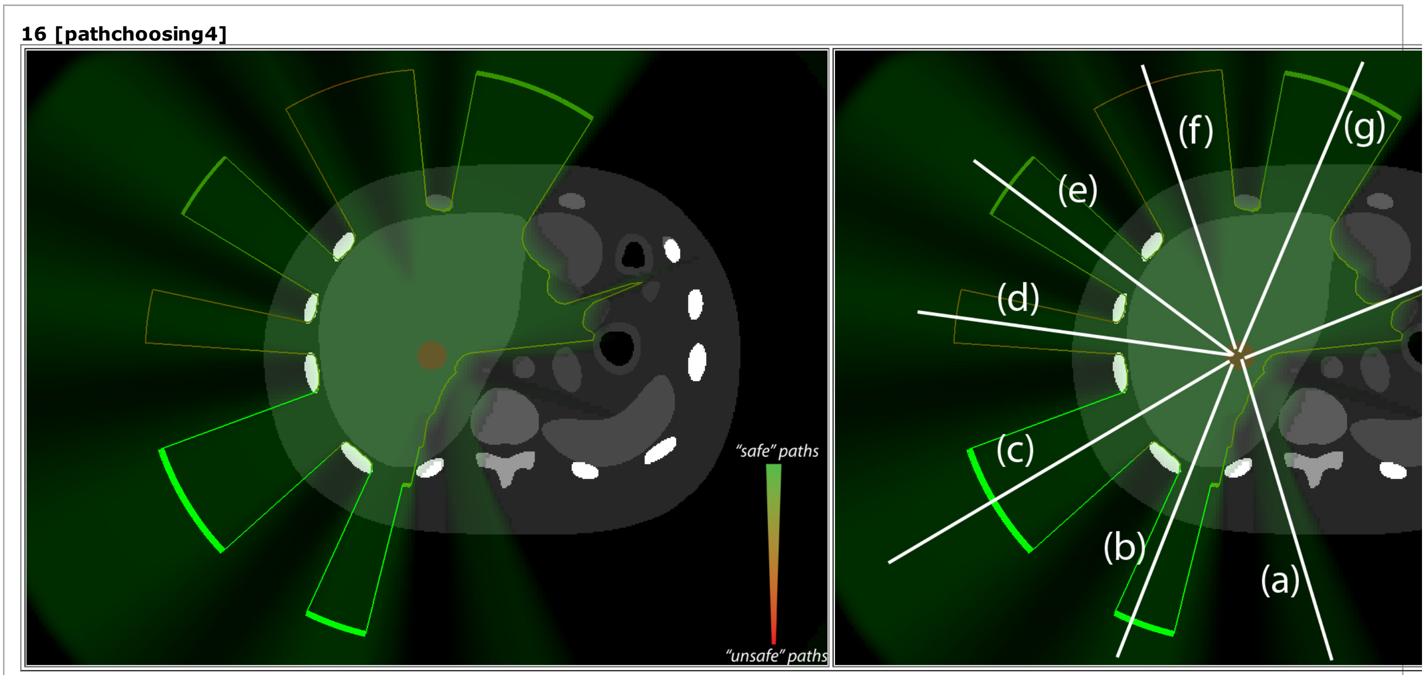
□ f

□ g

□ h

□ Other: □

16 [pathchoosing4]



Which path would you choose to access the tumor shown in red in the center?
The green areas indicate all paths from which at least 80 % of the tumor volume is reachable.
The geometric lines indicate areas from which a certain part of the surface of the tumor is reachable without hitting an obstacle.

The thickness and the color of the lines indicates the size of these areas.

Vulnerable and impassable structures have been determined from CE-CT (vessels).

(multiple answers possible)

*****

Please choose all that apply:

□ **a**

□ **b**

□ **c**

□ **d**

□ **e**

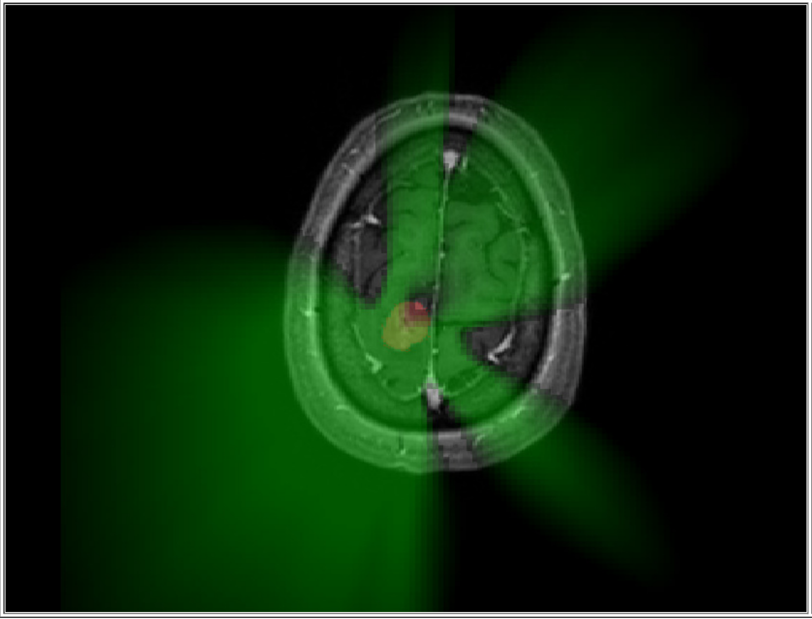
□ **f**

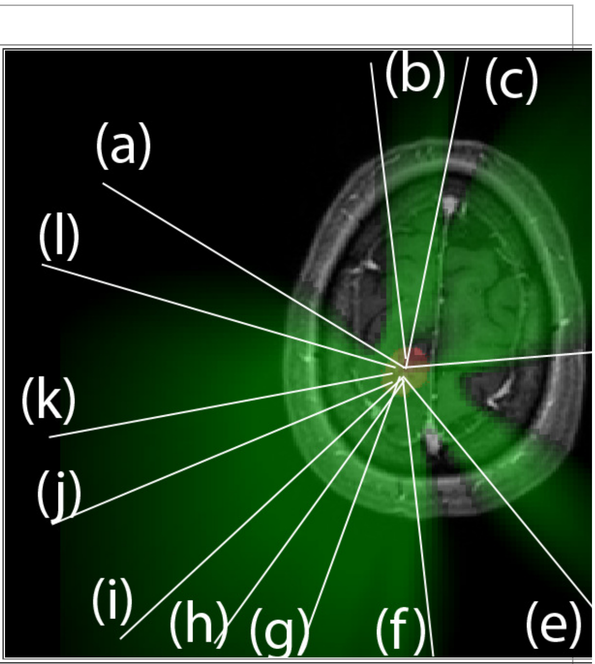
□ **g**

□ **h**

□ **Other:** ▭

17 [pathchoosing6]





Which path would you choose to access the tumor shown in red in the center?

The green areas indicate all paths from which at least 80 % of the tumor *volume* is reachable.

Vulnerable structures have been determined from fMRI, DTI and CE-MRI (vessels).

(multiple answers possible)

*

Please choose all that apply:

□ a

□ b

□ c

□ d

□ e

□ f

□ g

□ h

□ i

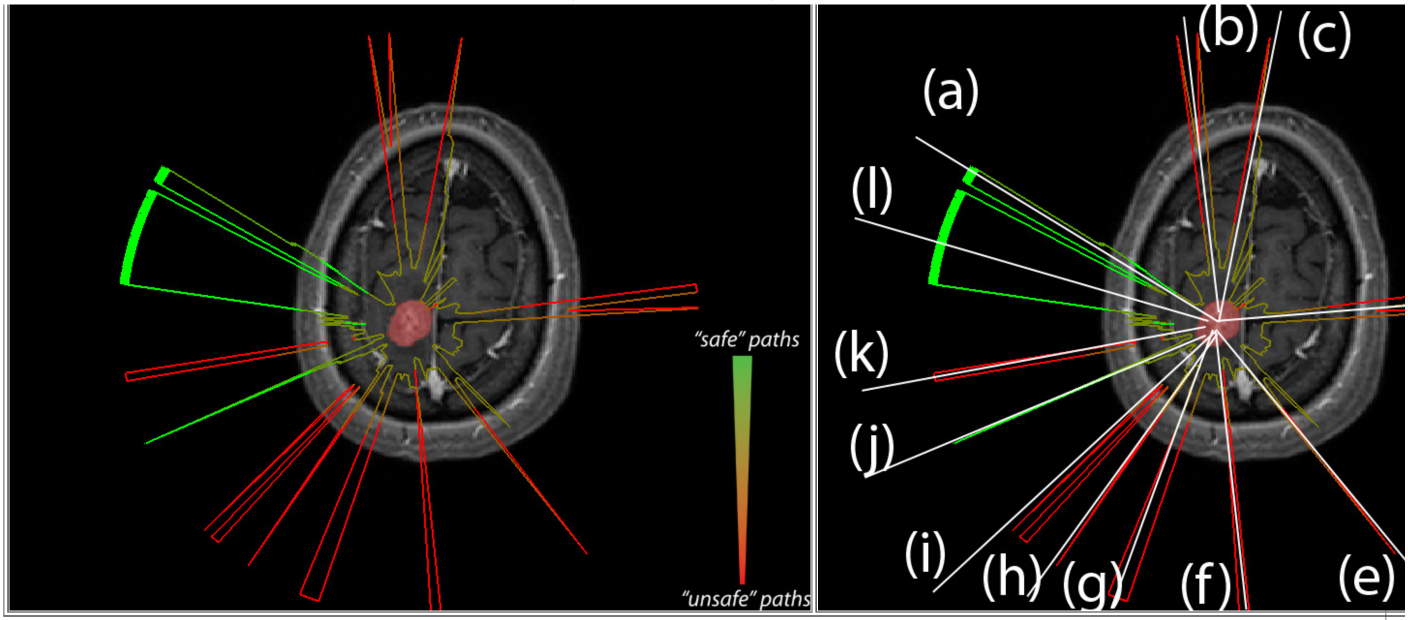
□ j

□ k

□ l

□ **Other:** ▭

18 [pathchoosing7]



Which path would you choose to access the tumor shown in red in the center?

The geometric lines indicate areas from which a certain part of the *surface* of the tumor is reachable without hitting an obstacle. The thickness and the color of the lines indicates the size of these areas.

Vulnerable structures have been determined from fMRI, DTI and CE-MRI (vessels).

(multiple answers possible)

*

Please choose all that apply:

☐ a

☐ b

☐ c

☐ d

☐ e

☐ f

☐ g

☐ h

☐ i

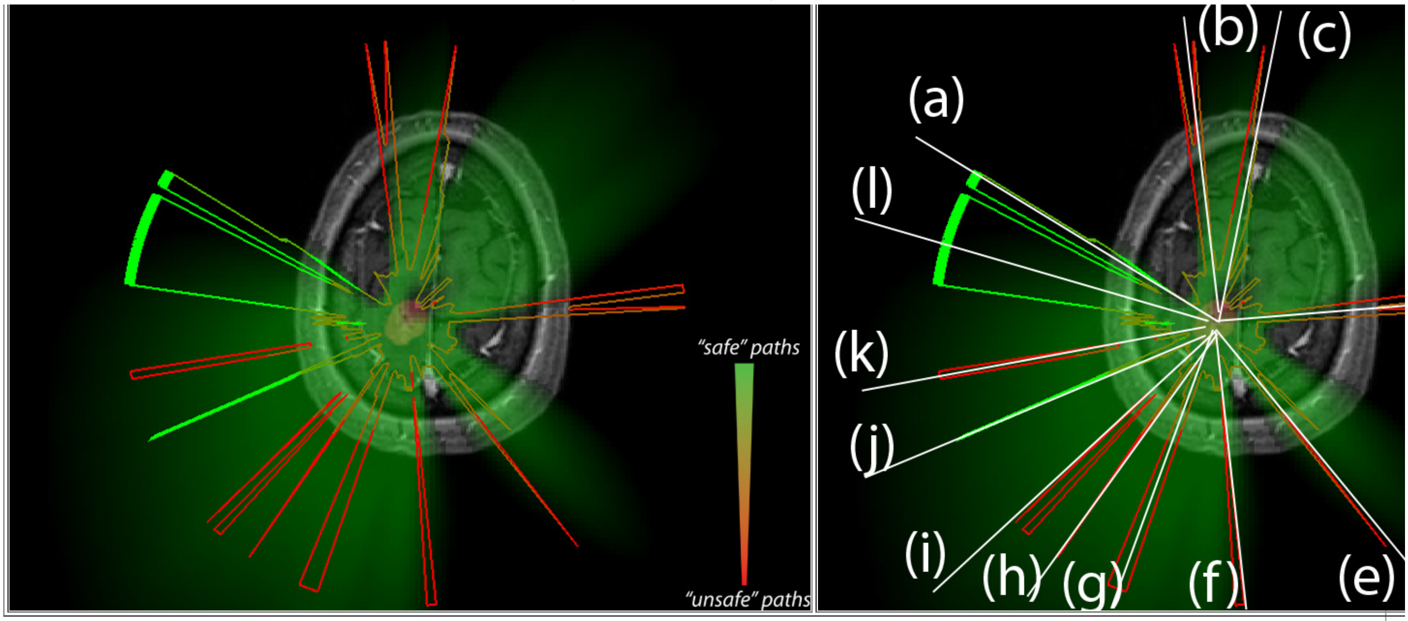
☐ j

☐ k

☐ l

☐ Other: ▭

19 [pathchoosing8]



Which path would you choose to access the tumor shown in red in the center?

The green areas indicate all paths from which at least 80 % of the tumor *volume* is reachable.
The geometric lines indicate areas from which a certain part of the *surface* of the tumor is reachable without hitting an obstacle. The thickness and the color of the lines indicates the size of these areas.

Vulnerable structures have been determined from fMRI, DTI and CE-MRI (vessels).

(multiple answers possible)

*

Please choose all that apply:

☐ a
☐ b
☐ c
☐ d
☐ e
☐ f
☐ g
☐ h
☐ i
☐ j
☐ k
☐ l
☐ Other: ▭

Which of these variations do you like?

20 [like3d2]

1) Would you use this view to choose the best way to access a tumor?

(click on the video to play).

The green areas show all paths which do not hit any vulnerable structure.

(link if embedded video is broken: http://www.youtube.com/embed/F_qMuWtqG-s) *

Please choose only one of the following:

○ never
○ 1
○ 2
○ 3
○ 4
○ 5
○ 6
○ 7
○ always

21 [like3d3]

2) Would you use this view to choose the best way to access a tumor?

(click on the video to play)

The green areas show all paths which do not hit any vulnerable structure.

The yellow/orange areas show all paths which pass minor vulnerable structures.

(link if embedded video is broken: http://www.youtube.com/embed/6iyEqWiyuyc) *

Please choose only one of the following:

○ never
○ 1
○ 2
○ 3
○ 4
○ 5
○ 6
○ 7
○ always

22 [like3d4]

3) Would you use this view to choose the best way to access a tumor?

(click on the video to play)

The green areas show all paths which do not hit any vulnerable structure.

The yellow/orange areas show all paths which pass minor vulnerable structures.

The red areas show all paths which pass highly vulnerable structures or impassable areas.

(link if embedded video is broken: http://www.youtube.com/embed/k4U8Vc9_0WA) *

Please choose only one of the following:

○ never
○ 1
○ 2
○ 3
○ 4
○ 5

○ 6
○ 7
○ always

23 [like2Donly1]

4) Would you use this view to choose the best way to access a tumor?

(click on the video to play)

(link if embedded video is broken: http://www.youtube.com/embed/JlYpdm8btZo) *

Please choose only one of the following:

○ never
○ 1
○ 2
○ 3
○ 4
○ 5
○ 6
○ 7
○ always

24 [like2Donly2]

5) Would you use this view to choose the best way to access a tumor?

The tumor is the red part of the scan.

The green areas are the green areas mapped from 3D (questions 1-3).

(click on the video to play)

(link if embedded video is broken: http://www.youtube.com/embed/8dct1Bhivag) *

Please choose only one of the following:

○ never
○ 1
○ 2
○ 3
○ 4
○ 5
○ 6
○ 7
○ always

25 [like2Donly3]

6) Would you use this view to choose the best way to access a tumor?

The geometric lines indicate areas from which a certain part of the tumor surface is reachable without hitting an obstacle. The thickness and the color of the lines indicate the size of these (safe) areas.

The tumor is the red part of the scan.
(click on the video to play)

(link if embedded video is broken: http://www.youtube.com/embed/Iik0z_tFraE) *

Please choose only one of the following:

○ never
○ 1
○ 2
○ 3