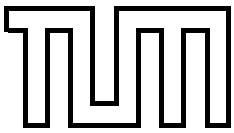
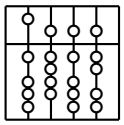


Technische Universität München
Fakultät für Informatik

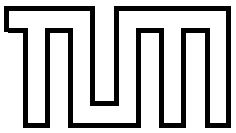


Diplomarbeit

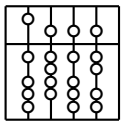
Development of a Planning and Navigation Tool for Endoscopic Treatment of Aortic Aneurysms - Computer Supported Implantation of a Stent Graft

heARt: Heart Surgery Enhanced by Augmented Reality Techniques

Martin Groher



Technische Universität München
Fakultät für Informatik



Diplomarbeit

Development of a Planning and Navigation Tool for Endoscopic Treatment of Aortic Aneurysms - Computer Supported Implantation of a Stent Graft

heARt: Heart Surgery Enhanced by Augmented Reality Techniques

Martin Groher

Aufgabensteller:	Prof. Gudrun Klinker, Ph.D. PD Dr. med. Robert Bauernschmitt
Betreuer:	Dipl.-Inf. Martin Bauer Dipl.-Inf. Hesam Najafi Eva Schirmbeck
Abgabedatum:	16. Dezember 2003

Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 16. Dezember 2003 Martin Groher

Abstract

Keyhole surgery is becoming a standard method to treat patients at a minimally invasive level. Especially in the field of cardiac and vascular surgery these treatments are fancied throughout hospitals. However, in order to enable clinicians to operate without opening the patient, certain imaging techniques are necessary to visualize regions of interest. Many procedures have been developed for this purpose, ranging from X-raying to computed tomography or magnetic resonance scanning. A problem that is raised within this context is the availability of the information created by imaging techniques since some can just be performed preoperatively whereas others are executed during an operation. The goal of this project is to enable clinicians to access imaged and processed data acquired pre- or intraoperatively during treatment, i.e. merging all visual information about a patient in order to disburden physicians when operating. In particular, a stenting operation, where leaks in the aorta are fixed by injecting a cylindrical wired tube (stent graft) that is guided to the problem region through the aorta via a catheter will be augmented virtually. Preoperatively acquired computed tomography images can be processed with the proposed system and the outcome will be visualized in the operation theatre. Moreover, intraoperative X-ray images are aligned with the CT data and hence navigational information is produced including a three-dimensional surface model, metrics, and current as well as planned locations of the stent graft. This shall help to avoid too much X-ray exposure as well as reduce damaging contrast injections and operation time.

Zusammenfassung

Minimal invasive Chirurgie entwickelt sich zu einer Standardprozedur für Patientenbehandlungen. Insbesondere in der Herz- sowie der Gefässchirurgie werden solche Behandlungsmethoden gern verwendet. Um jedoch dem Arzt einen Eingriff am Patienten ohne ein vorheriges Öffnen zu ermöglichen, sind gewisse Bild- und Visualisierungstechniken anzuwenden, um Problem- oder Behandlungszonen aufzuzeigen. Viele solcher Techniken wurden bereits entwickelt, heutzutage kann man auf ein Spektrum zurückgreifen, das von gewöhnlicher Röntgenbestrahlung bis zur Computer- oder Kernspintomographie reicht. Ein Problem, das mit diesen Visualisizerungsverfahren einhergeht ist die Verfügbarkeit der Bildinformation, da manche Verfahren nur präoperativ, andere intraoperativ verwendet werden können. Das Ziel dieses Projektes ist es den Ärzten im Operationssaal prä- sowie intraoperative Bilddaten während der Operation zur Verfügung zu stellen, d.h. jegliche visuelle Information des Patienten zu verschmelzen und somit anzureichern, um den Arzt während der Behandlung möglichst zu entlasten. Konkret wird eine Stenting Operation virtuell augmentiert, bei der Risse in der Aortawand stabilisiert werden, indem ein zylindrischer, verdrahteter Schlauch (Stentgraft) mithilfe eines Katheters durch die Aorta zur Problemzone geschoben wird. Hierfür können präoperative CT Bilder verarbeitet und selbst im Operationssaal dargestellt werden. Darüberhinaus sollen intraoperative Röntgenbilder mit den CT Daten registriert werden, um Navigationsinformation für den Operateur in Form eines dreidimensionalen Oberflächenmodells des Patienten, Metriken und momentane sowie geplante Lage des Stentgrafts bereitzustellen. Das System soll die hohe Strahlenbelastung für Arzt und Patient sowie die Injektion von schädigenden Kontrastmitteln und Operationszeit verringern.

Preface

About this work This work is the Diplomarbeit (Diploma thesis) for the studies of Dipl.-Informatik (Informatics) and was done under the supervision of Eva Schirmbeck, Martin Bauer, Hesam Najafi, Priv.-Doz. Dr. med. Robert Bauernschmitt, and Prof. Gudrun Klinker at the Chair for Applied Software Engineering and the German Heart Center Munich, respectively, at the Technische Universität München.
It is part of the Heart Surgery Enhanced by Augmented Reality Techniques (heARt) project. heARt itself is divided into two subprojects called PORT and STENT. This thesis is part of the latter one, which provides planning of stent operations and registration of angiogram images with computer-tomographic images in order to visualize information during surgery. It aims to support surgeons in the endoscopic treatment of thoracoabdominal aneurysms using stent grafts by visualizing both, the region of interest and the current position of the stent.

Acknowledgements At first, I want to thank my professors and supervisors, Gudrun Klinker, Robert Bauernschmitt, Martin Bauer, Hesam Najafi, and Eva Schirmbeck for guiding me through the computer scientific as well as the medical research field, for aggregating and granting access to information, and for arranging meetings with all persons potentially providing knowledge. Moreover, I want to thank my thesis group mates, Jörg Traub, Marco Feuerstein and Peter Keitler, who always motivated me when frustration dared to overcome myself. Thanks also to my family for making it possible to study computer science and special thanks to my sister Evi who endured and also encouraged me to carry on.

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, cardiac surgery or vascular surgery, respectively, rather seldom use techniques where the thorax is opened. In many cases it avails of keyhole surgery or even endoscopic methods. Here, only small cuts are made on the patient's skin where special instruments are injected, guided to the region of interest and handled – sometimes remotely – by the surgeon. Treatments differ from catheter injection to injection of robot or telemanipulator arms. Its benefits are obvious:
First, the mortality rate as well as the morbidity rate is much lower when endovascular treatment is performed compared to open surgery; moreover, the patient's convalescence is very short, which is not only a benefit in terms of money. In almost all endovascular operations the heart-lung-machine is not used and it is even possible to treat high-risk patients with endovascular or endoluminal techniques [4, 5, 21, 23, 41]. Similar advantages also apply to keyhole methods in cardiac surgery [15, 70]. On the other hand, there is one problem which must be solved when open surgery should be avoided: How can the surgeon actually see where to operate, discern her instruments or spot the region of interest? In order to solve these difficulties a number of imaging techniques have evolved which make it easier or principally possible to treat patients via keyhole surgery.

1.1 Imaging Techniques for Endoscopic Surgery

Basically, imaging techniques for endoscopic surgery are divided into two parts; a preoperative part, where surgeons plan an operation and an intraoperative part, which happens during treatment. On the one hand, the preoperative planning of the operation is done one day before the actual surgery; there is no time pressure – the physician can take an extended look at all image material of the patient, can compare, select or even process the data. Imaging techniques for intraoperative purposes have, on the other hand, different requirements. As time plays an important role while treating a patient, the faster the surgeon can get and view images, the better. Moreover, images used during operation should not require much interaction and thus should be edited to contain all available information at once.

Preoperative planning is mostly aided by computer tomographic (CT) or magnetic resonance (MR) images which allow surgeons to view the region of interest in 3D or browse through axial, sagittal or frontal (coronal) slices made of the body's interior (see figure 1.1). Axial slices are recorded by the CT or MR machine itself – CT mostly uses a spiral structure (Helical computed tomography) – whereas sagittal and frontal slices are computed from the axial raw data. A three-dimensional model of the tomographic data can be reconstructed by image segmentation. Its basic principle is to separate the human viscera by their characteristic gray-scale values in the CT or CTA[1] images. Most commonly, this is accomplished by defining the local minima via thresholds in a gray-scale histogram [24], see also section 3.2.2. As

[1] Computer Tomography Angiography

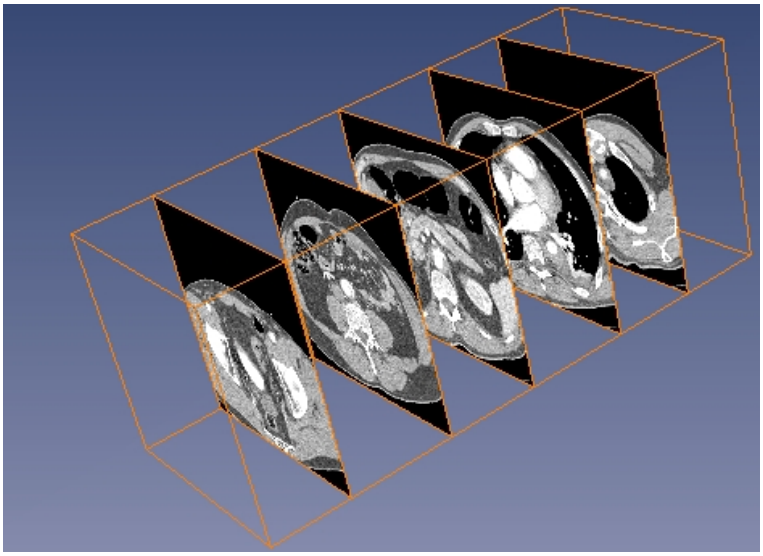
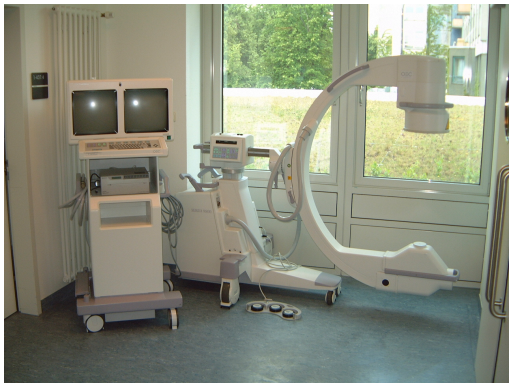


Figure 1.1: Axial CT slices showing the thorax enhanced by contrast to visualize vessels

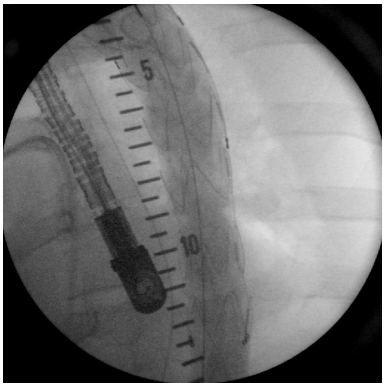
these scanning and reconstruction procedures take quite a long time and require large technical equipment, they can only be performed *before* but not during the surgery. However, CT and MR images provide a vast range of information. They visualize both, bones and organs and can, when extended by angiography even display vessels and thrombus [6]. There are two major disadvantages of MR – first, MR cannot display calcium which is essential for, e.g. vessel investigation and second, MR images can be distorted by ferro-magnetism which emerges when using metallic grafts [6, 69]. On the other hand, MR mechanisms do not expose patients to X-rays and thus are safer than CT analyses. However, CT is the dominant technique to do preoperative planning. Angiography itself or Digital Subtraction Angiography is a method to visualize the lumen (the vessel's interior) of arteries. A contrast, a liquid which reflects X-rays is injected in the vessel-to-visualize, followed by the shooting of several X-ray images. Thus, a film-sequence can be generated where the lumen of the contrast-enhanced vessel (e.g. the aorta) can be captured [20]. Angiography is used in both parts, the pre- and intraoperative one. During operation, angiography can be done with a "C-Arm", which shoots X-Ray images in short intervals (see figure 1.2(a)).[2] The X-Ray images can be viewed instantly by the operating physicians but only provide a 2-dimensional impression of the patient's body without showing any thrombus (see figure 1.2(b)).

As a matter of fact, many other techniques for visualization in medicine are used, like endoscopic (stereo-)cameras or Transoesophageal Echocardiography (TEE), but as they are of little use for image overlaying or not used in stent operations at all, they are not discussed in detail here.

[2]The reason for taking a picture sequence instead of just a single picture is the unpredictability of the contrast location at a certain point of time. Hence, for full angiogram information a sequence of images is needed.



(a) C-arm creating intra-operative X-Ray images



(b) C-arm-angiogram showing a part of the (contrast-enhanced) aorta where a stent is located (wires)

Figure 1.2: C-arm system at the DHM[3]

1.2 Aortic Stenting – Description and Practice

An aneurysm is a sac formed by the dilatation of the wall of an artery, a vein, or the heart. In this work, aneurysms in the aorta are regarded. Mostly, they are caused by degenerative diseases such as atherosclerosis and cystic medial necrosis [48]. In fact, aneurysms can also be caused by trauma or infection and are then referred to as false aneurysms whereas the former ones are called true aneurysms. A third group of aneurysms is called pseudo aneurysms mostly caused by injuries of the aorta. Aneurysms are also distinguished by their location in the aorta. Aneurysms in the abdomen are called Abdominal Aortic Aneurysms (AAA), whereas those that lie in the thorax, i.e. in the ascending or descending part of the aorta are named Thoracic Aortic Aneurysms (TAA). Big aortic aneurysms can cover both areas and are called Thoracoabdominal Aortic Aneurysms (TAAA). The wall of the aorta has three layers (adventitia, media, and intima). True aneurysms involve all three layers whereas pseudo aneurysms involve only one or two of them. Pseudo aneurysms are caused by small leaks in a layer that lets the blood stream inside the wall. The blood thus congests and coagulates leading to an expansion of the aorta at that point. A false aneurysm is one in which the blood is contained by surrounding tissues because all three layers of the aorta wall are injured [68]. Moreover, leaks in the wall of the aorta can lead to dissections, i.e. the pressure of the blood that streams between two layers causes another leak further down where it can escape the wall again. Then, two lumen are created, the one between layers is called the false lumen. The more the blood streams through this false lumen the bigger the leaks can get finally leading to a dissolving of the inner wall. Two types of dissections are distinguished, Type-A dissections where the descending and ascending aorta is involved or Type-B dissections where only the descending part is involved. There are two approaches to treat certain aortic aneurysms (AAAs, TAAs, or TAAAs) or dissections: the insertion of a bypass via open surgery or the endovascular method (stenting) where a cylindric prosthesis

[3]Deutsches Herzzentrum München

(stentgraft) is pushed through the aorta to the problem region. There, it replaces (in the case of true aneurysms) or stabilizes the inner wall of the aorta. Interestingly, mortality rates of the first approach are significantly higher (about 50%) than those of the second (less than 10%) [10, 42, 45, 46].

1.2.1 Preoperative Planning for Stenting

Mostly, aneurysms are not recognized by the patient unless they reach quite large dimensions, which is only possible in the abdominal area – TAAs lead to rupture earlier. A diameter of 6 cm or more of the aneurysm would also cause rupture in the abdomen [52]. Thus, the most common way to diagnose aneurysms is a routine check via CTA. Here, the problem is categorized and the data is prepared for the operating surgeon to be processed. As a matter of fact, there is a strict separation between surgeons and radiologists in German hospitals unlike in other country's hospitals. Radiologists merely edit and pre-process the raw data of the CT machine, which mostly results in a colored 3D reconstruction of the aorta and the spine for orientation reasons (see figure 1.3). They do not perform any surgical treatments on the patient. This is done by the operating physician who just receives the edited CT data the day before surgery, detects the location in the patient's aorta affected by an aneurysm or a dissection and determines an appropriate stent, which is mainly characterized by its length and its diameter. Sometimes the stent graft must have a special shape, if the aneurysm covers more branches of the aorta like the two iliac arteries as shown in figure 1.4. In planning, as well as during implantation, special attention has to be paid on not covering branching-off vessels like the carotid artery or the coronary arteries. The needed information is offered by CTA, and, additionally, CTA data provides important knowledge about the aortic anatomy which is necessary for catheter navigation and also for detection of potential problems in operation, like kinkings of the aorta or the femoral artery which also has to be traversed by the catheter. Unfortunately, all this data is not accessed during the operation. Hence, the physician has to keep in mind where the aneurysm is located, where kinkings would cause the injected catheter to stop or where branching-off vessels are connected to the aorta.

1.2.2 Stent Implantation

When the patient has been anaesthetized and prepared for surgery, a pigtail catheter is inserted into the patient's shoulder and advanced to the region of interest. It is used to inject contrast into the aorta in order to visualize the lumen in X-ray images shot by a C-arm. Then, the femoral artery, which is well palpable since it is located directly below the skin, is detected and dissected with a scalpel. By using a guide wire with a flexible tip a catheter is inserted on which the stent graft (see figure 1.5(a)) is attached (Seldinger technique, [49]). It is advanced through the aorta until the surgeon gets the feeling of increased pressure (she may have reached a kinking) or that she has reached the aneurysm's region. In order to assure the current location of the stent graft an X-ray picture sequence is taken by the C-arm enhanced by contrast as described in section 1.1. When treating TAAs, the way of the catheter is quite long: it must be pushed all the way through femoral, iliac, (infra)renal and thoracal to de- or ascending aorta. Therefore, many X-ray images must be taken and contrast must be injected to avoid complications. Moreover, as the imaging possibilities of C-arms

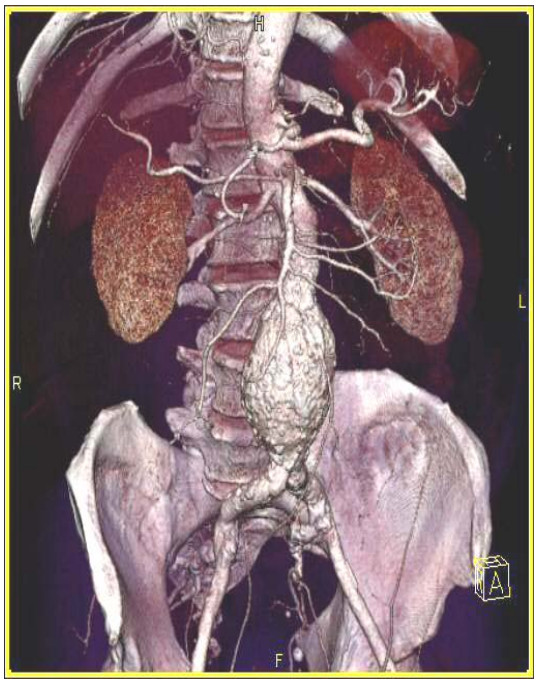


Figure 1.3: A 3D reconstruction of an abdominal aortic aneurysm

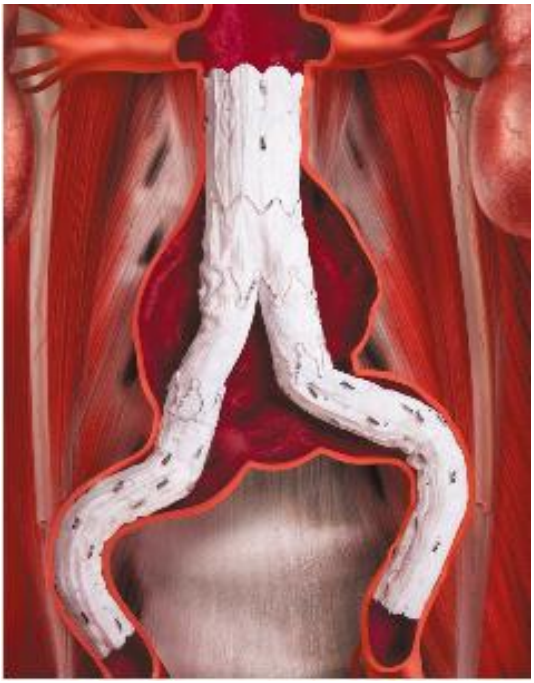
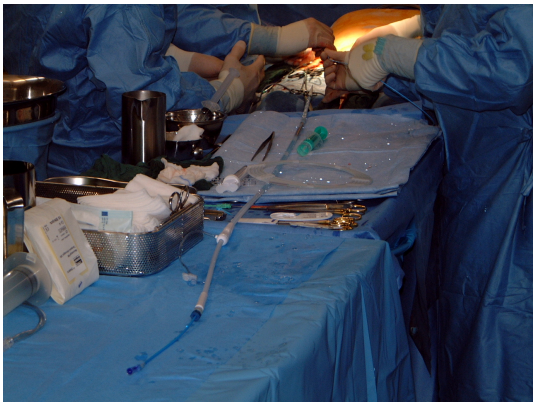


Figure 1.4: Stent graft for an AAA which also covers the two iliac arteries

are rather limited an image is taken in the very beginning of the operation in order to have a certain impression of the aorta and its branches. This image is captured by merely drawing the contours on the monitor as shown in figure 1.5(b). When another X-ray image sequence is taken, it is displayed on the monitor and compared with the drawn contours. Naturally, after drawing the contours, the C-arm must not be moved any more. When the surgeon is certain to have the stent in the right place, it is expanded and the catheter is extracted. In some cases, an inflatable balloon is injected and guided to the stent in order to fix it to the aorta wall by using air pressure (balloon-expandable stent grafts).

1.3 Computer Aided Navigation and Planning Tool

A Computer Aided Navigation and Planning Tool (CANP) will be developed in this project. It will help the surgeon to plan a stent operation by determining the best location a stent is to be placed for fixing an aneurysm or a dissection. Image modalities to be used in this procedure are CT data (axial, frontal and sagittal slices) and an appropriate three-dimensional reconstruction (3D surface model). The planning of the best location can be accomplished by setting landmarks in the slices and the 3D reconstruction. An algorithm calculates an approximate surface including all landmarks. Thus, CANP is providing methods to virtually place and display a stent at the intended position in an existing CTA model. The information can be stored persistently and will serve as a reference for the surgeon to control the implantation progress during operation.
Moreover, a navigation procedure will be provided by the system which, after registering



(a) A new stent graft attached to a catheter

(b) Monitor showing an X-ray image where the contours of the aorta and its branches are plotted (left screen)

Figure 1.5: Stent and C-arm used in an implantation operation

two different image modalities (3D CT data and 2D X-ray data), displays the momentary position of the stent during the operation in the 3D surface model. The registration is accomplished by simulating the creation of an angiogram via the C-arm: The C-arm's camera parameters are determined (see chapter 4) and a virtual camera is initialized with them. From the CT data an artificial X-ray image (DRR[4]) is computed and viewed via the virtual camera. Then, the real X-ray image and the DRR are aligned and hence registered. Afterwards, the stent's current position, which is only shown in the real X-ray image can be transferred to the CT data and thus displayed in the 3D reconstruction. When augmenting the 3D data model with the planned stent, the approximate distance from current to desired location of the stent can be determined, which tells the surgeon how far the stent-catheter is to be advanced inside the patient's body. Since CT data includes metric information, the actual dimensions of aorta and stent can be incorporated in the proposed system.

1.3.1 Medical Motivation for CANP

Stent operations are an alternative for conventional open surgery methods. As these keyhole methods are rather new and always rely on computer aided imaging, physicians sometimes prefer the traditional way of treating aneurysms. The technique, however, is believed to be very promising and more and more hospitals tend to treat aneurysms via keyhole instead of conventional open surgery. Due to the early stage of development it is not surprising that the usage of stents is not very experienced but has a rather improvisational character as shown above. In particular, there is no technical equipment to meet the special needs of stent implantation besides the standard equipment like CTA and C-arm. Currently, the three-dimensional CTA model is only used for preoperative planning but is not available during operation except in the surgeon's memory. This situation is unsatisfactory for CTA provides

[4]Digitally Reconstructed Radiograph – an artificial X-ray volume created from CT data via volume rendering

by far more anatomical information than the two-dimensional intraoperative fluoroscopic[5] images taken by a C-arm. The C-arm, however, is necessary for providing information about the current position of stent or catheter. Merging the two image modalities would provide the surgeon with a vaster range of information which could result in a faster operation procedure: surgeons are reported about the current positions of the catheter, thrombus, the region where the stent should be placed and where kinkings or anomalies occur. Furthermore, to use the C-arm means grave X-ray exposure to both, patient and operators. This exposure as well as the damaging contrast agents can be reduced since the C-arm just has to be consulted for controlling reasons – ideally two times, at the beginning and at the end of the surgical treatment. Compared to the 10 to 15 times the C-arm is used in current stent operations this would mean an obvious improvement of the implantation procedure in time and patient's health.

1.3.2 Feasibility of CANP – Medical Reasons

CANP helps physiologists in treating aortic aneurysms via keyhole surgery. The method appears to become standard treatment for aneurysms due to its many advantages compared to open surgery. Consequently, preoperative CT data must always be on-hand and a C-arm must be used for image acquisition during treatment. This data and equipment is always available in hospitals doing keyhole surgery, and hence, can serve as input for CANP. Additionally, CANP uses a 3D reconstruction of the CT data, a 3D surface model. In many hospitals, radiologists are capable of and have the applications to reconstructing three dimensional models from CT data, which can then be processed by CANP. As there is no other preliminary data necessary to deploy CANP, the project's feasibility is assured from medical view.

1.3.3 Feasibility of CANP – Computer-scientific Reasons

As time plays a significant role in the operating room only short computation delays are accepted. Thus, to gather the image information provided by the C-arm and by the CTA, to extract and merge it and to compute and output the result must be accomplished quickly. Due to rapid development in the semiconductor and IT industry, image processing becomes more and more feasible even with handling large data sets[6]. Besides main memory and CPU power, the most important device for doing graphical applications are graphic cards. For CANP, graphic cards must supply volume rendering via texture mapping (ideally 3D). In this project an NVidia Quadro4 900 XGL was used, which could handle all requirements. Besides hardware issues, certain algorithms must be used to accomplish the project's task. Mainly the registration of CT data and C-arm data is meant to be done efficiently. Several registration techniques allow 2D pictures (C-arm images) and 3D pictures (CT model) to be overlaid quickly, which makes it possible to visualize all information needed by the surgeon [47].

[5] The term fluoroscopy is actually equivalent to X-ray with the slight difference that the intensity of radiation is lower than the radiation when doing X-ray. Nowadays, however, the radiation intensity is at a low level and the two terms can be used as synonyms

[6] CT data when saved in the standardized uncompressed DICOM format can have more than 500 MB for the whole thorax region

1.3.3.1 Accessing Medical Data

One of the issues that have to be addressed when processing CT data is how to access and work with this data. Thus, there must be a certain format whose description is accessible. Fortunately, a standard format for CT data has evolved, which has replaced all the proprietary older formats. The DICOM[7] format is used in many hospitals to digitally store and process CT data. Its header is capable of saving plenty of information from the patient's name and the hospital the CT was made to slice dimensions (x- and y-pixel pitch, slice thickness and slice spacing) [44].
Actually, there is no such format for C-arm data, which either means to deal with proprietary formats of the C-arm's manufacturer or convert the data into a well-known format.

1.4 Related Work

There is a similar project at the Swiss Federal Institute of Technology in Lausanne (EPFL) covering the planning of the right stent placement and an intraoperative navigation [55, 17]. The approach was to register CT data and X-ray data preoperatively and merely compute a transformation (2D-2D homography) of preoperative and intraoperative X-ray images, which was transferred to the CT data. Thus, during preoperative registration time is not the critical factor and a costly non-rigid registration was additionally performed implicating tissue deformations. This will not be accomplished in the heARt project since X-ray images are only available during operation and preliminary work should be minimized. Moreover, a stent planning tool was conceived at the EPFL focusing on the simulation of stent and aortic movement.

1.4.1 Calibration

Calibration using one of the approaches the heARt project followed is elucidated in [72] for augmented reality applications. In the medical field, especially for calibration of X-ray cameras or image intensifiers [7, 57] explain all difficulties encountered when solving this problem. The papers claim a calibration body that is fixed rigidly to the C-arm since calibration is always necessary when moving the C-arm due to a dependency on the earth's magnetic field as described in 4.1.1.4. Distortion correction was vastly evaluated by [64, 58, 65] describing all forms of distortion that can come up when calibrating X-ray image intensifiers as well as different correction methods.

1.4.2 Registration

An interesting work covering the 2D/3D registration of X-ray images and CT data using rigid as well as non-rigid registration methods is the Ph.D. thesis of Michael Roth from the Technische Universität München [56]. Here, the intention was to conceive new methods for 2D/3D registration based on DRRs and avoiding fiducials used in conventional approaches.

[7] Digital Imaging and Communications in Medicine

However, the registration techniques were developed for single spinal bones and were not applied to thoracic X-ray images. More information about 2D/3D registration in medical environment using X-ray and CT modalities can be found in [47, 22, 27]. A general overview of registration is given by the survey [8], and, more recently, [80]. Confining the registration problem to the medical area, a nice classification of registrational problems can be found in [40].

1.4.3 Volume Rendering

Ertl et al.[8] provide a vast overview of all possible direct and indirect volume rendering techniques [12, 53]. Members of this research group also offer an open source project with an application for volume rendering and visualization[9]. An overview of texture mapping is given by Heckbert in [26], texture mapping applied to direct volume rendering is presented in [13, 50, 78].

Leaving the research sector, there are some companies in the industrial area covering the field of image-guided surgery. For instance, BRAINLAB[10] offers applications that guide physicians through neurosurgical but also orthopedical processes. It also sells a tool for registration of computed tomography with fluoroscopic images. As a matter of fact, BRAINLAB possesses a rather general patent for 2D/3D fluoroscopy/CT applications. Thus, besides mere visualization of stent graphics, more systems concerned with computer aided navigation and planning for stent implantations could not be found in industry.
Image processing tools for CT images or fluoroscopy, however, are widespread throughout the medical-technical market. Almost every company that offers medical devices, like SIEMENS or GENERAL ELECTRIC also provides software to view and process image data. For instance, SIEMENS sells their SOMATOM computed tomography series (see section 3.1.1) with software that can segment and visualize acquired image data. It can create 3D models from DICOM slices. Unfortunately, these models cannot be processed exteriorly – the only possible way to export models is via a film sequence. Moreover, such applications are highly specific and are just available in combination with the according device.

1.5 Structure of the Document

After this short introduction the requirements as demanded by clients and users at the German Heart Center Munich (DHM) are elucidated and analyzed in chapter 2. After the stent implantation has been evaluated as-is, a visionary scenario is presented followed by a software engineering process ending in the definition of the solution domain and the proposed system.

In chapter 3, the proposed system is going to be described in detail. An overview of hardware and software used in the system is given. It covers the preoperative image acquisition step, computed tomography as well as the intraoperative one, fluoroscopy. Afterwards, the basics of image processing for CT data as well as the principles of scene graphs managing

[8] http://www.vis.uni-stuttgart.de/eng/
[9] http://openqvis.sourceforge.net/index.html
[10] http://www.brainlab.com

complex visualization tasks as demanded by the graphics engine of the system are introduced. Then, the engine realizing the image processing and computer graphic requirements is presented. This section is followed by a short tract about the design of the system's architecture. Afterwards, the planning part of the system is highlighted and functional as well as implementational details are provided.

For the navigation part of the system, it is necessary to introduce camera calibration and registration of two- and three-dimensional data sets. Thus, chapter 4 explains the crucial foundations of camera calibration followed by a detailed description of the application of all calibration methods to the intraoperative image machine, the X-ray image intensifier. Then, an evaluation of the quality of the camera calibration is given and compared to similar calibration tasks.

Chapter 5 will present the principles of 2D/3D registration in the medical field by introducing a catalogue that summarizes all registrational problems faced in the medical sector. This is followed by a section of all basic principles of the registration problem that was coped with in this project. The chapter concludes with the actual description of the navigation module applying all results and procedures of the previous sections as well as of chapter 4.

Finally, chapter 6 concludes the work with a short summary of the outcome. Then, the potential benefits are highlighted followed by a review of requirements and an evaluation if they have been met or not. Afterwards, problems encountered during the project are worked out and discussed thoroughly. Finally, possible future work is proposed, which could grow this prototype to a full-fledged medical application.

2 Requirements Analysis

This chapter describes how information was extracted from the user in order to define the system's purpose and boundaries. First, the scenario as conceived by developers and client is shortly mentioned, then the process of acquiring and sharing information is illustrated and finally, the analysis of the application domain is finished by defining use cases, boundary, entity and control objects as well as sequence diagrams in order to describe static and functional behavior of the system. The requirements analysis was performed as depicted in [9].

2.1 Scenario

2.1.1 Current Situation

As already mentioned in sections 1.2.1 and 1.2.2 the operating physician only accesses CT data the day before the actual operation and decides where to place a stent and which dimensions (characterized by diameter and length) it is supposed to have. Sometimes she also receives a 3D reconstruction of the CT data from radiology for better visualization. When operating the patient the CT data cannot be accessed any more, the only resource for image information is the C-arm producing sequences of X-ray pictures of a small area of the aorta enhanced with contrast.

2.1.2 Visionary Scenario

With CANP the surgeon can plan her operation in advance with all image information available: CT data (axial, frontal and sagittal slices) and a scaled 3D reconstruction (3D surface model) of the patient. The slices can be browsed through until the aneurysm or the dissection is found. Its region is marked interactively and a graphic of a stent is computed and displayed in all slices and the 3D surface model. Additionally, the aorta is marked as a preliminary task for accurately registering CT data and X-ray data in the operation theatre. This must be accomplished as fast as possible to avoid loss of convenience. The augmented model can be saved. In the operating room, the surgeon or an assistant loads the augmented model and the latest X-ray image from the C-arm. After either aligning the two image modalities interactively or automatically (*registration*, *overlay*), the user decides whether the registration is satisfactory (then, the current position of the stent can be visualized in the 3D surface model by marking it on the X-ray image and back-projecting the mark to the CT data) or not in which case the procedure has to be repeated.

2.2 Requirements Elicitation

There were numerous occasions where information about the application domain could be extracted from users and clients, as well as meetings in which the purpose of the system was defined in terms of the solution domain. There were two demos at the DHM, one as the project started and one at its end. In the first one the scenario was described and some user interface mock-ups were shown, in the second one a finished prototype of the system was introduced. Both times users as well as clients granted plenty of feedback, which was included into the system's requirements.
Moreover, the design and implementation work was partly done at the DHM where feedback could be given by surgeons and assistants or radiologists could be consulted. There were meetings where developers and physicians could confer. Some issues of (medical) lessons learned should be pointed out:

- The question arose how to accomplish the registration of C-arm and CT data. Basically, two approaches were worked out, one with using markers to either track the patient or the C-arm or both or to have some reference points for the image overlay to be made easier; the other approach abandons the use of markers, so the overlay is only done via references to boney structures (which also meant a certain knowledge of the body's anatomy). The latter one was the preferred method for physiologists.
- For surgeons it was most important that metric measures could be made after registering the two image modalities. That is, when the current and desired stent location are determined they want to know how far (in cm) they have to advance the catheter attached to the stent graft to reach the region of interest. Furthermore, when planning the stent its dimensions (at least its length) should be calculated.
- The user interface was supposed to be as simple as possible. During operation the handling of the software happens in a small, crowded and thus sometimes quite hectical environment where too many functions could confuse the user easily.
- The clinical requirements like sterility were not as important as expected since the system was considered a prototype from the outset. After the project the clinical requirements could be fulfilled in a second phase.

2.3 Functional Requirements

The functional behavior describes the relationship between the system and its environment. It should be independent from implementation details.

2.3.1 Preoperative Planning

The following requirements have been worked out:

- The system should show all, axial, frontal and sagittal slices. Since the raw CT data only contains axial slices, the two other slice directions must be computed. Moreover,

the system should offer functionality to browse through, rotate, scale and translate the slices.

- The system should be able to handle 3D reconstructions (3D surface model) of CT data in order to create a three-dimensional impression for the user. Also, the model should be freely movable in space to regard it from any position and direction.
- Virtual markers can be set in axial, frontal and sagittal slices as well as in the 3D surface model in order to describe pose and measures of the stent graphic. These markers can be added, removed and moved at least. For convenient planning, the shape and color of the markers must be distinguishable from the actual images or model in which they are placed.
- After computing the stent graphic, it is displayed in all image modalities and can be altered when adding, moving or removing one or more markers. Again, the color is to be distinguished from model or slices. Moreover, there should be a possibility to make the model transparent in order to visualize the stent even when it is enclosed by the aorta. Additionally, the stent's length is displayed.
- In order to reuse the planned stent graphic during operation, it can be stored persistently. Optionally, the virtual markers can also be saved in case of re-planning the treatment.

2.3.2 Intraoperative Navigation

For intraoperative navigation the system's interface should appear as simple as possible to avoid confusion of the user:

- Three viewers are shown, one shows the latest X-ray image (a selected picture of the angiogram) from the C-arm, one the artificial X-ray image (DRR) computed from the CT data and one both, angiogram picture and 3D surface model for orientation.
- Sliders for rotation, translation and scaling allow the user to interactively transform and thus align DRR and X-ray image (The transformation is applied to the DRR as well as the 3D surface model)
- A button starts the automatic registration which fine-tunes the alignment.
- A second button calculates, after the current position of the stent is marked in the X-ray image, a back-projection and displays the stent-position in the 3D surface model.
- The size of the stent as it is used in the operation theatre is considered and the stent-model that appears in the 3D surface model is adjusted to this size. Furthermore, when current and desired stent location is determined, the distance between them is shown.

2.4 Analysis

This section describes shortly how use cases were defined, boundary, control and entity objects were derived and sequence diagrams were produced.

2.4.1 Use Cases and Flow of Events

The scenario was abstracted and use cases, actors and a flow of events were determined. Appendix A shows the use cases depicted in figure A.1 as well as a formal description of the flow of events. Shortly summarized, two use cases were determined, one that covers the planning procedure of the stent placement, **markStentLocation**, and a second that describes the intraoperative navigation procedure, **placeStent**.

The former one consists of seven steps beginning with the activation of the planning tool and the loading of the required CT data. Then, the surgeon can examine the data, detect the aneurysm and plan the best location for the stent to be placed. The planning is accomplished by marking the stent location. This is followed by an automatic computation of an approximate stent graphic from all markers set by the surgeon. Afterwards, the surgeon can examine the approximate graphic and move or delete it if not appropriate. The run of the aorta must also be marked by the surgeon for the registration procedure during the navigation process. This can be done at any time during the planning procedure. In a last step, the surgeon confirms all planned data by saving the augmented model.

The latter use case, **placeStent**, is composed of six steps that describe the interactive and automatic registration as well as its outcome and the benefit of back-projection. After starting the navigation mode and loading the latest fluoroscopic image from the C-arm as well as the augmented CT data given by planning, the system offers an interface for aligning CT data and C-arm data interactively. After an approximate overlay of the two image modalities, CT and X-ray, the tool computes an exact registration and thus merges all information from the two data sets via back-projection as described in chapter 5. The current location of the stent is displayed in all modalities as well as navigational information, i.e. distances from current to desired position. This informs the clinician about the route of the catheter to take and where to stop for putting the stent to the right position. Naturally, the alignment can be repeated if the registration is not satisfactory due to, for instance, a bad quality of the latest X-ray image. After advancing the stent to the desired position, another X-ray image can be recorded in order to test if the location of the stent is suitable. Then, the stent can be expanded and the implantation procedure is finished.

2.4.2 Identifying Objects

From the use cases and the flow of events, objects can be derived and divided in three classes. *Boundary objects* communicate directly with actors, i.e. all display and interaction objects are boundary objects, *control objects* control the work flow and *entity objects* describe software or hardware entities or entities of the real world. Usually, actors are transferred to entity objects, however, this step was left out and only those objects that were implemented in the system are described in the following.

2.4.2.1 Objects for Preoperative Planning

One boundary object and one entity object were depicted:

1. *PlanningDisplay*: The boundary object which handles the user interaction. It also communicates with the second actor, the `CT data`.

2. *StentComputation*: An entity object for computing a graphic for the stent. This object gets markers as input and creates a triangle surface.

2.4.2.2 Objects for Intraoperative Navigation

One boundary and two entity objects were determined:

1. *RegistrationDisplay*: The boundary object which handles the user interaction but also the back-projection from X-ray to CT data. Only this object communicates with the actors `Surgeon`, `C-arm` and `CT data`.
2. *Interactive and Automatic Registration*: Two objects – one handles the interactive registration, one the automatic registration of the two image modalities.

In both, the planning and the navigation procedure the work flow is handled by the boundary objects. Control objects were abandoned.

2.4.3 Sequence Diagrams

Sequence diagrams show how the identified objects communicate with the actors and each other, i.e. how the system functionally behaves. Images of the diagrams can be found in appendix A.

2.4.3.1 Sequence of Preoperative Planning

As shown in figure A.2, the system creates, when started, a display (`StentDisplay`) for the planning procedure, which initializes the other objects and receives the `CT data`. It then displays all slices (axial, frontal and sagittal) as well as the 3D reconstruction (3D surface model). The `Surgeon` now marks the pose of the stent and, if appropriately planned, initializes the computation of the stent graphic (`StentComputation`). When the computation is finished, a stent graphic is returned. Then, the `Surgeon` marks the aorta. Both, stent graphic and aorta markers are stored in the CT model (slices and 3D surface model).

2.4.3.2 Sequence of Intraoperative Navigation

Figure A.3 shows how the objects derived in 2.4.2.2 interact with each other and the `Surgeon`, `CT data` and the `C-arm`. After starting the navigation tool, a display is created (`RegistrationDisplay`), which initializes the interactive part of the navigation (`InteractiveRegistration`) and fetches the `CT data` and the latest X-ray image. It displays the 3D surface model, the X-ray image and an artificial X-ray image (DRR). The `Surgeon` aligns the X-ray and DRR image interactively, which is handled by the `InteractiveRegistration` object. After roughly aligning the images, the `Automatic Registration` is initialized, which fine-tunes the alignment and returns the fully registered images displayed by `RegistrationDisplay`. The `Surgeon` now has to mark the stent in the X-ray image (visible only in this image modality) and the system automatically shows the stent in both image modalities, X-ray image and `CT data` (3D surface model).

2.5 Nonfunctional and Pseudo Requirements

The *Nonfunctional Requirements* cover all aspects of the system, which are observable by the user but do not apply to functional behavior. Since the system was conceived as a prototype error handling, behavior in extreme situations, quality of service and security were left out.

The *Graphical User Interface* (GUI) will be adopted from amira, the graphical core of the system. amira is a graphic application developed by INDEED VISUAL CONCEPTS in Berlin, Germany, which is capable of dealing with complex graphical scenes by adopting a scene graph methodology as well as render two- and three-dimensional data in various ways. Moreover, many features are provided for processing and displaying all kinds of graphical formats. For an extensive explanation of all features of amira used in this project, refer to 3.3. The code that will be generated is documented by an automatic documentation tool, Doxygen[1]. All the hardware that is used depend on the system requirements of the graphic engine, amira. OpenGL and texture mapping support must be provided, ideally graphic hardware with a texture memory capable of three-dimensional mapping. The main memory size should exceed at least 128 MB, a size of more than 512 MB is preferable. The Central Processing Unit (CPU) must be fast due to some interpolation and arithmetic computation for graphic and scene processing – frequencies should be above 500 MHz. A swap memory should be provided with a size of about one gigabyte.

Pseudo Requirements, i.e. requirements, which are demanded by the client are also part of the analysis and will shortly be described in the following. The new graphic application mentioned above, amira, is already available and licensed at the DHM and shall be investigated, extended and tested such that it eventually can replace legacy systems for processing CT data and X-ray images. The current version of amira is 3.0. Moreover, the programming language in which all module are to be implemented will be C++ due to compatibility with amira. In fact, C++ is also preferred when realizing graphics software since the alternative, Java, is too slow for real time processing of large graphic data.

[1] `http://www.stack.nl/~dimitri/doxygen/`

3 Description of the Planning and Navigation System

Deriving all requirements from chapter 2, the system will now be described in detail. First, an overview about all hardware that was used is given in section 3.1. This includes the preoperative image acquisition step, computed tomography and the intraoperative image intensifier, the C-arm GENERAL ELECTRIC *Medical Systems OEC9800*. Regarding software, the basics and their application for processing complex graphical scenes as well as creating surface models of 3D data are explained in section 3.2. The graphical framework that implements the graphic engine for CANP will be pointed out in section 3.3 as it provides a vast functionality for medical image processing and visualization, which was partly used in this system. Then, in section 3.4, the actual design of the system will be discussed, i.e. the decomposition in subsystems as well as the interior architecture are going to be highlighted. Afterwards, the functionality of the module for preoperative planning is presented in detail in section 3.5.

3.1 Hardware Setup for CANP

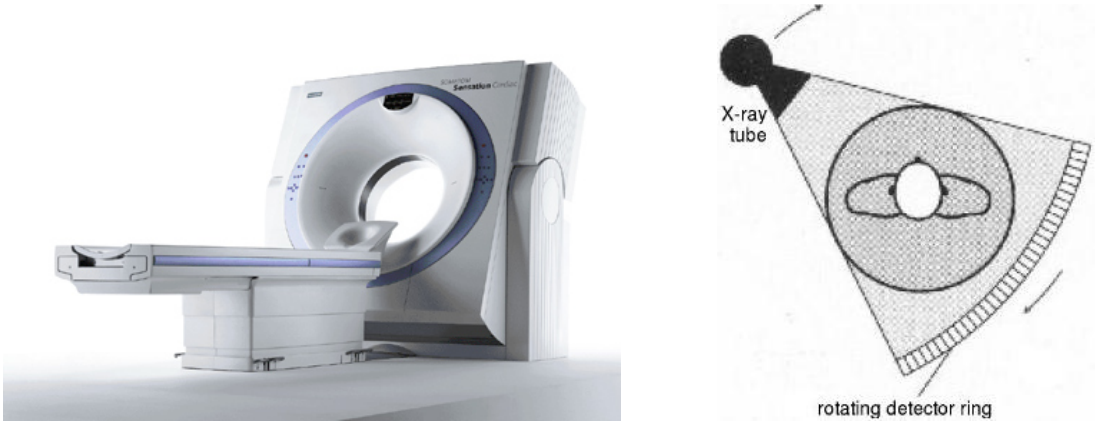
Two widespread medical image acquisition machines are used in the system, the computed tomography for preoperative as well as the C-arm shooting X-ray pictures for intraoperative imaging. Both have, as already mentioned in chapter 1 advantages and flaws. These will be described in the following.

3.1.1 Computed Tomography

The principle of computed tomography is to create cross-sectional images from projections (line integrals) of the object that is to be recorded at multiple angles. Then, a computer is used for reconstructing the image out of the raw data[1]. In X-ray CT machines, the raw data is expressed as a function describing the distribution of *linear attenuation coefficients*. These are calculated by shooting X-ray fan beams from a source tube to a receptor plate and recording the attenuation obtained from the rays' traversal of the object. The attenuation is quantified in a CT number, a *Hounsfield unit* that maps certain attenuation coefficients to different tissues. For instance, water has a Hounsfield unit of zero whereas air is assigned to -1000. Performing this step from different angles allows the computer to produce an axial slice picturing a cut through the imaged object. There are several different types or generations of CT machines based on this principle, which are distinguished by their hardware

[1]This image reconstruction must not be mixed up with a 3D reconstruction of already existant axial slices

realizations. In the following, the concept of the SIEMENS *SOMATOM Sensation Cardiac*[2] will be described. For an extensive essay about computed tomography and the SOMATOM family refer to [63].
The SOMATOM Sensation Cardiac has one X-ray tube and a large receptor plate shaped as an arc. Thus, it can record X-rays at a large range. The receptor plate and the tube are connected to each other and can be rotated around a table where the recorded object is placed, see figure 3.1. By rotating the receptor-tube-system and emitting X-rays at different points of



(a) (b)

Figure 3.1: (a) The CT scanner SOMATOM Sensation Cardiac[3], (b) the principle of third generation CT scanners, from [63]

time, the attenuation of the rays can be recorded at arbitrary angles and an axial slice can be reconstructed. Moreover, the SOMATOM Sensation Cardiac can build both, stationary axial slices and *helically* scanned images. Helical CT scanners actually translate the table with the object to be recorded during an entire rotation, i.e. the object is moved perpendicular to the plane spanned by the receptor-tube-system while rotating the system about $360°$. The advantage of helical structures is the interpolation benefit while reconstructing a 3D model. If stationary slices (using the step and shoot technique) are used, images for in-between slices can be interpolated. However, these images have lower quality than the original recorded ones. This is not the case when using helical scanning since there are no preferred slice locations, and thus, images at any slice location can be produced by interpolation, which have the same quality.
The SOMATOM sensation cardiac produces axial slices, which can be stored in the standardized DICOM format (as introduced in section 1.3.3.1) or processed directly by the proprietary SIEMENS software. This software provides standard color maps as well as the possibility to adjust the mapping of colors to Hounsfield units interactively resulting in 3D reconstructions as already shown in figure 1.3.

[2] an entire description can be found on the manufacturer's web site, `http://www.medical.siemens.com/`
[3] image from `http://www.medical.siemens.com`

3.1.2 Image Intensifier (GE MS OEC9800)

As shown in figure 3.2 the C-arm consists of an X-ray source, an image intensifier, and a CCD camera. An optical coupling system is located between image intensifier and CCD camera; image intesifier, coupling system, and CCD camera are composed in the receptor. The recorded object lies between X-ray source and receptor. When grabbing an image, X-rays are sent through the object and captured by a photocatode, which emits electrons where X-ray photons are detected. Electron lenses accelerate the electrons (intensification) which reach a phosphor plate that in turn emits light. Another reason for intensfying the image is the "minification", i.e. the useful diameter of 210 mm (where X-rays can actually produce an image) is projected to a 25 mm output window. The 25 mm image is sent through the camera lens and is captured by the CCD camera that produces a digital video signal consisting of twelve bits. A gamma curve of 0.5 is applied to the signal. The GE MS OEC9800 has three modi, distinguished by the useful diameter captured by the image intensifier, 210, 140, and 110 millimeters. The smaller the diameter, the more magnified the image. Since the output image is circular because of masking the periphery of the images, there are 950 pixels in horizontal and vertical direction. The pixel pitch is, dependent on the mode, 221, 147 or 115 microns.

The C-arm is able to shoot X-ray image sequences (for angiography), which can either be

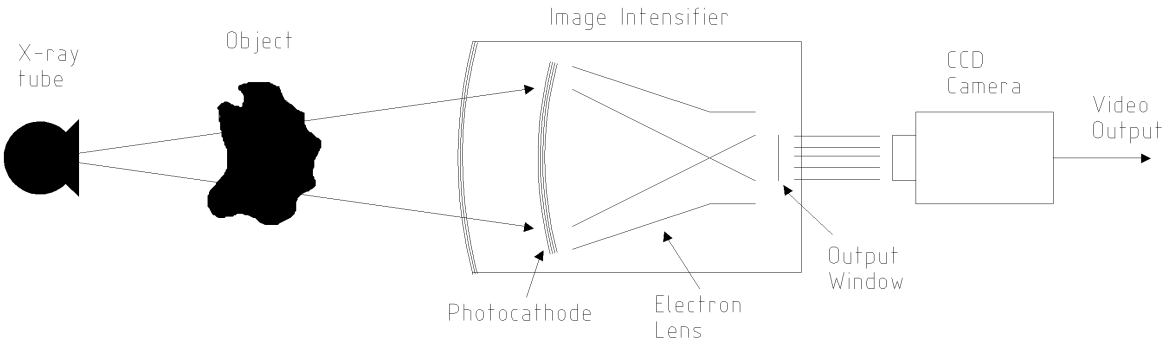


Figure 3.2: Setup of the C-arm GE MS OEC9800

stored as films or as separate images. Unfortumately, when storing a film, the proprietary .oec format is used. Thus, a further processing of image sequences is not possible. However, the data can be stored as single images in bitmap format. These images were taken for registration as described in chapter 5.

3.2 Demands on the Graphic Engine for CANP

For the system, certain requirements are to be met for visualization. It must be able to process raw CT data[4] (mostly given in the DICOM format), and have the capability of processing large data sets not only in DICOM format but also as 3D models of CT data like surface models or directly rendered volumes. Consequently, for processing and displaying 3D surface models a tool should be provided to create them, i.e. perform segmentation on

[4]Raw CT data means in this context axial slices, not, as in the previous section, linear attenuation coefficients.

CT image data and assign different color or shading properties to segments for them to be distinguished. Moreover, graphical scenes must be editable and adjustable, i.e. colormaps or virtual cameras can be altered as well as complex scenes with more than one object can be rendered. This section includes the basics of CT image processing (segmentation, see section 3.2.2) as well as a concept for dealing with complex graphical scenes (scene graphs, refer to section 3.2.1). The application of both, image segmentation and scene graphs will be described in section 3.3.

3.2.1 Scene Graphs and their Application in OPEN INVENTOR

Scene Graphs are the core of all complex graphical applications. They allow to decompose complex scenes with many objects into subcomponents and provide methodologies for processing and altering these smaller components. Thus, not only transformations can be applied to components (*nodes*) in order to change each node's pose individually, but also color and shading attributes (*materials*) can be assigned separately. Often, even cameras can be modelled virtually by a node in order to change viewing direction or position or to view a scene, e.g., perspectively or orthographically. A scene graph is built as a hierarchical tree (a directed, acyclic graph) with a root node and its children representing leaves or intermediate nodes with subnodes. In order to render a scene, the scene graph is traversed from top to bottom and left to right (thus, order is important). Methods of linear algebra allow the nodes to change their pose arbitrarily.
Among others, OPEN INVENTOR is a widespread standard for scene graph implementation. It will be shortly described, for further reading refer to [77]. There are three types of nodes in OPEN INVENTOR, *shape nodes* representing three-dimensional objects, *property nodes* modelling appearance and other qualitative properties of a scene, and *group nodes* used as intermediate nodes that allow the scene graph composition. Moreover, OPEN INVENTOR provides a vast range of basic geometric functions for calculations enabling the user to apply actions on graphs.

Shape Nodes Shapes are geometrical entities ranging from simple shapes like spheres, cylinders, or cubes to complex shapes like line or face sets and even Non-Uniform Rational B-Spline (NURBS) curves or surfaces. The rendering of those entities depends on the shape's nature as well as some geometrical properties like radius, height, edge length or simply points. For instance, creating a face set can be accomplished by providing coordinates in 3D space, an array of indices assigning each face a certain number of coordinates, as well as normals for each polygon in the set.

Property Nodes Basically, there are two kinds of property nodes. Property nodes like material, texture or labelling nodes *replace* the preliminary property nodes and assign states to all following shape nodes. Transformation property nodes have a *cumulative* character, i.e. if two transformations are adjacent, the shapes following those two nodes are transformed by both transformations. Mathematically, the two transformation matrices are multiplied and the result is applied to all following shapes. "Replacing" property nodes are, for instance, materials, draw styles or the complexity of shapes but also light models and cameras. Light models determine the scene's lighting and shading properties and cameras make the scene

visible for the user. More details about cameras, also in OPEN INVENTOR, will be given in chapter 4.

Group Nodes There are three important group nodes. The basic group node just serves as a container for an arbitrary number of subnodes and applies an order to them. When the scene graph is rendered, the group node just passes all cumulated transformations and current properties to its children. The separator node is different from the group node since it saves the current state of the graph before traversing its children. When the traversal is finished, the previous state is restored and other nodes are not affected by the properties below a separator node. Switch nodes actually just visit one of their children when traversing the subgraph. Hence, basic animation and blinking objects can be rendered.

Actions Actions are not nodes of a graph but are applied to a graph in order to affect its state. All computations are performed by actions. When rendering a graph, but also when computing the intersection of, say, a line and a plane, an action can be created to compute the result. Moreover, input/output can be accomplished via actions, i.e. writing and reading scene graphs to and from a file. Searching for specific nodes in a scene graph or assigning event handlers to nodes can be accomplished by actions. Furthermore, callback functions allow the user to apply own actions to scene graphs.

There are many implementations of OPEN INVENTOR, even some with open source code. One of the most popular open source realizations of OPEN INVENTOR is the COIN library[5]. Another realization, which is commercial can be obtained at TGS[6]. Since this system works with amira (see section 3.3), which is based on the TGS OPEN INVENTOR library, the implementation of TGS was used.

3.2.2 Image Segmentation

As shortly mentioned in 1.1 image segmentation is an important issue that has to be addressed when processing medical image data. There are many segmentation methods partly interactive, semi-automatic, or automatic. Some of them are going to be presented and a selection for suitable methods when using CT data will be given.
Basically, segmentation techniques can be partitioned into *structural techniques*, *stochastic (or pobability) techniques* and *hybrid approaches* [33]. The first class covers all algorithms that try to find structural properties like intersecting surfaces (or edges) and combine them to segments of a region. The second class performs segmentation on statistical analysis of pixel/voxel properties like attached intensity values whereas the third class is a mixture of the first two. It is rather important to choose the right segmentation technique for a certain image modality because there is no "gold standard" succeeding in segmenting all different kinds of medical images. Mostly, highly hybrid approaches, i.e. a combination of several methods are preferred in order to gain better results. In the following, a selection of approaches of all three classes will be presented.

[5] `http://www.coin3d.org`
[6] `http://www.tgs.com`

3D Edge-Detection This structural approach tries to find the contours of different segments by building a difference image and grouping all local edges together to form the boundary contours. Difference images can be created by applying a filter kernel to the original image like Robert's Cross or the Sobel filter causing the extraction of gradients. Thus, the areas where voxel/pixel intensity differ largely can be reduced to edges defining a boundary between two logical segments. In a second step, these boundaries can be grouped such that closed entities evolve, which are assigned to segments. Naturally, edge-detection works well on images with good intensity differences between regions but is quite error-prone when noise is involved. Hence, this approach is almost always combined with other segmentation methods.

Deformable models Also a structural approach, deformable models are based on a delineated region that is exposed to external and internal forces, which deform the region such that it fits to a certain segment [51]. As a preliminary step a region has to be demarcated by drawing, mostly interactively, a closed boundary around the area that is to be segmented. The demarcation is a curve, surface or solid depending on the dimensionality of the region. The external forces try to move the demarcation towards the data whereas the internal forces try to keep the model smooth during deformation. Deformable models immediately create closed curves in contrast to 3D edge-detection approaches where curves must be closed by computation. Efforts were made to apply this techhnique for segmentation of abdominal aortic aneurysms [67, 66].

Thresholding This is the simplest and most commonly used stochastic technique for image segmentation. It is based on pixel/voxel intensities. Those are charted according to their occurence, i.e. the number of pixels/voxels with a certain intensity is recorded. From this two-dimensional diagram local extrema can be depicted. The boundaries of the segments mostly lie in a valley between two maxima where thresholds are defined assigning pixels/voxels to different segments. Naturally, there is also the possibility to assign more than one threshold, which causes the region to be parted into more segments (*multi-thresholding*). A flaw of thresholding is that a certain value has to be assigned to the threshold, which is mostly user-defined. Thus, segmenting an image is a procedure of trial-and-error. Moreover, noise can cause this method to give bad results. For gaining a good segmentation of an image, thresholding is combined with at least dilation and erosion procedures to diminish "bubbles"[7] and smoothing contours.

Classifiers and Clustering Classifier methods as well as Clustering are pattern recognition techniques. Both try to partition a feature-space, i.e. a space where all the pixels'/voxels' properties are defined, derived from the image to be segmented. While Classifier methods require manually segmented training images to automatically segment image data in one step, clustering iterates between segmenting the image and characterizing the features of each class. Classifiers are functions that try to compare and classify each pixel/voxel to image elements of the training images. For instance, the *nearest neighbor classifier* tries to get the image element of the training data that is nearest to the regarded pixel/voxel and puts it into

[7] small segments within a large segment caused by noise or an inexact threshold

the same segmentation class. Clustering methods train, by iteration, themselves using available image data. For example, the *K-means* clustering algorithm iteratively computes a mean intensity for each segmentation class and segments the image by putting each pixel/voxel in the class with the closest mean. Both methods, classifiers and clustering, are stochastical approaches.

Region Growing and "Split and Merge" Being a hybrid approach, region growing extracts a connected region by evaluating a connectivity criteria on pixels/voxels surrounding a "seed point". In other words, a starting point (seed point) has to be set and a criteria has to be defined based on features of image elements. Then, the region is growed until the criteria is no longer satisfied. Like thresholding, it is hardly applied solely to image data but mostly in combination with other techniques. A disadvantage of region growing is the need for interaction placing a seed point in a region. Split and Merge algorithms do not require a seed point but a structure lain on the image data like a pyramid grid structure of regions. The structure can be split and merged according to a criteria as in region growing.

Atlas-Guided Approaches Here, a preliminary segmentation of certain anatomy is performed and stored in a database. These segments are called atlas. When segmenting a new image, it is adjusted to the atlas and the region that fits is extracted. Actually, this technique is derived from registration (see chapter 5) performing transformations on images to align atlas and the image to be registered. As a matter of fact, an atlas-guided approach was applied to the detection of already implanted stents in angiographic images [31].

3.3 The Graphic Engine of CANP: amira

As defined earlier in section 2.5, amira (version 3.0) is chosen as the graphical core for the implementation of the planning and navigation tool. amira is able to visualize and reconstruct three-dimensional data from resulting data of acquiring methods such as computed tomography, magnetic resonance imaging, and ultrasound scans, which are all common in the medical domain. For instance, amira can load DICOM files and convert them to a proprietary format, which serves as a base for any further processing such as displaying and computing. In this step metrics are preserved. Additionally, amira is capable of importing and exporting three-dimensional file formats like OPEN INVENTOR or Virtual Reality Modeling Language (VRML). Especially large data sets are visualized with satisfying speed. Unfortunately, loading these sets requires an enormous amount of RAM (Random Access Memory), leading to unpredictable problems.[8]

Scene Graphs in amira amira can handle scene graphs built with OPEN INVENTOR. It provides viewers, each one having an own scene graph, arranged in (separate) windows where scenes can be rotated, translated or scaled. Also single objects are transformable in respect to others in the scene. Thus, amira can actively alter camera or transformation settings

[8] In the proposed system amira was crashing sometimes while loading DICOM data having a size of more than 700 megabytes, although the total amount of RAM was 1.5 gigabytes. This was tested on Debian Linux.

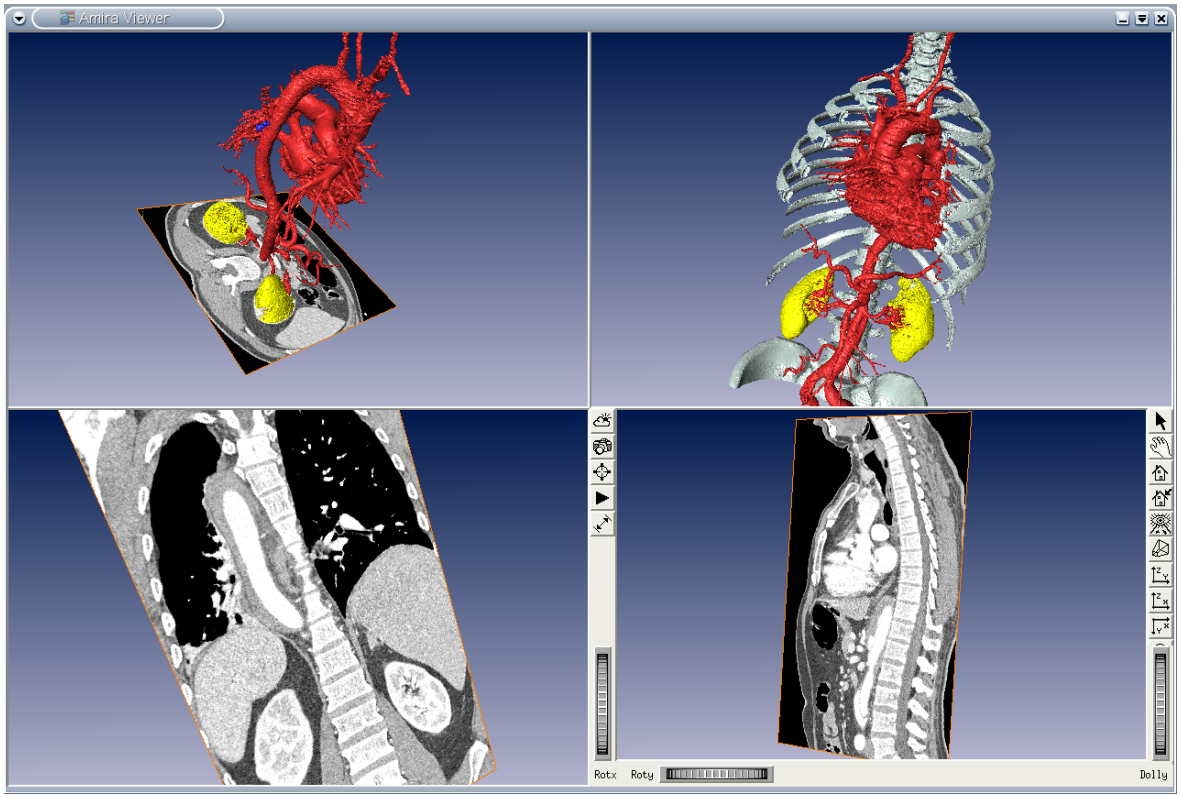


Figure 3.3: Four viewers showing different objects of a human thorax. The upper left viewer shows a 3D surface model and axial slices. The upper right viewer shows the entire 3D surface model with bones. The other viewers show frontal and sagittal slices respectively.

in scene graphs. The viewers as shown in figure 3.3 provide additional functionality like a switch between browse mode, i.e.manipulating cameras or draggers but no objects that are displayed in the scene, or interactive mode, i.e. altering the objects in the scene and adding or removing nodes. A nice feature is the snapshot tool giving a snapshot of the actual viewer image and storing it in bitmap format.

Image segmentation in amira Segmentation in amira always starts with a (multi-) thresholding of volumetric voxel-data. Then, for increasing the quality *bubbles*, i.e. small segments created by noise can be deleted and the segments' boundaries can be smoothed using dilation and erosion methods. Moreover, amira offers a segmentation editor that can, additionally to thresholding, segment all voxel-based volumetric data providing tools for region growing or interpolation as well as algorithms for deformable model segmentation like *snakes* [29] and methods for detecting edges. Figure 3.4 shows the segmentation editor with four viewers, the upper right and the two lower ones showing axial, sagittal and frontal slices whereas the upper left viewer displays the segments that are selected currently. A selection of segments can be performed partly interactive and partly automatic and can then be assigned to a certain material. Either one slice or the whole volume may be segmented.

Besides scene graph manipulation and image segmentation, amira provides lots of function-

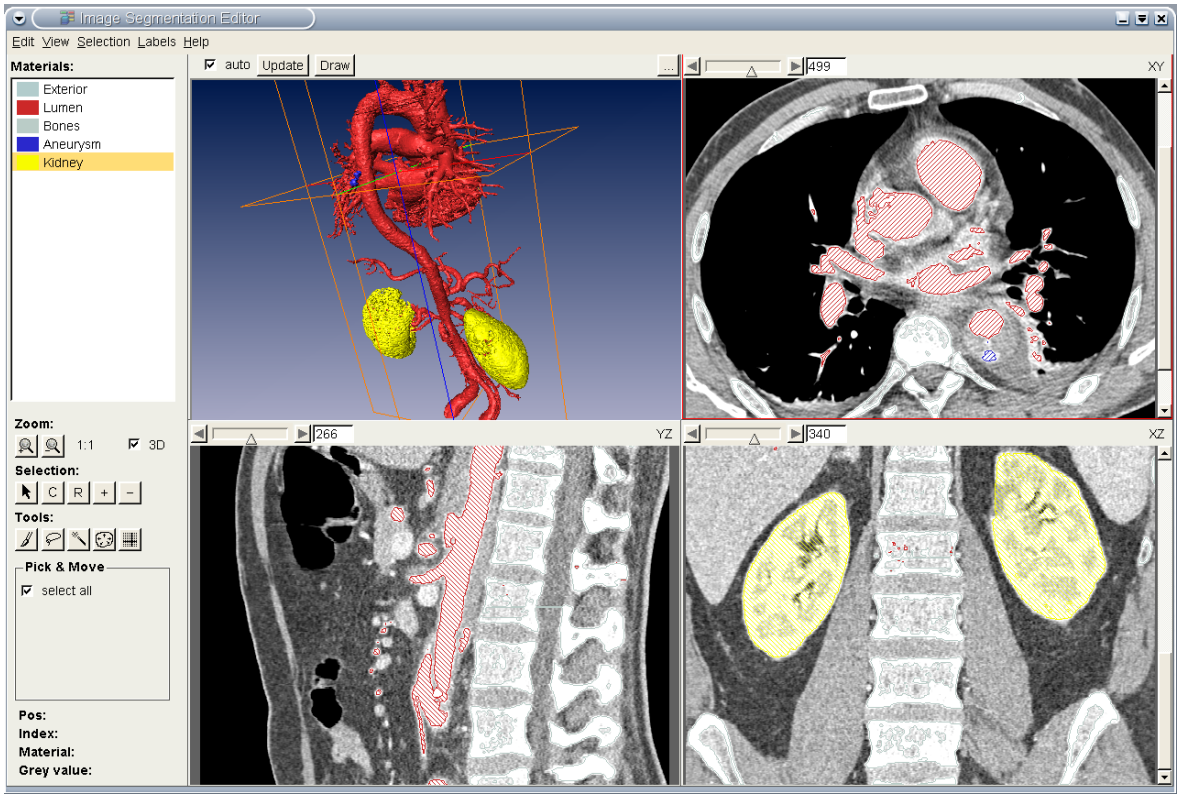


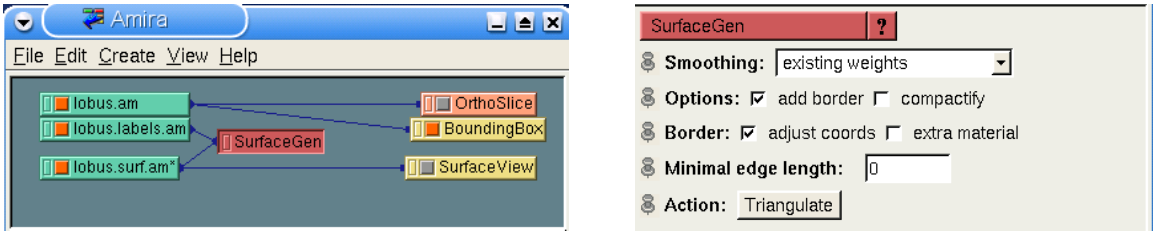
Figure 3.4: Segmentation editor of amira. The upper left viewer shows a 3D model of the selected segments, the other viewers show axial, frontal and sagittal slices respectively with selected segments of the aorta/heart as well as the kidney region.

ality. Some of it will be presented in the following after the general philosophy of amira, modules, was described.

3.3.1 Modules in amira

The functionality of amira relies on its main components, divided into modules and data objects. The latter are visualized by display modules. However, besides visualizing it, some modules are designed to perform computations on the data.
In order to make use of modules and data objects they are generated and placed into the so-called object pool, their central management repository. Inside the object pool they are represented by colored icons. As shown in figure 3.5(a) there are separate colors for data objects, display modules, and compute modules, respectively. If modules or data objects are interrelated in any way, their corresponding icons are connected by lines via ports. Moreover, figure 3.5(b) shows ports that allow the user to interact with amira's modules by providing adequate graphical user interfaces (see section 3.3.2).

In the following sections a selection of amira modules and data that are crucial to the system is worked out in detail. Not only the structure but also the functionality of each module will be described. An extensive description can be found in amira's User's Guide and Reference Manual [32]. Figure 3.6 shows the top of amira's class hierarchy. The class



(a) Modules in the object pool of amira. Yellow and orange objects are display, green are data, and purple are compute modules

(b) Ports for user interaction in amira

Figure 3.5: Object pool and port structure of amira

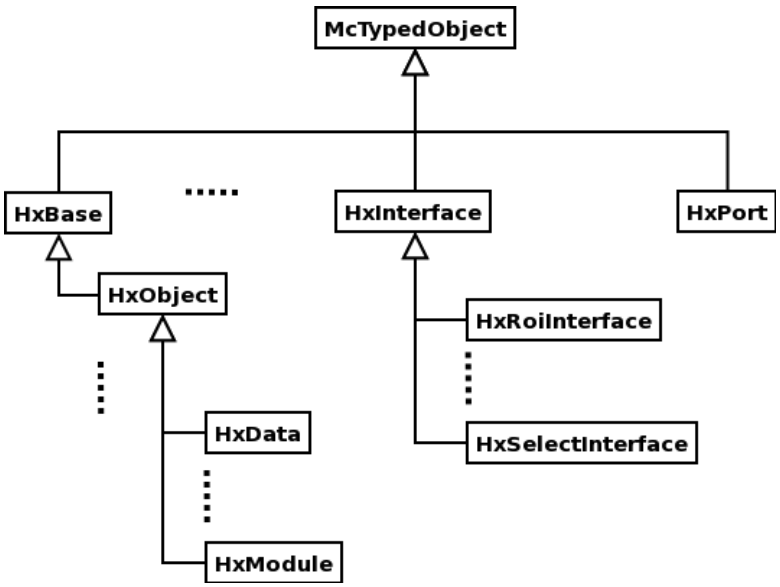


Figure 3.6: The top layers of amira's class hierarchy.

`McTypedObject` provides a typing facility, which, in particular, enables programmers to perform type-checking at runtime. Two other important classes are the directly derived `HxBase` and `HxPort`. `HxBase` is the base class of all objects in amira, except the user interaction classes, which all derive from `HxPort`. Apart from this hierarchical object-oriented approach, amira also offers some global objects out of performance reasons[9]. For printing to a Tcl (Tool Command Language) console provided by amira, the global variable `theMsg` instantiated from `HxMessage` can be used offering a `C`-like `printf(...)` syntax. Moreover, in order to gain access to all objects of the current configuration of amira lying in the object pool mentioned above, an instance of `HxObjectPool` is given, `theObjectPool`. `theWorkArea` is an object of the class `HxWorkArea` and grants manipulative access to the section in the amira window where all ports and the progress bar are displayed. Most importantly, controlling amira viewers and thus scene graphs is provided by `theController`, an

[9]Only those global objects that are relevant to the system are introduced

object of `HxController`. With this instance, new viewers can be created, and scene graphs can be edited, manipulated and saved. amira offers the object `theInterpreter` of class `HxInterpreter` for executing Tcl commands via the C++ interface. Since amira manages the memory, in particular the garbage collection automatically, i.e. does not allow to delete formerly created objects from memory, `theInterpreter` is crucial for cleaning up the object pool as well as changing values at ports without using the GUI. For more information about Tcl and `theInterpreter` refer to section 3.4.2. Besides these global objects, amira strictly sticks to the object-oriented paradigm, which will be described in the following.

3.3.1.1 Data Objects

Data Objects actually represent different *file formats*, which are converted to certain proprietary *data types* by amira. Besides importing procedures for a vast range of different well-known graphic formats, amira also offers processing and display methods for all available data types. Known file formats vary from JPEG (Joint Photographic Experts Group), GIF (Graphic Interchange Format), or BMP (Bitmap) to standard formats from different areas in science, like the medical DICOM format, the Bio-Rad Confocal Format for biological microscopy, or the DXF (Drawing Exchange Format) format used by CAD (Computer Aided Design) applications. They are converted into types like scalar or vector fields and line or vertex sets. Since amira features many different data types and file formats, only those relevant to the system are presented.

Data Types In amira all data types derive from the superclass `HxData`, which implements generic load and save functions. Three-dimensional data all derives from a subclass of `HxData`, `HxSpatialData`, which mainly divides up into fields (like vector or scalar fields) and point sets (representing a set of three-dimensional points in space that can be manipulated). Hence, the two classes `HxField3` and `HxVertexSet`. Figure 3.7 shows the classes of interest inheriting from `HxData`, `HxField3`, or `HxSpatialData` respectively. Below, they will be highlighted shortly.

- The *ColorField3* data type is a 3D field with four entries (the colors red, green, blue, and an alpha value (opacity)) for each voxel. Usually, it is used in combination with direct volume rendering for directly mapping color values to voxels without any intermediate transfer function. However, a bitmap image, for instance, can also be converted to a *ColorField3* and viewed via an *OrthoSlice* module described in the next section.
- A *Colormap* stores 256 different RGBA-tupels (again, red, green, blue, and alpha value) and two coordinates defining the range used for color interpolation. *Colormap*s can be used to attach them to, say, a DICOM slice stack mapping all voxel values, e.g., Hounsfield numbers, (which are mostly much more than 256) to 256 grey values.
- Typically, DICOM data is stored in a *UniformScalarField*, which represents a field in $\mathbb{R}^3$ using scalar values, not vectors to describe an object in 3D space in a uniform lattice. A field can be accessed at any point in a 3D domain unlike discrete scalar arrays.

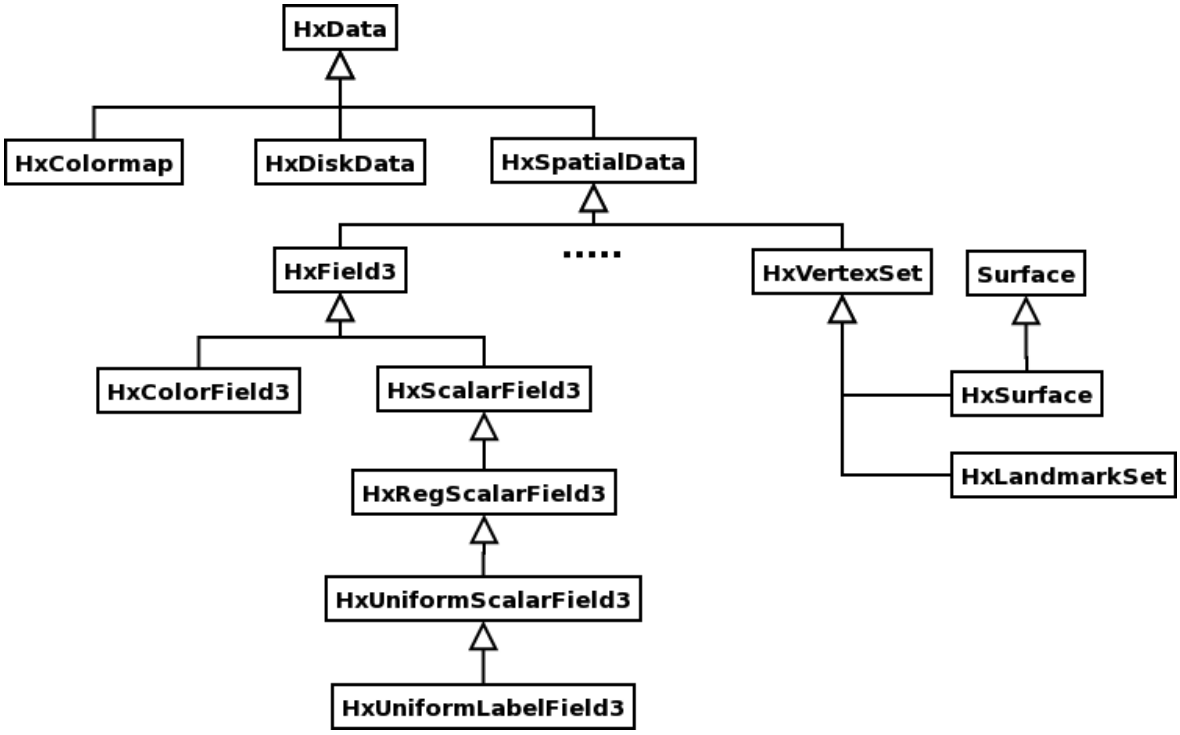


Figure 3.7: The class `HxData` and its derived classes.

- Labelled data like segmentation results can be expressed via the *LabelField3* data type. Prerequisite for labelling is volumetric data containing voxels. A label is assigned to each of these voxels indicating the region they belong to.
- The *LandmarkSet* data type describes specific points or markers in 3D space. This is actually most important for aligning (registering) data. amira already provides methods for registering two 3D data sets, like two surfaces by setting landmarks. Landmarks can be set interactively in a viewer window. When setting a landmark amira automatically performs a collision detection and attaches the marker to the first object found along a line parallel to the viewing direction passing through the landmark.
- With the *LargeDiskData* type (from class *HxDiskData*) amira allows to load large image data that does not fit entirely into random access memory. This is accomplished by loading subvolumes as a field object.
- Finally, *Surface* data is used to describe triangulated surfaces. Besides 3D coordinates identifying the triangles' vertices, additional information can be stored, like materials, patches, contours, edges etc. Patches actually part a *Surface* into several sub-surfaces. Each patch has two sides, also referred to as inner and outer regions. To all regions materials are assigned. Contours define the boundary of a patch.

File formats Most file formats for graphical processing but also for medical, biological or physical purposes can be imported and exported by amira.

- *AmiraMesh*: In order to standardize file in- and output, amira offers a general-purpose format called *AmiraMesh*. It can store many different data types like vertex sets, segmentation results or colormaps. Moreover, some formats are, when loaded, automatically converted to *AmiraMesh* objects, e.g. DICOM slices. The flexibility of *AmiraMesh* is realized by saving arbitrary multi-dimensional arrays into a file. As a matter of fact, all data types but *Surface*s can be stored in *AmiraMesh* files.

- *DICOM*: For loading DICOM data amira offers a straightforward file dialog. After loading the slices into RAM, amira automatically sorts slices and detects different stacks by evaluating, for instance, slice location or image number. The stacks can then be stored as different data objects and processed. If DICOM information is missing or corrupt, e.g. slices are set with wrong locations or numbers, stack criterias can be altered or turned off. Moreover, all the data available from the extensive DICOM header can be viewed while loading the data.

- `HxSurface`: *Surface* data is ex- and imported by the file format `HxSurface`. All information from coordinates to regions and contours can be stored using this file format. There is a simplified version of this format just saving or loading materials, coordinates and patches. All further information can be computed. The extended version also stores additional topological information like regions, boundary curves or branching points.

3.3.1.2 Display Modules

Display modules visualize data given by data objects. There is a tight correlation between the scene graph methodology described in 3.2.1 and the display modules. Figure 3.8 shows the common superclass of all (display) modules, `HxModule`. Only those classes that are presented below are charted.

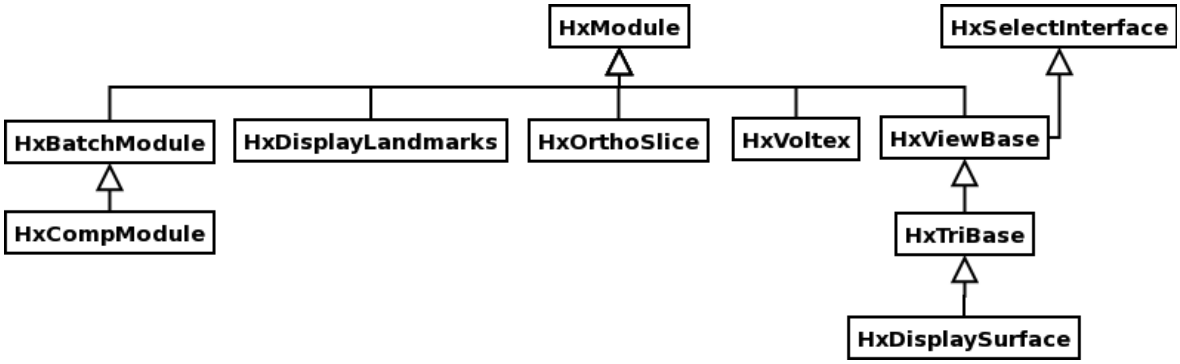


Figure 3.8: The fundamental class `HxModule`, which is most important for developing own modules.

- *HxDisplaySurface* modules visualize *Surface* data. Several parameters can be adjusted providing many visualization techniques. Drawing varies from pointed and lined to shaded and transparent styles. Materials can be added and removed in order to utilize the possibilities of image segmentation to their full extent. An important issue for the

quality of modules visualizing surface data is their efficiency. Especially rotations and translations should be performed in real time. Large data sets with more than, say, two million triangles can be transformed fluently with the *HxDisplaySurface* module in shaded draw style mode. This is due to the combination of efficient storage of surface data via the *Surface* data class and successful hidden surface removal provided by OPEN INVENTOR, all capsuled by the class `SoTriSurface`. When it comes to transparency, however, the performance of surface transformations is becoming rather bad. *HxDisplaySurface* derives from *HxViewBase*, which is providing all basic ports for viewing the *Surface* data. The data is actually represented as a collection of nodes in an OPEN INVENTOR scene graph, for instance as a *SoTriSurface* node along with attribute nodes defining the material or draw style. The selection of materials in order to show or hide their corresponding triangles is implemented in the *HxSelectInterface* class, which is itself a parent of *HxViewBase*. It serves as an interface for individually picking a set of elements such as triangles, points, lines, and more.

- *HxDisplayLandmarks* modules are quite straightforward since they just have to visualize a set of points in 3D space (landmarks or markers). However, amira provides different kinds of displays – whatever is most suitable. Either spheres or points (small squares) can be selected and their size can be adjusted. Furthermore, one can change the level of detail of a sphere in order to enlarge or reduce the memory space occupied by landmarks. Since *LandmarkSet*s can model not only one but many point sets, there is also the possibility to choose which set(s) the display should show.

- The *HxOrthoSlice* module provides a cut through volumetric data. As it's name already suggests, the cuts are orthogonal, i.e. three cuts for xy-, xz-, and yz-plane can be shown that are perpendicular to each other. Those cuts are named after their pendants in medicine – axial, sagittal, and coronal slice directions. Given a volumetric data set, slices can be browsed through in each direction, i.e. the adjacent slices are shown when moving a slider.

- The *HxVoltex* module creates an object in 3D space from volumetric data using direct volume rendering. It's performance is quite good, if 2D, ideally 3D texture mapping is supported by the graphics hardware. If no hardware acceleration is provided by the graphics hardware, the module can actually perform its computation software-based. However, this results in an immense loss of performance. In this case, *HxVoltex* provides a downsampling of data causing the volume to reduce the number of voxels to be rendered and thus speeding up computation again.

3.3.1.3 Compute Modules

Compute Modules perform calculations on data objects. amira provides a vast range of modules whose computation methods differ from segmenting volumes via thresholds or generating an isosurface via cube-marching to scaling volumes or applying filters like Sobel or Robert's Cross to images. As can be extracted from figure 3.8 compute modules do not directly inherit from `HxModule` but from two intermediate classes, `HxBatchModule` and `HxCompModule`. The first class actually provides a threading concept for all derived compute modules. Hence, if computations are very time-consuming, they can be submitted to

background and the user can proceed to work with amira. The second class is the base class for all compute modules and offers some useful generic functions. Figure 3.9 enlists all the compute modules that were consulted in the system.

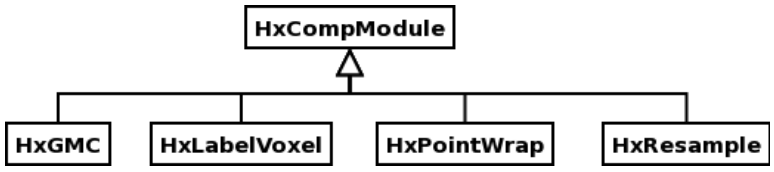


Figure 3.9: The basic class `HxCompModule` and a selection of important compute modules. `HxGMC` is the name of the *SurfaceGen* module.

- The *LabelVoxel* module can create a *LabelField* out of volumetric data such as a *UniformScalarField* by performing thresholding. In fact, even multi-thresholding is provided computing up to five labels that can be edited and reworked with a segmentation editor. For finding minima and placing thresholds in histograms, a histogram editor is provided in this module showing the number of voxels that occur at a certain greyscale.
- A very useful module is the *Resample* module that allows to up- or downsample 3D fields. Often, a downsampling of large data sets is necessary to speed up segmentation and visualization processes. Moreover, the *Resample* module can scale voxel sizes to cubes if required. To non-labeled data fields different filter kernels can be applied like Lanczos, Mitchell, or B-Spline filter. Labeled fields can only be downsampled, not enlarged.
- *SurfaceGen* computes a triangle approximation of a surface described by a *LabelField*. Mostly, when applying this module to a *LabelField* directly created with *LabelVoxel* the resulting surface will have a huge amount of triangles. Hence, resampling the *LabelField* in advance is recommended.
- The *PointWrap* module computes a triangle surface out of a set of points. The problem when attempting to form a surface out of an unordered set of points is how to create triangles. Each point belongs to three triangles for a closed surface, at least to one triangle for an open surface. Moreover, every three points that are closest to each other should be formed to a triangle. The *PointWrap* module meets these requirements by creating a virtual sphere of an arbitrary radius, 'dropping' it down an axis (of the coordinate system the points are set in) until the point set is hit and 'rolling' it over such that all points in the set are covered. If no initial triangle can be found, the 'drop' axis can be changed or the radius can be enlarged.

3.3.2 Communication in amira – Ports

In amira the only way of graphical interaction between users and objects is using ports. Ports provide the functionality and user interface to change, store, or receive parameters as well as to start computations and function calls. Also, if two objects want to communicate with each other they have to use ports. For instance, a *Resample* module gets the data to be resampled

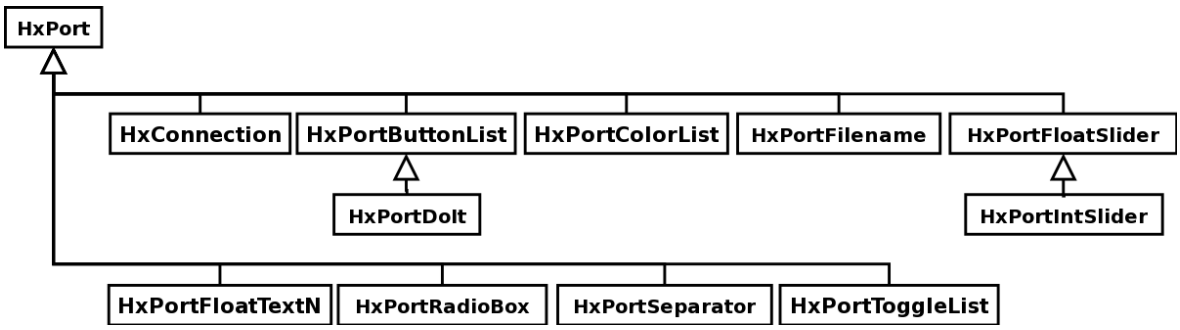


Figure 3.10: The parent class `HxPort` and its subclasses providing several ways for interaction.

via a *HxConnection*, which is a port. The general design and some important ports will be presented in the following.

- *HxPort*: This is the abstract superclass of all ports. It provides generic functionality that all ports have in common. For instance, a function determines whether a port has been modified by an extern object or a user.
- *HxConnection*: This is a special port describing a connection between two objects. Naturally, data is exchanged via this connection. This port, unlike others does not provide its own user interface. However, it is drawn in the object pool as a mere line connecting two objects.
- Besides those two elementary ports, there are several others for object and user interaction providing a vast range of functionality and GUI, like button lists, radio and toggle boxes, or integer and float sliders as shown in figure 3.10.

Having described existent hard- and software of the system, the following section highlights the design of additional subsystems, modules and classes.

3.4 System Design for CANP

Decomposing the system into subsystems led to three different modules having minimal coupling and maximal coherence[10]. First, a framework with tools for camera calibration and image registration was conceived. Chapter 4 and 5 will describe how the navigation tool is realized by an overlay of images. For now, it should just be pointed out that for successful navigation X-ray and CT data have to be registered, which is accomplished by creating an artificial X-ray image out of CT data, a DRR as described above, and viewing it from the same perspective as the real X-ray image, which in turn makes it necessary to calibrate the C-arm the real X-ray image is recorded with. Then, a transformation can be applied turning the DRR into the real X-ray image an thus aligning (registering) X-ray and CT data. The calibration and registration framework was designed to be entirely independent from the amira

[10]Coupling is the generation of associations or dependencies between subsystems whereas coherence expresses the dependencies inside a subsytem.

framework. However, subclasses deriving from the registration classes in this framework were glued to amira in order to utilize the vast range of functionality it provides. The second and third subsystem consist of modules for planning and navigation, which were integrated in amira obeying the specific amira class hierarchy. Consequently, the navigation module used the methods and results from the calibration and registration framework in order to align the two data sets, X-ray and CT.

3.4.1 Framework for Calibration and Registration

The framework for calibration and registration is shown in figure 3.11. It consists of a mainly abstract superclass `CcTransformation` that inherits to two subclasses `CcCalibration`, which can calibrate cameras as described in chapter 4, and `CcRigidRegistration`, which is able to register two images by performing a rigid transformation. `CcTransformation` includes the method that registration and calibration use both, `estimateTransRot()`, determining a translation and a rotation as will be described in chapters 4 and 5. In the registration part, the application was separated from the implementation. Thus, the method for aquiring images that are to be registered is capsuled in the class `CcDRRFactory`, which is responsible for creation of Digitally Reconstructed Radiographs. This is just one way of image aquisition for registration and can be used in other fields addressing problems different from registrational ones as well. `CcRigidRegistration` and `CcDRRFactory` are superclasses for the classes `HxAutoRegistration` and `HxDRRFactory`, which actually realize the automatic registration using modules from amira. The separation of application and solution (implementation) is commonly known as the *Bridge Design Pattern*[18]. For calibration, two different implementations are provided, both using different calibration algorithms. Furthermore, `CcParamFile` allows persistent data storage of computed parameters for calibration and `CcTypes` provides some structs for mathematical entities like lines, points in 3D or 2D space, etc. The framework does not offer any graphical user interface, just a main method with a number of targets described in appendix C.

3.4.2 Extending the Graphical Core: amiraDev

The calibration and registration module was independent from amira, however, the other two modules, one for planning and one for navigational display and controlling, were integrated into the amira class hierarchy. amira has been made extensible by providing a developer version, amiraDev, which will be described in the following.

The amira Developer Edition, amiraDev , extends and customizes the functionality of its core, amira 3.0. amiraDev allows for creating additional module and data classes such as modules for visualizing or processing data and read or write routines. According to object-oriented methodologies these additional module or data classes can inherit from existing ones extending their functionality and are used in the basic amira edition again. A modification or change of existing classes is not possible though.
Basically, amiraDev is a collection of all the C++ header files needed to compile modules. As a matter of fact, not all the modules' header files are provided resulting in a very limited extensibility. However, most of the functionality of amira can be queried by a built-in Tcl

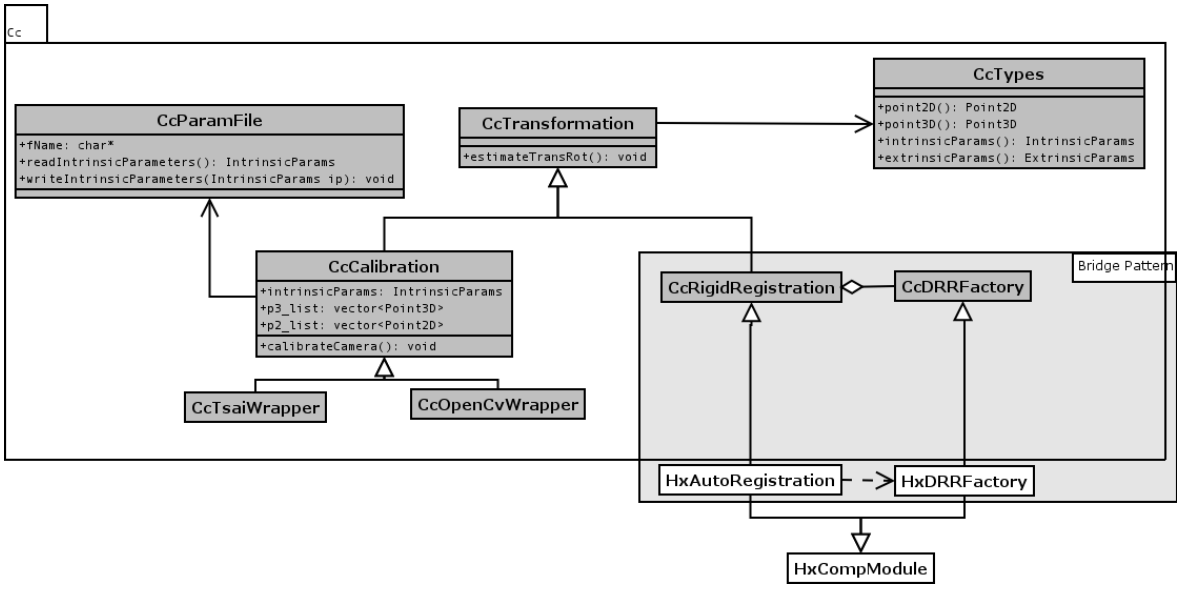


Figure 3.11: Class diagram for the calibration and registration framework

interpreter. Tcl is a scripting language library written in C. For convenience reasons, the presented system offers a C++ Tcl interpreter wrapper class that is able to convert C++ function calls into Tcl commands as an add-on to amiraDev (see section 3.4.2.1). amiraDev is not only using Tcl, but also other platform independent industry standards such as OPEN INVENTOR, OPENGL, and QT. amira as well as amiraDev is built on top of OPEN INVENTOR that is again based on OPENGL, which is a two-dimensional and three-dimensional graphics application programming interface (API). OPEN INVENTOR itself simplifies the access to and interaction with OPENGL, being the de facto object-oriented standard toolkit for complex three-dimensional visualization applications as described above. QT is another toolkit for the C++ development of graphical user interfaces (GUIs) and applications.
Using all these standards, amiraDev and amira, respectively, aim to remain platform independent as well. SGI IRIX, HP-UX, Sun Solaris, Linux, and Windows are officially supported development platforms.

To develop own components such as read or write routines for data classes and display or compute modules, respectively, a so-called development wizard can be used that provides a choice of steps to create own components. Every component is a member of a certain package, which is a shared object that can be dynamically loaded when requested by the runtime environment. The technique of providing shared objects enables the programmer to extend amira's core without recompilation of the main program. Additionally, the system only requires to allocate a minimal amount of memory.
By completing all steps demanded by the development wizard the framework for a new module is automatically created, which is ready for compilation.

Whereas read and write routines are either static member functions of an arbitrary class or global functions, modules are always inherited classes. Own compute modules derive from `HxCompModule`, all other kinds of ordinary modules such as display modules derive from `HxModule`. Usually, display modules are attached to data objects, which encapsulate any

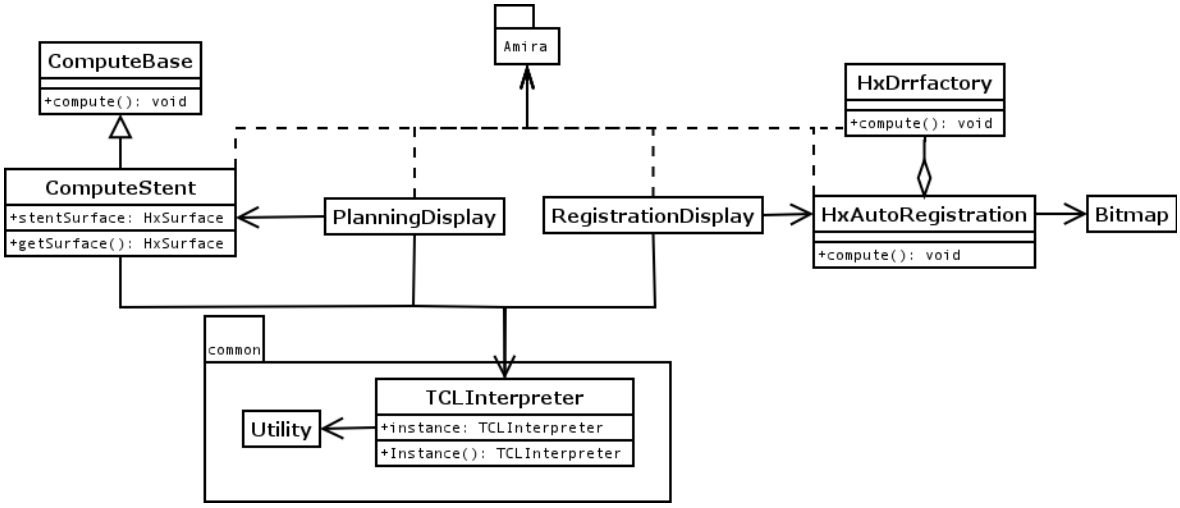


Figure 3.12: Class diagram for planning and navigation

files that are loaded into amira . For instance, a `HxOrthoSlice` display module that is capable of visualizing computed tomography scans is attached to a `HxUniformScalarField3` data object, which is amira's internal representation of DICOM files.
It is, however, quite difficult to separate computation methods from displaying methods in order to guarantee amira's module concept. Sometimes *Doit*-buttons emerge in a display module's user interface intitializing a certain computation, i.e. cube marching as in *isosurfaces*. Thus, computations are performed without using compute modules. That is why even ordinary modules have a `compute()` method derived from `HxModule`. This fact has to be considered by extension programmers.

One important issue about amira is the saving procedure. All objects visible in the object pool have (or use) save functions that allow to store a number of classes, i.e. their current state to a file. This is accomplished by writing creational and port-specific Tcl commands into this file, which is called a network. When loading this network into amira the Tcl commands instantiate classes and initialize their ports. Thus, the former state can be restored. Sometimes a parameter is not expressed via a port. This can lead to inconsistent states since it is, if no extra Tcl function is provided, not stored when saving a network. Hence, an amira developer must either extend amira's Tcl interface or avoid using parameters without ports.

3.4.2.1 Preoperative Planning Module

Figure 3.12 shows the classes conceived for the planning as well as the navigation module. A class, which both modules have in common is the class `TCLInterpreter`, which offers a C++ wrapper that allows to call Tcl functions via a C++ API. This was necessary since first amira manages object creation and deletion itself and objects can thus just be created or removed by calling a Tcl procedure. Second, the amiraDev edition does not include the header files of all modules available in amira, just a small selection. In particular, most of the compute modules' header files have been left out. Thus, the only way of accessing these classes is via the Tcl interface. Since more than one class addresses the `TCLInterpreter` class, the *Singleton Design Pattern* has been implemented ensuring the existence of at most

one instance of `TCLInterpreter`. The class `Utility` provides methods for string manipulation used by the interpreter class. Numbers can be converted to strings in this class and vice versa.
For planning, the classes `ComputeBase`, `ComputeStent`, and `PlanningDisplay` are provided. `ComputeBase` serves as an abstract superclass for all possible implementations that can be implemented for creation of a stent graphic. One realization is offered in this system, capsuled in the class `ComputeStent`. This class provides the possibility to create the amira class `HxPointWrap` described above and invoke or iterate on its methods. The class `PlanningDisplay` offers all ports and creates all objects that are necessary for planning the stent. For instance, an object of the class `HxLandmarkSet` is utilized by `PlanningDisplay`. For an extensive explanation of the functionality and procedures of the planning tool refer to section 3.5.

3.4.2.2 Intraoperative Navigation Module

As mentioned before, the navigation tool offers a registration of images by alignment of real X-ray and DRR images. This can be accomplished interactively or automatically. The interactive part is capsuled within the user interface class `RegistrationDisplay`. It also offers methods for back-projection as will be described in chapter 5. Again, those amira classes whose header files are not available are accessed via an instance of the class `TCLInterpreter`. For automatic alignment the class `HxAutoRegistration` was implemented aggregated to the class `HxDRRfactory` that is handling the creation of DRRs, i.e. making a snapshot of a DRR volume with a specific transformation. A `Bitmap` class offered functionality to process bitmap files since both, X-ray image and DRR are stored in the .bmp format. The functionality of this modules will be presented in chapter 5.

3.5 Planning the Stent Implantation

As displayed in figure 3.13, the planning module provides a four-viewer-system showing all available information about the CT data. Assuming the best case, the upper left viewer shows a 3D surface model of the segmented CT data, and the upper right shows axial slices whereas the two lower viewers shows CT slices in frontal and axial direction. However, if some data cannot be provided, in particular the 3D surface model is missing, the system is reduced to three-viewer-mode enabling the physician to plan just on slice data. Since preoperative CT scans are obligatory for stent implantations at the DHM axial slices are always present and frontal and sagittal slices can easily be computed from them. The user interface as shown in figure 3.14 provides several ports for information and user interaction. With the **mode** port the way of interacting with the system can be chosen. In view mode the cameras of all viewers can be changed, i.e. slices as well as 3D surface model can be translated, scaled, and rotated. The interactive mode allows the user to browse through the slices in axial, sagittal or frontal directions, locating the aorta and, in particular, the aneurysm.

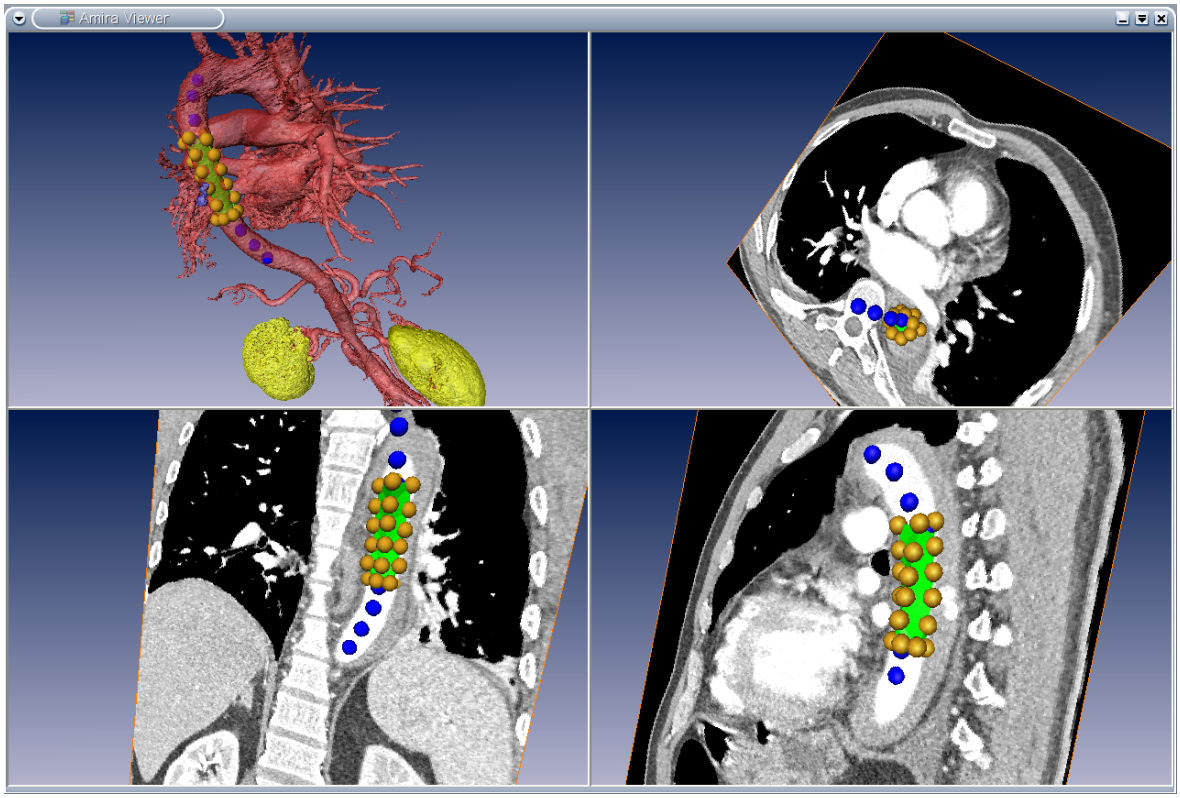


Figure 3.13: Planning mode of CANP – the golden spheres are landmarks for stent, the blue spheres are landmarks for marking the run of the aorta and the green surface is the planned stent as computed by CANP.

3.5.1 Placing Landmarks

The Stent mode can be chosen when the user determined the region of interest, i.e. the section of the aorta the stent is to be placed. Referring to figure 3.13, this mode provides the possibility to place markers – small golden spheres realized with the class `HxLandmarkSet` – around the aneurysm in order to confine the stent's measures. Naturally, those markers can, when already set, be moved or removed by changing the entry in port **Edit mode**. The Aorta mode can be chosen for marking the run of the aorta. Especially in frontal and sagittal slices large pieces of the aorta region can be visualized in one slice which makes it easy to place landmarks there. Marking the aorta is necessary to enable the system to accurately back-project the current position of the stent in the X-ray image to the 3D surface model during operation. This additional effort was accepted due to the benefit gained for intraoperative navigation: after back-projection, a line set was created from these markers and the nearest location to the back-projected ray was determined. Thus, expensive image processing for locating the aorta could be avoided and intraoperative navigation could be sped up. The markers of this set were displayed with a blue color in order to distinguish them from the stent landmarks. Port **ViewLandmarks** offers two toggle buttons hiding or showing stent markers or aorta markers or both in the case of mutual occlusion.

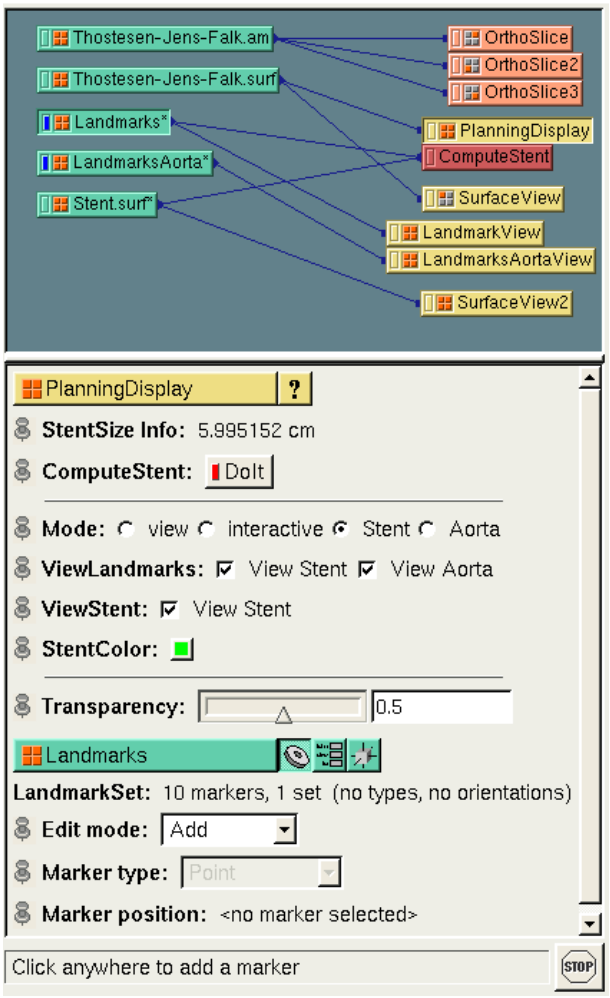


Figure 3.14: User interface for the planning module

3.5.2 Computing the Stent Graphic

When all stent markers are set[11], the DoIt button of port **ComputeStent** can be pressed invoking the computation of an approximate stent graphic. For this, the compute module `HxPointWrap` is used, trying to create a directed and connected surface out of a set of points by dropping and rolling a sphere over the scene as described in section 3.3.1.3. The set of points is taken from the stent markers placed by the user before. Actually, the `HxPointWrap` module tries to create a surface dropping the sphere with a specific radius from a particular axis. However, this is not the best result in most cases or does not yield a result at all. This fact is considered by the `ComputeStent` class by performing an iteration over the method used in `HxPointWrap` to compute a surface. Three iterations are executed, one for each axis. The running variable is a float value expressing the radius of the sphere starting from 0.1 centimeter to the maximum value of the port with a step size of 0.1 cm in each iteration. The `HxPointWrap` module provides after every computation the number of faces that could

[11] aorta markers are independent from the stent landmarks and thus can be placed before or after the computation of a stent graphic

be created in the surface. If this number increases an optimal surface is replaced with the new one, naturally.

3.5.3 Altering and Saving the Stent Graphic

After the computation, a new surface is added in the object pool and displayed in all viewers. Additionally, the size of the stent is displayed in centimeters at port **StentSize**. It is determined by measuring the dimensions of the bounding box of the surface that was created. The color of this stent surface can be changed via the port **StentColor**. Moreover, a **Transparency** port emerges enabling the user to make the 3D surface model transparent such that the stent graphic, which, if planned properly, lies inside the aorta, is also visible in this scene. Keep in mind that the preformance decreases when making a model transparent due to the lack of hidden surfaces that could be removed in opacity mode. Thus, transparent models impact the fluency of viewport changes, i.e. transformations of the camera. If the user is not satisfied with the placement of the stent graphic, the landmarks can be moved – for this, the stent graphic can be hidden using port **ViewStent** – or removed and new markers can be added as well. A further computation of the stent graphic generates a new surface based on the currently set markers. When the user is content with her stent placement, the whole scene, in amira terms network, can be saved and thus the stent graphic can be stored persistently in order to make it available for intraoperative navigation. All ports' values are stored in a file with Tcl commands as well as creational instructions for the modules present in the object pool. This arose problems because the `PlanningDisplay` module inherently created the modules `HxOrthoSlice`, `HxDisplaySurface`, and `HxLandmarks` as well as `ComputeStent` created a `HxPointWrap` module. Naturally, when a network was loaded, these modules would have been created twice. Thus, an additional Tcl command was introduced, setting a flag *loadedFromNetwork* when saving the network, indicating that the network was created by executing a network script. With this, the second creation of all these modules could be avoided.

3.5.4 Transferring Preoperative Planning to Intraoperative Navigation

As already mentioned, the network could be restored entirely after it was saved. This network, however, did load all modules involved in the planning procedure and thus was not suitable for intraoperative navigation where only the data objects (slices, surface model, and stent graphic) are important. Since a new surface was created during planning, a .surf file was generated storing the stent graphic. It could be loaded easily, as well as the .am file for slice data and the .surf file for the 3D surface model reducing the number of modules that are to be loaded for intraoperative navigation to three.

Since the registration module requires some theoretical background for the preliminary work, i.e. calibration of the C-arm and generation of Digitally Reconstructed Radiographs, as well as for the registration procedure itself, the description of the module is postponed to section 5.3 after having learned about camera calibration in the next chapter and all basics of registration used in this project in chapter 5.

4 Calibration

This chapter describes the mathematical background and the practical application of camera calibration. As worked out in chapter 2, the intraoperative navigation requires an overlay of two image modalities, CT data and X-ray data. This is accomplished by preliminarily calibrating the camera with which the X-ray images are shot, the C-arm GE MS OEC9800. When the calibration is finished, the camera can be simulated in another viewer, such as a simple virtual viewer window on a PC, and the CT data can be displayed from the same perspective as the C-arm is capturing the real patient. The basics and application of calibration are described in sections 4.1 and 4.2. Now, if the CT data is rendered in this viewer to resemble the X-ray data, both image modalities can be overlaid and, hence, are registered. This is accomplished by first creating an artificial X-ray image out of CT data (Digitally Reconstructed Radiograph), which could be seen as a three-dimensional "X-ray object". Then, by adjusting the object's transformation such that the viewer shows the section of the object which matches the real X-ray image, the CT data and the X-ray data can be registered and all information from X-ray images can be transformed to CT data and vice versa. The registration's basics and their application will be described in chapter 5.

4.1 Basics of Camera Calibration

Calibrating a camera means to determine the camera's geometry. This includes the computation of the internal (intrinsic) camera parameters like focal length or height angle respectively, depending on the underlying camera model. Moreover, the camera's external (extrinsic) parameters must be determined describing the pose (position and orientation) of a camera in a reference coordinate system, e.g. the world coordinate system. Merging intrinsic and extrinsic parameters results in a projection, which maps any given 3D point of a scene in reference coordinates to a 2D point on the camera's image plane in image, i.e. pixel coordinates.

Mathematically, the calibration of a camera can be expressed as a 3×4 projection matrix P, which maps a 3D point X in homogeneous coordinates to a 2D point x in homogeneous coordinates:

$$x = PX. \tag{4.1}$$

Hence, P has 11 degrees of freedom up to overall scaling. In the following, most ideas and formulas were depicted according to [25] and [14].

4.1.1 Camera Models

There are plenty of different camera models beginning with the way of projecting a scene – perspectively or orthographically. Since the heARt project focuses on calibrating an X-

ray image intensifier which is a perspective camera, orthographic camera models are not covered. Moreover, cameras can be distinguished by the location of their center, either at the plane at infinity (infinite camera models) or in finite space (finite camera models). Infinite camera models are a generalization of orthographic camera models enabling also, e.g., shearing. In the following, only finite camera models will be described.

The *extrinsic* camera parameters of perspective camera models are always described as a rotation followed by a translation, which transforms any given point in reference (world) coordinates into camera coordinates. The translation is always written as a 3D vector t and the rotation as a 3×3 matrix R. Refer to section 5.2.2 for an extensive explanation of the representation and notation of rotations. It can be easily seen that rotation matrices are **orthonormal**, i.e. their inverse is the transposed matrix ($RR^T = I$) and their determinant is one ($det(R) = 1$).
Let C be the center of the camera, X a 3D point in the reference coordinate system and X_{cam} a 3D point in the camera coordinate system. The transformation can be written as

$$X_{cam} = R(X - C),$$

or, more commonly, since the center of the camera does not have to be expressed directly

$$X_{cam} = RX + t,$$

where $t = -RC$. One can imagine this transformation as a change of basis where all 3D points are now expressed in a coordinate system having the camera center as origin and the camera's view direction as an axis.

Internally, perspective cameras can be described arbitrarily. The most common models determine either a focal length or a viewing angle.

4.1.1.1 Camera Models based on Focal Length

Camera models based on focal length as shown in figure 4.1 have the following intrinsic parameters:

- *focal length*: The distance (mostly measured in mm) from the camera center to the point where an object is in focus, f,
- *principal point*: The intersection between principal axis (line from the camera center perpendicular to the image plane) and image plane, (p_x, p_y),
- *pixel pitch*: the distance of the centers of two adjacent pixels in x- and y-direction on the image plane, m_x, m_y [1].

[1] Mostly, the pixel pitch is known an does not have to be calculated

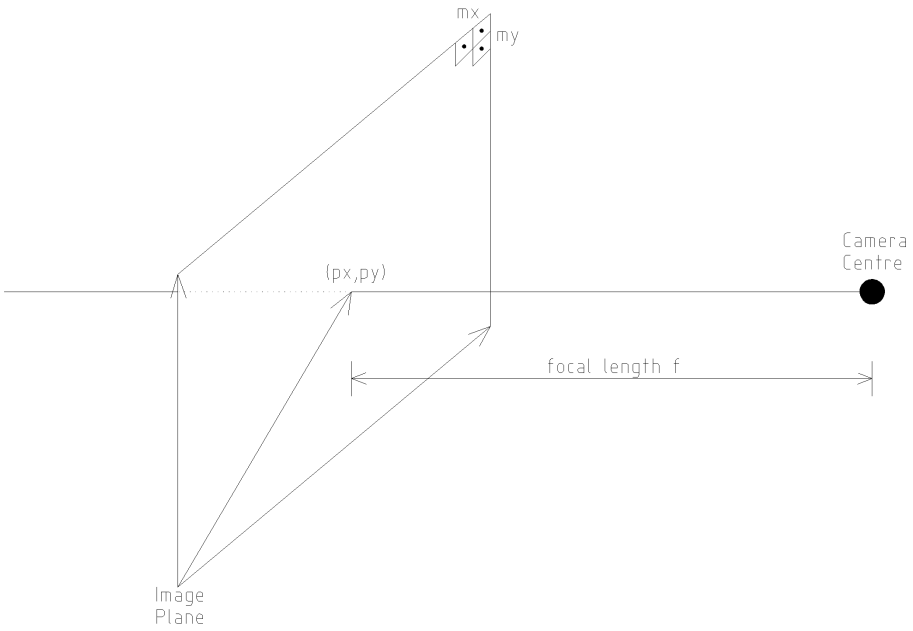


Figure 4.1: Camera Model based on Focal Length

Pinhole Cameras Pinhole cameras assume to have exact square pixels and thus do not model a pixel pitch. The camera models a central projection mapping a 3D point (X_{cam}[2]) to a 2D point x. A central projection can be expressed in inhomogeneous coordinates:

$$(x_{cam_1}, x_{cam_2}, x_{cam_3})^T \mapsto (x_{cam_1}/x_{cam_3}, x_{cam_2}/x_{cam_3})^T,$$

or in homogeneous coordinates[3]:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_{cam_1} \\ x_{cam_2} \\ x_{cam_3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_{cam_1} \\ x_{cam_2} \\ x_{cam_3} \\ 1 \end{pmatrix}.$$

Here, the image plane is at $x_{cam_3} = 1$ and has its origin in the principal point (p_x, p_y).
Now, including focal length, i.e. the image plane is now at $x_{cam_3} = f$, a projection P' can be described with

$$(x_{cam_1}, x_{cam_2}, x_{cam_3})^T \mapsto (f x_{cam_1}/x_{cam_3}, f x_{cam_2}/x_{cam_3})^T,$$

or, homogeneously, with the following matrix[4]:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{P'} \begin{pmatrix} x_{cam_1} \\ x_{cam_2} \\ x_{cam_3} \\ 1 \end{pmatrix}.$$

[2]derived from 3D point X after the transformation into the camera coordinate system
[3]The division by x_{cam_3} is performed internally.
[4]Remember that rotation and translation have already been performed

In the following inhomogeneous notation of the described projections will be abandoned since the homogeneous coordinates are more convenient to read.

Now, the principal point is modelled with a pinhole camera. p_x, p_y model the offset from the image plane's origin to the principal point in *image coordinates* (thus, no central projection). Hence, the matrix is extended:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{P'} \begin{pmatrix} x_{cam_1} \\ x_{cam_2} \\ x_{cam_3} \\ 1 \end{pmatrix} = \begin{pmatrix} f x_{cam_1} + p_x x_{cam_3} \\ f x_{cam_2} + p_y x_{cam_3} \\ x_{cam_3} \end{pmatrix}.$$

CCD cameras CCD[5] cameras can have pixels that are not square and thus model a pixel pitch. Let m_x be the number of pixels in x-direction and m_y the number of pixels in y-direction, Matrix P' can be written as

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_x & 0 & c_0 & 0 \\ 0 & \alpha_y & c_1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{P'} \begin{pmatrix} x_{cam_1} \\ x_{cam_2} \\ x_{cam_3} \\ 1 \end{pmatrix}, \tag{4.2}$$

where $\alpha_x = f \cdot m_x$, $\alpha_y = f \cdot m_y$, $c_0 = p_x \cdot m_x$ and $c_1 = p_y \cdot m_y$.

P' can also be written as a 3×3 matrix K' and a fourth column. Then,

$$P' = [K'|0] = K'[I|0], \tag{4.3}$$

where I is the 3×3 identity matrix.
Merging the former transformation into the camera coordinate system $X \rightarrow X_{cam}$ and the actual perspective projection P' using all intrinsic parameters, the 3×4 matrix P can be expressed (using the syntax from (4.3)):

$$P = K[R|t],$$

where $t = -RC$ and C is the center of the camera in world coordinates. K is a 3×3 matrix describing all intrinsic parameters:

$$K = \begin{pmatrix} \alpha_x & 0 & c_0 \\ 0 & \alpha_y & c_1 \\ 0 & 0 & 1 \end{pmatrix},$$

where α_x and α_y are defined as in equation (4.2).
Counting the degrees of freedom for camera models based on focal length sums up to 10 (4 for matrix K and 6 extrinsic parameters).

Finite Cameras Finite cameras[6] describe one more parameter in K, the skew parameter s and thus sum up the degrees of freedom of the projection matrix P to 11, which is the most general case. However, for most cameras s is zero and hence it is not modelled in this project.

[5] Charged Coupled Device
[6] Cameras whose center is not at infinity

4.1.1.2 Camera Models based on Viewing Angle

Camera models based on a viewing angle (see figure 4.2) have – amongst others – following intrinsic parameters [77]:

- *aspect ratio*: ratio between camera viewing width and height,
- *height and width angle*: angles spanning the image plane,
- *near distance*: the distance from the camera viewpoint to the near clipping plane,
- *far distance*: the distance from the camera viewpoint to the far clipping plane.

For specifying the camera's intrinsic parameters only aspect ratio and one of the angles must be determined since far and near clipping plane can be set to default values.

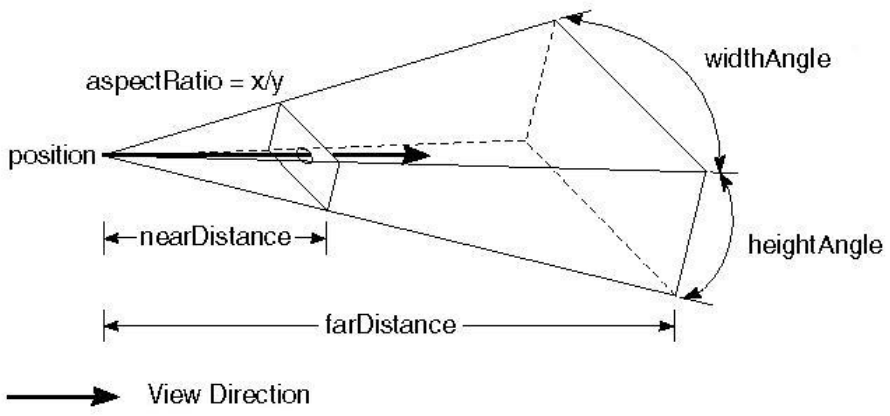


Figure 4.2: Camera Model based on Viewing Angle, from [77]

4.1.1.3 Changing Camera Models

Camera Models based on a viewing angle can be expressed with a model that is based on focal length and vice versa. This shall shortly be demonstrated because it is often necessary when "real" cameras are to be expressed via "virtual" cameras in a computer. Since most calibration algorithms are based on the focal-length-model as will be shown in section 4.1.2, the camera parameters of this model have to be determined. Then, as can be seen in figure 4.3, the height angle is determined as follows.

Let α be the height angle, f_{pix} the focal length in pixels and h the height of the image, i.e. the number of pixels in y-direction.

$$\tan\frac{\alpha}{2} = \frac{h/2}{f_{pix}} \Rightarrow \alpha = 2 \cdot \tan^{-1}\frac{h/2}{f_{pix}}.$$

The width angle is determined analogously. Then, the aspect ratio can be computed. Note that the focal length, which is usually expressed in millimeters is to be expressed in pixels for a sound result.

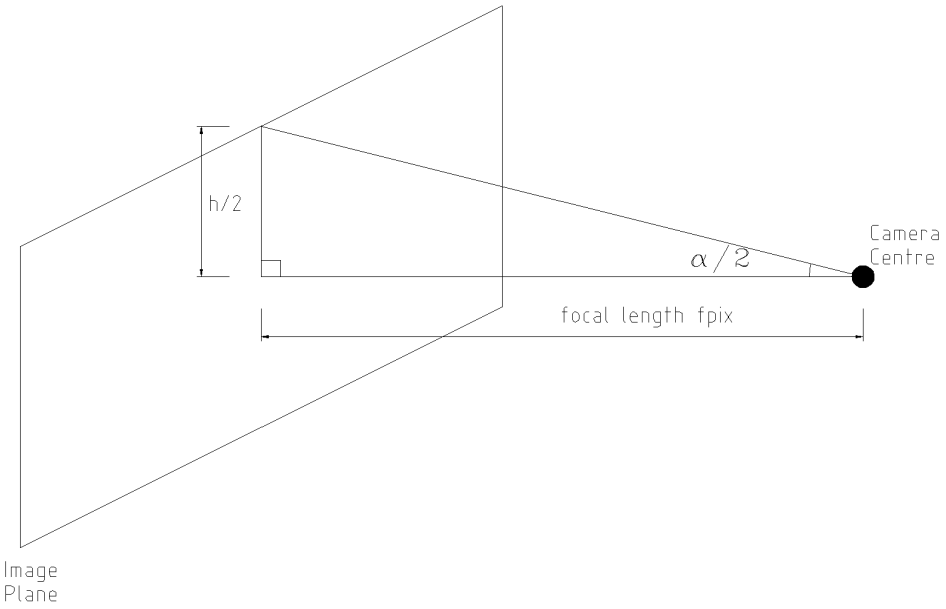


Figure 4.3: Determining α from focal length and aspect ratio

4.1.1.4 Distortion

Due to physical phenomena images of cameras are almost always distorted in a certain way. There are different kinds of distortion like radial, spiral or pincushion (barrel) distortion. Each kind applies to different camera types. One task of camera calibration is to undistort the image, i.e. find the distortion types that apply to the camera and calculate distortion coefficients for undistortion.

Radial Distortion Radial lens distortion depends on the value of focal length. The smaller the focal length, the more distorted an image is.
Mathematically, the distortion can be expressed as a deviation of the ideal (undistorted) projected point $(\tilde{x}, \tilde{y})$ captured, e.g. via a pinhole camera, and modelling the actual (distorted) image point (x_d, y_d) with a function $L(\tilde{r})$:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix},$$

where $L(\tilde{r})$ is a distortion factor only dependent on the radial distance from the center of the image $\tilde{r} = \sqrt{\tilde{x}^2 + \tilde{y}^2}$. Hence, for distortion correction, we can write

$$\hat{x} = x_c + L(r)(x - x_c) \qquad \hat{y} = y_c + L(r)(y - y_c),$$

with (x_c, y_c) the center pixel of the image, $(\hat{x}, \hat{y})$ the undistorted (corrected) image pixels and (x, y) the distorted (uncorrected) image pixels. r can be written as $\sqrt{(x - x_c)^2 + (y - y_c)^2}$. $L(r)$ may be expressed with a Taylor expansion $L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \ldots$. Thus, correcting radial lens distortion means to determine the coefficients $\{\kappa_1, \kappa_2, \ldots\}$.

Pincushion and Spiral Distortion Pincushion Distortion occurs when electrons from a curved input screen are mapped to a flat output (image) screen. Then, the recorded image appears to be either convex or concave. For correcting this distortion the radius of the curved input screen and other geometric information, like, in the case of C-arms, the distance from X-ray source to image intensifier must be considered [58].
The spiral, or S-distortion effect emerges when internal interact with external magnetic fields. In most cases, the external magnetic field is the earth's magnetic field causing the electrons, which are sent through, say, an X-ray image intensifier, to deflect from their usual path. This results in an image where straight lines have the shape of an "S". For correcting spiral distortion a calibration object furnished with visible control points is recorded by the camera with which distortion coefficients are determined. Then, a mapping between distorted image points to corrected image points is calculated by computing local transformations. The grey value of the corrected pixel is interpolated between the surrounding pixels of the distorted pixel [64].
Compensating both distortions can be achieved also globally [65].

4.1.2 Camera Calibration Algorithms

Determining extrinsic and intrinsic camera parameters, as well as distortion parameters is the purpose of camera calibration. There are several algorithmic approaches to solve this problem assuming a number of point correspondences (3D points and their corresponding projection on the image plane, 2D points) as input. They mostly calculate with camera models based on focal length since they are intended to calibrate physical cameras, which are always modelled likewise. Two algorithms are shortly described since they have been used in the heARt project:

4.1.2.1 The Gold Standard Algorithm

This algorithm shows the standard way of solving the calibration problem. Actually, it serves as a reference algorithm for determining cost and efficiency of other algorithms. Given are two sets of corresponding points in 3D X_i and 2D x_i, respectively. To compute the projection matrix P we can use equation (4.1) and perform a *Direct Linear Transformation* (DLT) [1, 72]:

$$x_i = PX_i.$$

Hence, using the cross product we have

$$x_i \times PX_i = 0.$$

Let $x_i = \begin{pmatrix} x_{i_1} \\ x_{i_2} \\ x_{i_3} \end{pmatrix}$ in homogeneous coordinates and $p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$, where p_1^T is the first **row** of P:

$$\underbrace{\begin{pmatrix} 0^T & -x_{i_3}X_i^T & x_{i_2}X_i^T \\ x_{i_3}X_i^T & 0^T & -x_{i_1}X_i^T \\ -x_{i_2}X_i^T & x_{i_1}X_i^T & 0^T \end{pmatrix}}_{A'} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = 0.$$

Since the three rows of A' are linearly dependent, choose two and name the 2×12 matrix A_i.

Having n point correspondences, stack up the equations in A_i: $A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix}$.

Now, the equation $Ap = 0$ is to be solved.

For getting an ideal solution for this linear system of equations, $5\frac{1}{2}$[7] point correspondences have to be determined. However, there are mostly more than 6 point correspondences in which case a best solution for P can be determined by minimizing an error:

- *algebraic error:* Minimize the norm $\|Ap\|$ after normalizing p.
- *geometric error:* Minimize $\sum_i d(x_i, PX_i)^2$, where $d(x, y)$ is the Euclidean distance from x to y. (Maximum Likelihood Estimation of P in case of Gaussian errors and exact 3D world points).
- *geometric error of inexact 3D points*: Determine $\min_P \sum_i d_{Mah}(x_i, P\hat{X}_i)^2 + d_{Mah}(X_i, \hat{X}_i)^2$, where $\hat{X}_i$ are the estimated 3D points, X_i the exact 3D points and $d_{Mah}(x, y)$ the Mahalanobis distance.

With this mathematical approach, the Gold Standard Algorithm determines an initial projection matrix via the Direct Linear Transformation (DLT) Algorithm based on a Singular Value Decomposition (SVD). Iteratively, the algorithm optimizes the (over-determined) solution by minimizing an error. The optimal solution P can then be split up into a camera calibration matrix K, rotation R and camera center C by using equation $P = [M| - MC] = K[R| - RC]$ and performing a simple RQ-decomposition. This approach does not consider lens distortion yet.

Given: A corresponding point set $x_i \leftrightarrow X_i$.

1. *Normalize x_i and X_i. Use a similarity[8] transformation T to normalize image points and U to normalize space points:*
$$\tilde{x_i} = Tx_i \quad \tilde{X}_i = UX_i.$$
2. *DLT:*
   a) *For each correspondence $\tilde{x_i} \leftrightarrow \tilde{X}_i$ compute matrix $\tilde{A}_i$.*
   b) *Assemble the n 2×12 matrices $\tilde{A}_i$ into a single $2n \times 12$ matrix $\tilde{A}$.*
   c) *Do the **SVD** of $\tilde{A}$: $\tilde{A} = UDV^T$; $\tilde{p}$ is the last column of V and $\tilde{P}$ is formed from the entries of $\tilde{p}$.*
3. *Minimize error iteratively[9].*
4. *Denormalize $\tilde{P}$ to receive the projection matrix: $P = T^{-1}\tilde{P}U$.*

[7] Every point correspondence constraints 2 degrees of freedom of which there are 11. $\frac{1}{2}$ means to just take the first row of the 6th point correspondence

[8] a transformation class preserving the transformed object's shape

[9] with an algorithm like Levenberg-Marquart, Newton or Gradient Descent

4.1.2.2 Tsai's Approach

Another approach for calibrating a camera is described in [71]. It suggests an image formation pipeline starting from points in world coordinates gradually calculating their corresponding points in image coordinates:

Given: Points in world coordinates (X_w, Y_w, Z_w)

1. A world-to-camera-transformation is performed transferring world coordinates to camera coordinates:
$$(X_w, Y_w, Z_w) \rightarrow (X, Y, Z)$$
The parameters necessary for this step are rotation R and translation t.

2. Then, an idealized 3D-2D-projection is made using the parameter f (focal length):
$$(X, Y, Z) \rightarrow (x_u, y_u)$$

3. Applying a radial lens distortion (κ_1, κ_2) results in the true image coordinates:
$$(x_u, y_u) \rightarrow (x_d, y_d)$$

4. After TV scanning and sampling, the computer image coordinates are determined:
$$(x_d, y_d) \rightarrow (x_f, y_f)$$
Here, the principal point (p_x, p_y) and a scaling factor s_x are used.

The algorithm computes all the parameters used in the image formation steps in two stages:

Input: *Point correspondences $\{X_i \leftrightarrow x_i\}$, pixel pitch and the number of pixels in the frame grabber's x direction are given.*
Stage 1*: Compute the rotation R, and the first two entries of t, t_x, t_y:*[10]

- *Get distorted image coordinates (x_d, y_d).*
- *Compute five mixed terms $t_x^{-1}r_1, t_y^{-1}r_2, t_y^{-1}t_x, t_y^{-1}r_4, t_y^{-1}r_5$.*
- *Determine R and t_x, t_y from mixed terms.*

Stage 2*: The effective focal length f, the distortion coefficients (κ_1, κ_2) and the third entry of t, t_z are computed:*

- *Compute an approximation of f and t_z ignoring lens distortion.*
- *Compute the exact solution for f, t_z and κ_1, κ_2.*

Unlike the Gold Standard Algorithm, Tsai's Algorithm does not optimize the overdetermined solution. However, the distortion is calculated within this approach. For an extensive explanation of the two steps, please refer to [71].

[10] The heart project used the Tsai algorithm for the coplanar approach for which the scale factor s_x is not computed but assumed to be given.

4.1.3 Forward Projection of 3D Points

When the projection matrix P has been determined, it can be applied to 3D points $\hat{X}$ resulting in the projected 2D points $\hat{x}$:

$$\hat{x} = P\hat{X}.$$

This step is called *forward projection* and it can be used to either calculate the corresponding point in image coordinates to a given point in world coordinates, or, by applying P to the 3D points of the point correspondences with which the camera has been calibrated, $\{x_i \leftrightarrow X_i\}$, testing the quality ot P.

4.1.4 Back-Projection of Points to Rays

Given the projection matrix P, a line (ray) can be sent from the camera center through a certain pixel of the image intersecting those 3D points, which project on this pixel. The back-projection is thus important for finding the corresponding points in world coordinates of a point in image coordinates as well as for testing the quality of a calibration method: Back-projecting all the 2D points x_i of the given point correspondences $\{X_i \leftrightarrow x_i\}$ must result in the points X_i. Assuming the calibration of a finite camera, the line can be depicted as follows:
For a line, two points must be given. If P is written as $[M|p_4]$, the camera center is given by $C = -M^{-1}p_4$ since $P = K[R|t] = K[R|-RC] = KR[I|-C]$ and $P = [M|p_4] = M[I|M^{-1}p_4]$. Also, the intersection of the ray and the plane at infinity lies on the line and gives us the second point. This point can be expressed as $D = ((M^{-1}x)^T, 0)$, where x is the 2D point the ray is sent through. Now, the line can be written as

$$l(\lambda) = \lambda \begin{pmatrix} M^{-1}x \\ 0 \end{pmatrix} + \begin{pmatrix} -M^{-1}p_4 \\ 1 \end{pmatrix} = \begin{pmatrix} M^{-1}(\lambda x - p_4) \\ 1 \end{pmatrix}.$$

4.2 Calibration of the C-arm GE MS OEC9800

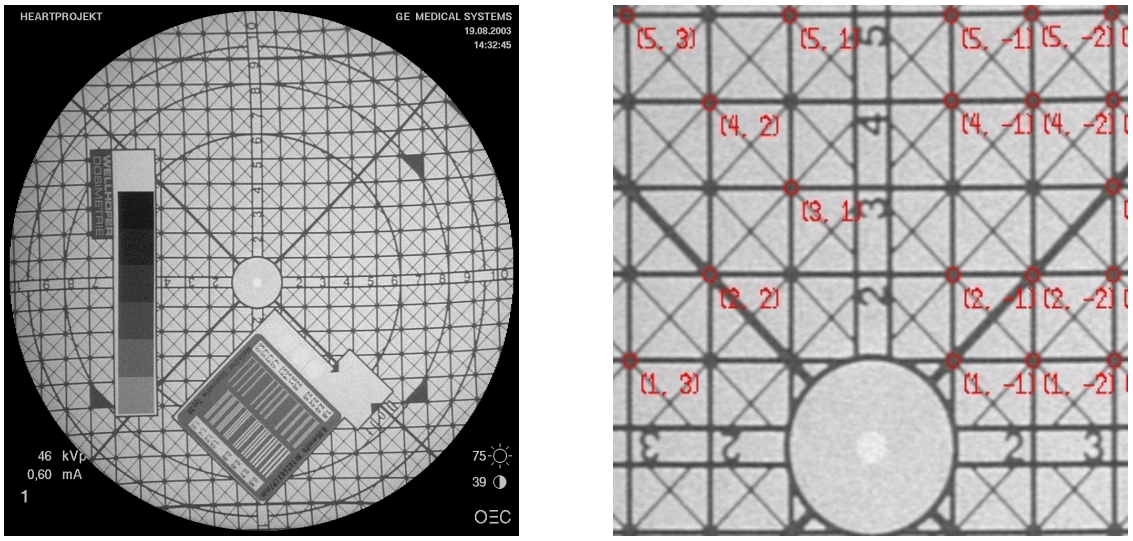
In this section the methods of section 4.1 are applied to the C-arm used at the DHM, the GENERAL ELECTRIC Medical System Series OEC9800. Both algorithms were used, the Gold Standard Algorithm based on the Direct Linear Transformation and the approach of Tsai. In both cases, libraries were employed implementing the algorithms. The Open Computer Vision library [35] (*OpenCv*) by INTEL[11] offers methods for calibrating via Gold Standard whereas the library by Reg Willson from the Carnegie Mellon University in Pittsburgh[12] copes with Tsai's approach.

As described in 3.1.2, the C-arm contains a CCD camera, hence the calibration methods of section 4.1 can be applied successfully.

[11] `http://www.intel.com/research/mrl/research/opencv/`
[12] `http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html`

4.2.1 Acquisition of Point Correspondences

In order to calibrate the C-arm several images of a calibration body must be taken which assign previously defined points in world coordinates to image coordinates. In the heARt project a calibration plate was used offering the possibility to perform a coplanar calibration of the C-arm, see figure 4.4(a). The world coordinate system's origin was placed in the



(a) The plate for C-arm calibration

(b) Section of the calibration plate with point correspondences

Figure 4.4: Acquiring point correspondences via a calibration plate

middle, the positive y-axis directs from the mid-circle to the left, the positive x-axis from the circle upwards. The 3D points are exact because the calibration plate was machined with high accuracy. Thus, the geometric error function could be used for optimization. Since this was a coplanar calibration, the z-coordinate of all 3D points was zero with a z-axis is perpendicular to the calibration plate. 12 images of the plate were recorded where the plate's angle and distance to the camera center were changed arbitrarily. Afterwards, a file was created where about 30 to 40 point correspondences were entered for each image. These point correspondences were not optimized algorithmically, i.e. no sub-pixel accuracy was applied. Figure 4.4(b) shows how the correspondences were set. The circles were drawn by pixel coordinates and the coordinates in brackets express the point in world coordinates[13].

4.2.2 Determining Intrinsic Parameters

For calculating the C-arm's intrinsic parameters an entire calibration must be performed, i.e. computation of intrinsic and extrinsic parameters. Calibrating with OpenCv's method `cvCalibrateCamera(...)` requires a set of image files and the point correspondences for each image. The code of Tsai just requires one image and its point correspondences for

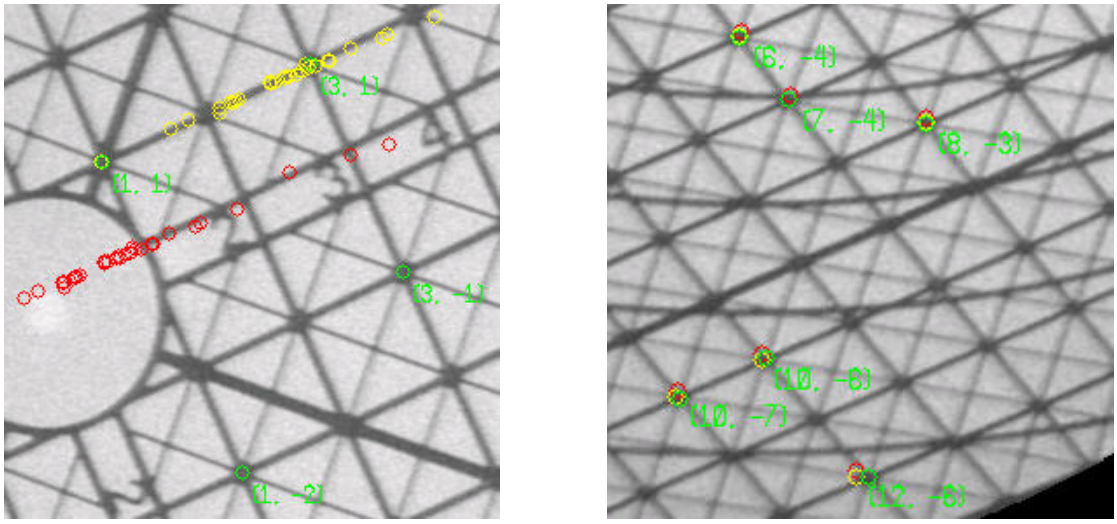
[13]Since the z-coordinate is always zero, it was left out

executing the method `coplanar_calibration_with_full_optimization()` but also needs some additional information like the number of pixels in x- and y-direction as well as the actual pixel pitch, distance of two and number of sensor elements in the camera and the principal point in pixel coordinates[14].

Calibration via Tsai The Tsai algorithm in its original form does not provide an error optimization for all determined parameters. However, the library of Reg Willson optimizes the calculated parameters with a modified Levenberg-Marquart iterative approach using the `lmdif` function of the MINPACK[15] library, which is programmed in FORTRAN and offers efficient methods for solving nonlinear systems of equations and nonlinear least squares problems (as in this case). As it turned out, this optimization could still be improved and thus an additional optimization algorithm was added based on the *Weighted Best Neighbor* iterative approach [30]. Here, the geometric error (distance between optimized point and actual point in the image in pixels) was minimized. The calculated intrinsic parameters were evaluated by doing a forward projection using the respective projection matrix of each image. It was tested whether the forward-projected 2D points matched the measured 2D points within a deviation of three pixels. This forward-projection was just performed on the image the intrinsic parameters were determined with, i.e. the quality of the parameters had just a "local" character and were not checked against the point correspondences of other images. This quality is further on referred to as *intern quality*. For instance, figure 4.5 shows the forward projection as well as its optimization of images four and ten with intrinsic parameters $f = 9.75673\ mm$, $(p_x, p_y) = (603.975, 375.091)$ and $f = 4375.21\ mm$, $(p_x, p_y) = (514.207, 490.170)$ respectively. The green circles highlight the exact (measured) 2D points, the red ones the forward-projection with the original parameters depicted via Tsai and Levenberg-Marquart, and the yellow ones the parameters optimized with Weighted Best Neighbor. Obviously, the parameters of image ten are by far better than those depicted with image four. Tables B.1 and B.2 in appendix B show the intrinsic parameters and their respective intern quality of all images depicted via the Tsai algorithm with and without Best Neighbor optimization. The optimization disregarded the radial distortion coefficient, κ. The unusual values of the focal length result from the electron optics and camera lenses integrated in the C-arm. Because of their magnifying or lessening properties, the actual focal length has nothing to do with the dimensions of the C-arm, like the distance between X-ray source and image plane.

Calibrating via OpenCv OpenCv offers an optimization algorithm minimizing also the geometric error between projected points and actual points.
Since OpenCv does not require to enter the pixel pitch in advance, it calculates two focal length variables, f_x and f_y expressing the values of α_x and α_y as described in 4.1.1.1. As mentioned above, OpenCv's calibration algorithm iterates over all images and calculates the best intrinsic parameters. For the 12 calibration images, following intrinsic parameters were determined by OpenCv:

[14] The principal point could be chosen as the center of the image, during the algorithm it is altered to match the actual principal point
[15] `http://www.netlib.org/minpack/`



(a) (b)

Figure 4.5: Projected and measured 2D points on the calibration plate depicted via Tsai for (a) image four and (b) image ten. The green circles show the measured 2D points, the red ones the forward-projected points and the yellow ones the optimized forward-projections.

- *focal length:* $f_x = 4502.27$, $f_y = 4712.48$,
- *principal point:* $(p_x, p_y) = (526.691, -35.3302)$,
- *radial distortion:* $(\kappa_1, \kappa_2, \kappa_3, \kappa_4) = (0.3282, 5.22586, -0.141093, -0.00823205)$.
- *intern quality (after optimization):* 5.89744%.

4.2.3 Determining Extrinsic Parameters

The computation of extrinsic parameters, i.e. the C-arm's pose in the world coordinate system, can be separated from the calibration problem. If intrinsic parameters are given, the Tsai code offers a method `coplanar_extrinsic_parameter_estimation()` that calculates rotation R and translation t of the camera's pose. OpenCv also offers such a method, called `cvFindExtrinsicCameraParams(...)`. Hence, when a camera is calibrated for one image and the intrinsic parameters are stored in a file, the pose of the camera could be calculated directly for all other images. This was necessary to evaluate the quality of determined intrinsic parameters as described in 4.2.5.

Whenever the world coordinate system changes, a new rotation and translation must be calculated. Since the calibration plate's pose was chosen arbitrarily during image acquisition, rotation and translation differ for each image. However, since the image plane is rather small, neither x- and y-position nor angle around x- and y-axis could be altered too much because most of the plate should be visible on the image. Comparing the calibration images

and, e.g., their translations computed by OpenCv as listed in appendix B , one can see the direct correlation between translation and images: The smaller the distance between calibration plate and camera, the smaller the z-translation-value.

4.2.4 Distortion

C-arm and image intensifiers respectively mainly distort the image with pincushion, but also spiral distortion [47, 22, 64, 58]. The GE MS OEC9800 has a convex receptor (image) plane with a solid radius of approximately 200 mm, which is the major reason for the pincushion distortion. However, this is largely compensated by the optics that couple the image to the CCD camera [2]. The spiral (S-shaped) distortion is not compensated as can be seen in the calibration images in appendix B. In the heARt project the S-distortion was not coped with, see section 6.3, which will be proved not necessary for the calibration accuracy as will be shown in the following.

4.2.5 Testing the Calibration Quality

A most important issue in this project was to test the quality of the calibration, i.e. rate the resulted projection matrices or the intrinsic parameters respectively, since a "virtual C-arm" has to have the same parameters as the real one. In general, the testing was accomplished by forward- or back-projecting points with a certain projection matrix including the intrinsic parameters and evaluating the error between calculated and measured point. As it turned out, projection matrices depicted with Tsai provided a better quality than the one calculated with OpenCv. Moreover, images with an inclination of about 30 degrees in respect to the image plane produced the best results.

4.2.5.1 Applying the Projection Matrix

Here, a forward-projection was used, i.e. the projection matrix P as determined from the calibration was applied to the measured 3D points (taken from the point correspondences $\{x_i \leftrightarrow X_i\}$). Then, the resulting 2D points $\hat{x}_i$ were compared to the actual measured 2D points x_i. If the distance between $\hat{x}_i$ and x_i was more than a certain number of pixels (deviation), the point was classified as *bad*, otherwise as *good*. This was done with all images and a sum of good points was compared to the sum of all point correspondences. The deviation was increased from 0 to 9 pixels (9.8 pixels are 1% of the image). Naturally, each time before a projection matrix was applied to an image, the extrinsic parameters had to be determined. The procedure was repeated for all intrinsic parameters with an *intern quality*[16] of more than 75% as listed in table B.2. Due to the bad quality of OpenCv's calibration, its determined intrinsic parameters were left out.

The result of the forward-projection is listed in table 4.1. The quality is measured by the number of points, which project well out of measured 390 points (of all images). The deviation is given in pixels.

[16] intern quality means that the quality was just evaluated with the same picture the intrinsic parameter were determined with. It was not tested with point correspondences of other pictures.

Quality of intrinsic parameters calculated by image *i* in [%]						
Deviation [pix]	image 1	image 6	image 9	image 10	image B	image F
0	0	0	0	0	0	0
1	12.8205	12.8205	4.87179	11.7949	8.71795	10.7692
2	28.9744	29.7436	15.1282	30.7692	22.3077	29.7436
3	42.5641	55.3846	25.1282	54.6154	31.0256	48.4615
4	54.6154	73.3333	34.1026	72.3077	39.4872	66.6667
5	62.8205	81.2821	42.8205	82.3077	48.2051	77.4359
6	72.3077	86.9231	47.9487	86.6667	56.9231	83.0769
7	77.9487	90.2564	51.7949	91.0256	65.1282	87.9487
8	81.2821	94.1026	59.4872	95.1282	71.2821	90.7692
9	87.4359	97.6923	66.6667	96.9231	75.641	94.3590

Table 4.1: Quality of intrinsic parameters depicted with forward-projection

4.2.5.2 Back-Projection

The second quality test was performed via a back-projection. Given a projection matrix, the camera center C could be computed. Then, a ray was sent from C through the pixel belonging to a measured 2D point x_i, see section 4.1.4. Afterwards, the closest 3D point of the measured 3D point X_i on the ray is determined via an orthogonal projection of X_i. This was accomplished by first determining the line's parameterized form characterizing the ray and by creating a plane including X_i plus the line's direction vector as normal. Putting the line described as a general point in the plane's equation results in the orthogonal intersection of plane and line:
Let l be the parameterized line, X_i the exact (measured) 3D point and E the plane.

$$l(\lambda) = t + \lambda v = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} + \lambda \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}, \qquad X_i = \begin{pmatrix} X_{i_1} \\ X_{i_2} \\ X_{i_3} \end{pmatrix},$$

$$E: \; v_1(x_1 - X_{i_1}) + v_2(x_2 - X_{i_2}) + v_3(x_3 - X_{i_3}) = 0.$$

Now, intersecting E and l (putting l in E) results in:

$$\begin{aligned} v_1(t_1 + \lambda v_1 - X_{i_1}) + v_2(t_2 + \lambda v_2 - X_{i_2}) + v_3(t_3 + \lambda v_3 - X_{i_3}) &= 0 \;\Rightarrow \\ v_1 t_1 + v_2 t_2 + v_3 t_3 - (v_1 X_{i_1} + v_2 X_{i_2} + v_3 X_{i_3}) + \lambda(v_1^2 + v_2^2 + v_3^2) &= 0 \;\Rightarrow \\ v \circ t - v \circ X_i + \lambda \|v\|^2 &= 0 \;\Rightarrow \\ \lambda = \frac{v \circ X_i - v \circ t}{\|v\|^2}, & \end{aligned}$$

where $x \circ y$ is the dot product of x and y.

Now, we obtain the orthogonal projection by setting λ in l.
The Euclidean distance (in world coordinates, i.e. in cm) between X_i and its orthogonal projection on the ray, and thus the quality of P were determined. Naturally, the distance

between 3D points is either larger or smaller compared to the distance of corresponding 2D points in the image plane, dependent on the 3D point's pose. Since the C-arm projects an object which lies between camera center and image plane, the distances become smaller. Table 4.2 shows the average distance (in cm) between 3D points and their back-projected counterparts. Comparing the back-projection results to those depicted in [57], the calibration appears to be quite accurate. Again, the intrinsic parameters with an *intern quality* of more than 75% were evaluated by calculating the back-projections of all measured point correspondences of all images.

Distances between measured and back-projected 3D points, in [cm]						
Points of image	image 1	image 6	image 9	image 10	image B	image F
image 1	0.0218141	0.0225314	0.0625993	0.0198608	0.0416237	0.0317641
image 2	0.0894220	0.0886159	0.1770170	0.0874283	0.1385500	0.0965018
image 4	0.1065360	0.1117410	0.1369610	0.1429720	0.1211800	0.1244330
image 5	0.0511403	0.0309952	0.0797013	0.0307571	0.0667306	0.0332513
image 6	0.0612515	0.0509424	0.1449240	0.0494305	0.1135880	0.0563015
image 7	0.0519014	0.0351270	0.0654322	0.0373280	0.0556715	0.1427340
image 8	0.0995274	0.0773950	0.1792030	0.0693770	0.1488310	0.0946012
image 9	0.0337086	0.0256236	0.0416045	0.0264901	0.0322850	0.0304796
image 10	0.1672140	0.0616630	0.2011360	0.0615496	0.1610370	0.0662518
image A	0.0665273	0.0456392	0.1013690	0.0409824	0.0726594	0.0575106
image B	0.0307379	0.0280492	0.1120460	0.0302969	0.0794892	0.0268466
image F	0.1235760	0.0444925	0.0941122	0.0404419	0.0871840	0.0536926
Average	**0.0752797**	**0.0519322**	**0.1184260**	**0.0533240**	**0.0933496**	**0.0616750**

Table 4.2: Distances between measured and back-projected 3D points

Tables 4.1 and 4.2 show that the intrinsic parameters with the best *intern quality* are **not** the best judging the projection quality in all images. For instance, image 1 has an *intern quality* of 100 percent but does obviously not provide the best parameters regarding point correspondences of, say, image 10. As a matter of fact, the intrinsic parameters – depicted with the Tsai calibration – of images 10 and 6 returned the best results. This is due to the angle the calibration plate was positioned in respect to the image plane. The Tsai algorithm returns best results when this angle is about 30° which roughly applied to image 10 as well as image 6. However, the forward- and back-projection showed that without enough deviation the quality is rather bad for the determined intrinsic parameters. For a better accuracy, sub-pixel methods are recommended and S-distortion should be considered which has been left out entirely. Moreover, for calibrating via OpenCv, another calibration body should be taken. A metal chessboard would be more suitable since OpenCv offers sub-pixel accuracy algorithms based on feature detection for chessboard images. However, the calibration of the C-arm provided intrinsic parameters that guaranteed the feasibility of the heARt project.

In conclusion, we have depicted a focal length of $f = 4375.21\ mm$ and a principal point of $(p_x, p_y) = (514.207, 490.170)$ from calibration[17].

[17]These are the intrinsic parameters from image 10. We could also have chosen the ones from image 6; a mean

4.2.6 From Focal Length to Viewing Angle

Up to now, the calibration of the C-arm was accomplished only with one camera model, the model based on focal length because it is more suitable to compute intrinsic parameters and all literature about calibration is based on camera models with focal length. However, most graphical applications, which can handle scene graphs like OPEN INVENTOR have an intern camera model based on a viewing angle. Since amira was used, which in turn builds on OPEN INVENTOR, the camera models had to be changed, i.e. height angle and aspect ratio had to be determined for the C-arm. The C-arm produces images with a resolution of 980×980 pixels, so the aspect ratio could be set to 1. Then, the height angle was computed. This was done by first expressing the focal length not in *mm* but in pixel. For this, two 3D world points were picked that lay exactly on one line. Since the calibration plate had metrics printed, it was easy to depict a distance in world coordinates. By applying the projection to those two world points, the distance in pixel of the two resulting image points was measured and thus, the focal length could be transformed. With the focal length in pixel and the resolution of the image, the height angle could be calculated using the method described in section 4.1.1.3.

A height angle of $2.81643885°$ could be determined. As will be described in the next chapter, this angle was used to initialize the OPEN INVENTOR camera of amira in order to model the C-arm properly in a viewer window. Only with this information for the camera, the alignment of the two images, CT and X-ray could be accomplished.

value of both intrinsic parameter sets did not give good results.

5 Registration

Registration is defined as the spatial alignment of two different image modalities rendering the same objects in 3D space. More precisely, besides registrational *problem statement* and *paradigm*, the actual registration procedure is the automatic or interactive (or both) *optimization* of this alignment using arbitrary techniques [40]. An explanation of the terms *problem statement*, *registration paradigm*, and *optimization procedure* confined to the medical sector as well as a classification of the registration problem faced in this project will be given in the following section. The alignment is expressed in a mapping function transferring points from one modality into the other. Its quality can be rated by an error function, which must be minimized (or maximized) [56]. For registration, a visual comparison of the two image modalities must be possible. This can be accomplished by simulating the first in the second modality, i.e. temporarily transferring one modality into the other and adjusting both images. In this context, the abstract term modality can differ in its meaning. In general, two image modalities can be distinguished by the viewpoint from which the images have been taken, the time when they have been shot or the sensor with which the pictures were recorded [8, 80]. Thus, it is sometimes sufficient to just rotate, translate, and scale one image so that it resembles the other (only the viewpoint is different in the image modalities). Here, only the camera's geometry as described in chapter 4 and suitable transformations are to be determined. Sometimes, preliminary steps have to be taken to fit the two image modalities, e.g., as in our project, creating Digitally Reconstructed Radiographs to make CT data look like C-arm data (here, also the modalities' sensors are different). In computer science, these registration techniques are referred to as *rigid registrations*. The basics of registration and creating DRR's and their application are presented in sections 5.2 and 5.3.
Moreover, there are cases where 3D objects change their shape as time passes. For instance, regarding medical imaging, picturing blood vessels two times in the same viewpoint and sensor modality does not mean to have identical images since vessels move in the body. Taking this fact into account leads to the second type of registration, the *non-rigid registration*. The non-rigid approach was not used in this project although the two image modalities do differ in time. However, since the registration was performed on (rigid) bones rather than vessels, time differences were not considered.

5.1 Classifying the Chosen Registration Method

As introduced in [74] for computer vision and adapted for the medical imaging sector by [40], a registrational problem may be classified and thus the right approach can be chosen to solve it. Generally, registration problems can be divided up into three major parts, the *problem statement*, the *registration paradigm*, and the *optimization procedure*. The first part covers the modalities that are used to acquire image data, or the object to be recorded. The second part influences some decisions regarding the general registration approach, like whether or

not markers are used, if the registration is rigid or not, or if the user must interact with the system in order to register images. Finally, the third part chooses the actual registration *procedure*, i.e. how the optimization algorithm iterates and which error functions are used to determine the quality of an alignment. In the following, these three parts are decomposed further to have an impression of the vast range of possibilities that is provided by science in the field of medical image registration. In order to trace the outline of the project's problem, it is also classified by this catalogue.

5.1.1 Registrational Problem Statement

The problem statement can be split up into four sections. It constrains the *domain* in which the registration is to be applied.

Dimensionality The dimensionality constitutes if all data is spatial or not. Altogether, three cases can be distinguished, determining either a 2D/2D (e.g., slices of CT data) homography[1], a 2D/3D alignment, or a 3D/3D (like MRT and CT data) homography. Obviously, our project covers the 2D/3D case since C-arm data is two-, CT data three-dimensional. The dimensionality also includes the fourth dimension (time), which is not applicable in this approach because only one X-ray image has been considered, not a series of them (as in angiography).

Modalities involved In medicine a variety of modalities for acquiring images is offered ranging from CT or CTA, MR or Positron Emission Tomography (PET) to endoscopic or X-ray imaging methods. Most importantly, it has to be determined *how many* modalities are involved for the registration task. Monomodal and multi-modal registration is distinguished, but there are also cases where only one modality is to be registered with a model of the patient or even the patient itself. This project uses multi-modal data, X-ray and CT.

Subject Here, the question is whether all images recorded are from one single patient (*intrasubject*, as in this work), from different patients, or from patient and model (*intersubject*). Moreover, if one image is created from training images and aligned with an image of a patient, the registration is called *atlas*, see section 3.2.2.

Object The imaged part of the patient's body is called the object. This project covers the thorax either entirely or constricted to the breast, but also abdomen should be considered since aneurysms can appear in all these areas[2]. The choice of an object decides, for instance, how accurate the registration must be. A registration of head images mostly has to be very precise since the alignment is used for neural surgery or tumor treatment in the brain. Regarding the thorax and the actual treatment that is performed the accuracy does not have to be that precise although a deviation of, say, 0.5 centimeter is not desirable.

[1] A homography is a projective transformation (collineation) taking each point of one set $\{x_i\}$ to a corresponding point of a second set $\{\hat{x}_i\}$.

[2] mostly, they occur in the abdomen, however, at the DHM most of the treated aneurysms were thoracic

5.1.2 Registration Paradigm

Again, four sections are relevant for the paradigm of a registration. They determine how the registrational problem was coped with, i.e. which algorithms are applicable to the domain that was depicted before.

Nature of registration basis The nature of the registration basis is the very core of all registration procedures. Basically, there are three approaches, *extrinsic*, *intrinsic*, and *non-image based* methods.
The first approach works on artificial objects attached to the patient and can be partitioned into *invasive* and *non-invasive* procedures. The former fixes fiducials[3] rigidly to the patient's body, like screw markers mounted on a patient's skull, while the latter just temporarily attaches fiducials to the patient, e.g. on the skin.
The second approach, intrinsic registration, can again be split up into three sub-parts: *Landmark based* methods just use a set of points that are anatomically or geometrically distinguishable from other features in the image for an alignment. Thus, the application area of the registration is reduced from images with much data describing voxels or pixels or both to point sets and can be sped up significantly using efficient point matching algorithms [3, 73]. *Segementation based* registration uses extracted features like surfaces or curves to align images. This is a very promising approach and actually used quite often, however, the quality of segmentation influences the registration outcome. Mostly, segmentation cannot be fully automated and is thus rather error-prone. The last intrinsic method is based on *voxel properties*, which work directly on the image data. To each voxel in volumetric data certain properties are attached, like grey scale or neighbor voxels. From these properties additional information can be derived like gradients or normals with which an alignment can be performed.
Finally, the third approach does not need any image data, but, by knowing all image coordinate systems, can align the different modalities. In order to gain this information, the imaging devices either have to remain at certain well-known locations (and orientations) or have to be tracked.

For this project, the focus as required by clinicians was set on using as little additional equipment as possible, i.e. ideally just using a PC or a workstation to align the images. One reason is that the more hardware equipment, e.g. fiducials, are used, the more clinical standards on sterility have to be met. Hence, all extrinsic methods were abandoned. Moreover, a hectical clinical environment cannot assure to track the position of the C-arm, which would also require additional hardware like markers to be fixed on the C-arm. Since the C-arm is a mobile device, the non-image based registration was also discarded. From the remaining approach the first intuition was to use anatomical landmarks, i.e. physicians must specify a set of corresponding points in the two image modalities for registration (obviously, automatically finding these correspondences is not possible without using fiducials). However, observing the images created with the C-arm and the (even segmented) CT data, this interactive task could not be accomplished due to a lack of really salient points. Due to the drawbacks of segmentation based methods mentioned above the third intrinsic approach was chosen based on voxel properties. By performing volume rendering of CT data via ray casting (see section

[3] artificial markers, which are salient to the image sensors being used; they are not to be mixed up with *virtual* markers used in planning.

5.2.1) with adjusted grey scales the emission and absorption properties of each voxel were used to create an artificial X-ray image resembling the real X-ray image. These two images were then used to carry out the registration.

Nature and domain of transformation The nature of the transformation step leading to registered image modalities can be rigid, affine, projective or curved. The letter three ones are named non-rigid transformations. The nature is thus split up into four classes, the first one is described in section 5.2.2. The fact that this work avails of rigid transformation should be explained here. Non-rigid registrations make sense, e.g. if the object to be registered is deformable or if image modalities are highly distorted. The distortion is neglected, as pointed out in chapter 4. The object is indeed deformable since contrast enhanced X-ray images of the thorax show vessels, e.g. the aorta. Since vessels are soft tissue, they do not remain rigid in the body. However, as already mentioned above, the registration was only performed on a part of the image where merely boney structures were visible, which are rigid and thus applicable to rigid registration procedures.
The domain of the transformation is partitioned into global and local transformations. In this project, only global transformations were performed.

Interaction Interaction implies the amount of user influence to the registration procedure. There are totally interactive registration systems, which only visualize the data in a correct way and provide transformation methods that can be applied by the user. For instance, mono-modal 2D slices of microscopic images can be aligned with the graphical framework that is used in this project, amira. However, the more interesting registration approaches for computer scientists are, clearly, the semi-automatic or automatic ones. Semi-automatic registration either requires user initialization for, or user correction of the automatic procedure or both. As a matter of fact, many registration algorithms require an initialization of the registration with, say, primary transformation parameters for the optimization method to avoid terminating in local minima or maxima. The heARt project also followed this approach performing an interactive global rigid transformation in order to roughly align X-ray and CT data and then optimize the transformation parameters by an algorithm. Most researchers tend to perform a fully automated registration, which is applicable, but often not feasible due to cost, efficiency or other clinical criteria.

5.1.3 Optimization Procedure

In general, two different optimization procedures are distinguished. Either registration parameters are computed or given parameters are optimized by computing a maximum or minimum. The entire computation of registration parameters can only be performed on point sets or images with very little information whereas optimizing given parameters allow an optimization on large data sets. Optimizing parameters mostly requires an iterative algorithm. For image registration, many iterative algorithms were proposed, like the *Newton* iteration or the *Levenberg-Marquart-Algorithm*. This project used two approaches, *Weighted Best Neighbor* and *Best Neighbor*. For details on the algorithms as well as a reasoning why they have been chosen refer to [30]. Moreover, the used error function that determines the quality of computed parameters is of importance. There are many error functions or *similarity*

measures like cross correlation, mutual information, or pattern intensity, which can be aggregated into area-based methods and feature-based methods. Area-based or intensity-based methods are used when images have much information about grey scales, pixels/voxels etc. whereas feature-based methods provide a better similarity measure when images with obvious structures like curves or separate faces are to be aligned [8, 80]. Referring to [47], the *gradient difference* as well as the *pattern intensity* methods were used to measure similarities. Additional information about similarity measures and their mathematical basics can be found in [62, 75, 47, 27]. For details about the application of similarity measures in this project refer to [30].

To put it in a nutshell, the system will provide a 2D/3D multi-modal intrasubject registration of the human thorax based on voxel properties of CT data and using an interactively initialized algorithm performing a global rigid transformation. The optimization is performed with a gradient difference or pattern intensity similarity measure iterating via the (weighted) best neighbor algorithm.

5.2 Basics of Registration via Digitally Reconstructed Radiographs

Having determined the camera's geometry, any other (virtual) camera can be parameterized to resemble the actual camera. If the same sensors are used in both cameras, the registration procedure could begin right away. However, in this project the two image modalities are different regarding sensors, i.e. one computed tomography and one X-ray sensor were used. Thus, the cameras must be adjusted not only by their geometry but also by their physical characteristics. This can be accomplished by modeling the rays emitted by the X-ray source of the C-arm and applying them as well as the right colormap (adapting the grey scales of C-arm images) to the volumetric CT data. Now, taking a picture of the CT volume with the physically and geometrically adjusted camera would result in an image equal to one recorded with the C-arm. What remains is to find the exact alignment of both pictures and the two modalities are registered. A technique to model X-rays is the *direct volume rendering* approach that is actually using the same principles as X-ray machines transmitting rays from the camera center through the image plane and detecting the object's color and extinction. Principles of (direct) volume rendering, their numerical approximation, and some efficient implementations followed by the basics of rigid registration are introduced below. Relying on this section, the chosen method and its application used in this project will be presented in detail in section 5.3.

5.2.1 Volume Rendering

Volume Rendering is the *representation*, *manipulation* and *illustration* of volume data[11]. The gist of volume rendering is how to deal with volume data in discrete form, e.g. arbitrary slices, when visualizing it. The main problem is to offer volumetric information where raw volume data does not provide it. For instance, CT raw data just consists of several two-dimensional slices with arbitrary distances. These are to be rendered such that even *between* those slices the volume can be visualized.

Mathematically, volume rendering can be described as follows:
Given is an unnormalized *mesh* or *grid* representing raw volume data. This can be written as a function $f : \mathbb{R}^3 \mapsto \mathbb{R}$ with a *discrete* domain $D \subset \mathbb{R}^3$ and codomain $S \subset \mathbb{R}$ assigning to each vector $d \in D$ a scalar value $s \in S$. For making the volume data continuous and thus providing a "fluent" impression of the 3D shape, the volume is normalized resulting in voxels with a cuboid shape representing an atomic graphical entity. This can be accomplished by interpolating the sampling points given by D. CT data, for instance, can be normalized by creating slices with equal distances via interpolation. The outcome is called a *normalized mesh* or *regular grid*. Now, to obtain the scalar value of an arbitrary point p in 3D space that lies within the volume's boundaries, an interpolation has to be performed computing the scalar value that belongs to p from the surrounding voxels. Hence, for rendering a volume a normalization step has to be prefixed and an interpolation function must be provided.
In general, two volume rendering techniques are distinguished. The first, indirect volume rendering, transforms the regular grid into a *surface representation* while the second, direct volume rendering, directly works on the normalized mesh.

Indirect Volume Rendering Indirect volume rendering techniques compute an intermediate representation of the grid and thus indirectly manipulate and illustrate the volume. The transformation into this intermediate representation causes a loss of information compared to the original unstructured volume, which is accepted due to higher performance. One technique for indirect volume rendering shall be pointed out, the *Marching-Cubes-Technique* for medical volumetric data consisting of slices [39]. Here, a triangle surface is computed plus the normals of all vertices at each triangle (for quality reasons). For surface creation the user must define a threshold – mostly, a grey value. Then, logical cubes are built in the volume (made of eight pixels, four on each of two adjacent slices). The intersection of surface and cube is determined by evaluating which pixel has a higher (or equal) value than the threshold; those pixels lie inside (or on) the surface and are assigned the flag value ONE, whereas the other pixels are assigned a flag value ZERO. In a second step, the exact nature of the intersection is determined and triangles are thus created by interpolation (in the original case, linear) plus the normals of the vertices enclosing all triangles. Naturally, also more than one user-defined value can be applied to the cube-marching technique (*multi-thresholding*). In order to obtain images with good quality, many triangles are needed to describe the surface. Thus, performance decreases significantly. However, many optimization techniques have been developed to ensure high-performance good-quality rendering [12].

Indirect volume rendering methods are not quite suitable for creating Digitally Reconstructed Radiographs because they only rely on defining a threshold and assigning region values to pixels. However, the rendering technique should be able to simulate the rays emitted by an X-ray source in order to make the DRRs as real as possible. An applicable technique to accomplish such X-ray simulations is *direct volume rendering*, which is thus used to create Digitally Reconstructed Radiographs.

Direct Volume Rendering Direct Volume Rendering displays 3D volume data by modelling emission and absorption of energy (e.g. light), i.e. simulates the physics of visualization. All its computations are performed on normalized grids, i.e. 3D voxel data, in contrast to the radiosity technique [16]. Hence, it can be adequately applied to CT data, which nor-

mally consists of voxels that can be equipped with emission and absorption properties. The goal of direct volume rendering is to visualize a 3D volume continuously unlike indirect volume rendering computing discrete values for e.g. triangle vertices. This continuity can be accomplished by displaying the volume as an entire data block where also the inner voxels contribute to the final image to be displayed [19]. Some benefits result when using this technique. From the 3D volume, all data is considered so a loss of details can be avoided. By considering inner voxels, the inner structures of a volume are also rendered. However, also some disadvantages occur when rendering a volume directly. Since all voxels must be scanned the performance is highly dependent from the size of the volume data. This, however, can be compensated by using advanced, sometimes hardware-accelerated displaying and computation methods to render a volume. Occasionally, the rendered volume is quite blurry such that structures cannot be distinguished any more.
For rendering, a relationship between display and volume has to be established via a transformation [79]. Two approaches are distinguished, *image order* and *object order techniques*. The former uses a kind of back-projection, i.e. (virtual) rays are sent through all pixels of the image and all the voxels that are intersected contribute to the pixels' values. The latter performs a forward-projection, i.e. each voxel of the object projects on the image plane and contributes to the values of the pixels that are hit. Only one image order technique is discussed in detail (see section 5.2.1.1) since it is used in this project to generate DRRs. After the transformation, by assigning color and transparency values to the data elements, evaluating all voxels that contribute to a pixel's value, and composing these voxels' values, an image can be generated.

The following sections will go further into detail of a particular image order technique, *ray casting*, and one of its efficient implementations using hardware acceleration.

5.2.1.1 Ray Casting

As mentioned above, the ray casting algorithm emits rays from the view center through each pixel of the image plane and detects the voxels that are intersected, whose values then predict the final value of the pixels. Unlike ray tracing, ray casting just follows the primary rays and thus, in its basic implementation, does not consider light and shading properties. However, later realizations of the algorithm do model shading as described in [13, 38].

The Volume Rendering Integral For modelling the ray casting mathematically, the ray is first defined as a line with parameter λ, $x(\lambda)$. Now, to each point in 3D space p (approximated by a voxel) two values can be assigned, representing the emission and absorption of light up to this point. These are determined by functions measuring the color and extinction density of p, $\mathbf{color}(p)$ and $\mathbf{extinction}(p)$, which are presumed to be given for each point. The units of the functions are color density per length and extinction factor per length respectively. Now, the color and extinction *up to* each point p can be modelled by the *volume rendering integral*, which also serves as a base for all other direct volume rendering techniques:

$$I = \int_0^D \mathbf{color}(x(\lambda)) exp\left(-\int_0^\lambda \mathbf{extinction}(x(\lambda'))d\lambda'\right) d\lambda,$$

having D as maximum distance where the color density is zero. The integral describes the color that is emitted up to the point $p = x(\lambda)$ *attenuated* by the integrated extinction coefficient between the viewpoint and the point of emission. For an extensive explanation about the volume rendering integral refer to [61, 28, 13].

Classification and Shading As mentioned above, a point in 3D space is expressed with a scalar value, i.e. not a real subspace of $\mathbb{R}^3$ but a *scalar field* is rendered. Thus, as every point in 3D space is mapped to a scalar value, we also must transfer the point's properties (color and extinction) to the corresponding scalar value. Let $s \in \mathbb{R}$ be the scalar value attached to a point $p \in \mathbb{R}^3$. Then, $\mathbf{color}(p)$ and $\mathbf{extinction}(p)$ are formulated as $\tilde{c}(s)$ and $\tau(s)$. $\tilde{c}$ is called the *primary color* of s, whereas τ is called the extinction coefficient. This step is known as *classification*. With this transformation, the volume rendering integral can be written as:

$$I = \int_0^D \tilde{c}(f(x(\lambda)))exp\left(-\int_0^\lambda \tau(f(x(\lambda')))d\lambda'\right)d\lambda, \tag{5.1}$$

where $f(x)$ assigns a scalar value to each point in 3D space as defined above.
For a realistic rendering, punctiform light sources are to be modelled plus the reflections of objects in a scene (light scattering) [16]. This step is called *shading* and is omitted since X-ray systems do not scatter light and only one punctiform light source is considered, the X-ray source.

As mentioned above, a 3D volume is approximated by a finite number of voxels. Hence, for an arbitrary point $p \in \mathbb{R}^3$ $f(p)$ has to be interpolated. For more details about this interpolation refer to [13].

Numerical Approximation of the Volume Rendering Integral For approximating an integral, the Riemann sum is mostly applied. Thus, dividing $x(\lambda)$ into n equal ray segments of length $d = D/n$ following approximations will be used:
The exponent (responsible for attenuation) in equation (5.1) can be approximated by

$$\begin{aligned} exp\left(-\int_0^\lambda \tau(f(x(\lambda')))d\lambda'\right) &\approx exp\left(-\sum_{i=0}^{\lambda/d}\tau(f(x(i\,d)))d\right) \\ &= \prod_{i=0}^{\lambda/d} exp\left(-\tau(f(x(i\,d)))d\right) \\ &= \prod_{i=0}^{\lambda/d}(1-\alpha_i), \end{aligned}$$

where the *opacity* of the i-th ray segment is approximated by

$$\alpha_i \approx 1 - exp\left(-\tau(f(x(i\,d)))d\right),$$

or further approximated by

$$\alpha_i \approx \tau(f(x(i\,d)))d.$$

Color $\tilde{C}_i$ is approximated by $\tilde{C}_i \approx \tilde{c}(f(i\,d)))d$. Then, equation (5.1) can be approximated by

$$I \approx \sum_{i=0}^{n} \tilde{C}_i \prod_{j=0}^{i-1}(1-\alpha_j). \tag{5.2}$$

Now, composing the ray segments in *back-to-front* order, following algorithm implements equation (5.1):

$$\tilde{C}'_i = \tilde{C}_i + (1-\alpha_i)\tilde{C}'_{i+1},$$

where $\tilde{C}'_i$ is the accumulated color in the i-th ray segment.[4]

5.2.1.2 Fast Volume Rendering via 2D / 3D Texture Mapping

Software implementations of the ray casting algorithm are mostly quite inefficient, thus direct volume renderer tend to avail of hardware-based solutions. Besides others, one approach is fancied throughout the science community – rendering via texture mapping. To map a texture means to mantle a certain image (or more copies of it), the texture, over a surface in 3D space. This can be accelerated by a Graphical Processing Unit (GPU) by providing a *texture memory* that can be efficiently used for the mapping. Basically, two methods are distinguished that differ in the dimensionality of the texture memory on the GPU. Earlier texture memories provided a two-dimensional map whereas recent GPU provide three-dimensional maps. The common base of both methods is the covering of 3D surface data with textures. Taking a triangle as example, the procedure will be described shortly [34]. A triangle consists of three vertices, v_1, v_2, v_3 and three edges $e_1 = (v_1, v_2), e_2 = (v_1, v_3), e_3 = (v_2, v_3)$. Now, to describe a point p *on* the triangle, a linear combination of basis vectors can be used. Since triangles are planar two vectors are sufficient to describe p. Now, if two edges of the triangle serve to describe the linear combination, p can be described as

$$p = \lambda e_1 + \mu e_2,$$

where $\lambda, \mu \in [0..1]$ can be interpreted as *texture coordinates*, determining the point on the texture map that defines p's color. For each point on the triangle the texture coefficients λ, μ are determined and the respective color is assigned resulting in a triangle covered with the image. However, one step must not be ignored. How can a texture map with *integer* pixel coordinates provide color information for coefficients λ, μ that are *real*? Naturally, by interpolation, which can be accomplished bilinearly in the planar case or trilinearly in 3D texture maps. Hence the better quality of images rendered with 3D texture mapping. For a closer look into texture mapping and its mathematical foundations refer to [26]. Another difference between the two texture mapping methods, 2D and 3D, is the *point of time* when the texture map is applied to the triangles of a surface. 2D texture mapping covers the triangle *after* the projection on the image plane whereas 3D texture mapping covers them *before* the projection. Transferring the texture mapping approach to medical, in particular CT data, the basic idea is to take separate slices as texture maps to render the volume. Clearly, GPUs with 3D texture mapping can store all slices at a time in the mapping memory whereas GPUs with 2D

[4] This is a recursive algorithm running from n to 0. Trying to set up an algorithm going from 0 to n, i.e. in front-to-back order does not make sense in this context.

texture mapping can just store one slice at a time in mapping memory[5]. In fact, 3D texture mapping hardware creates the slices out of the CT volume data perpendicular to the viewing direction and renders the volume by interpolation (*viewport aligned slices*, see figure 5.1, left). 2D texture mapping hardware cannot store the whole volume in the texture memory.

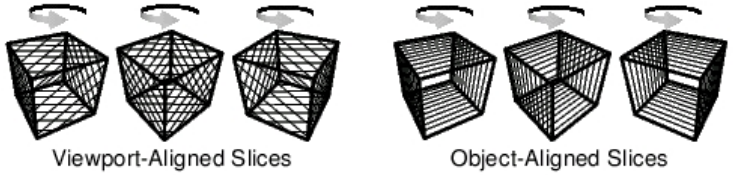


Figure 5.1: Viewport-aligned slices (left) and object-aligned slices (right) for a spinning volume, from [54]

Thus, it creates three stacks of slices in RAM, each containing slices perpendicular to one coordinate axis (*object aligned slices*, see figure 5.1, right). When the viewpoint is determined, slices of the stack with the smallest angular deviation from the view direction are loaded into texture memory, interpolated, and composed in back-to-front order. Hence, when rotating the volume in a viewer, 2D texture mapping hardware always switches stacks when 90 degrees have passed and the fluent impression of the rotation stagnates. Moreover, as 2D texture maps just interpolate bilinearly and thus lack spatial interpolation visual artifacts can occur in the rendered volume. These can, however, be reduced by scaling the opacity (transparency) value of pixels [54]. For a very extensive discussion about volume rendering via texture mapping refer to [53].

As a matter of fact there are much more methods to implement direct volume rendering efficiently like *shear-warp factorizations* or – also used in medical 2D/3D registration – *interpolation of training images*. However, these methods will not be included in this work since the system created DRRs via the texture mapping approach.

5.2.2 Rigid Registration

Transformation A rigid registration is based on a rigid transformation transferring one image according to the other such that both images are aligned. In other words, the coordinate system of one image modality is transformed into the coordinate system of the other modality. This can be accomplished, as already mentioned in chapter 4 by a *rotation* followed by a *translation*.
Mathematically, given two coordinate systems A and B, a point P_A expressed with the basis of the first coordinate system can be written in the basis of the other coordinate system. This can be put in a transformation expressing P_A in the coordinates of B:

$$P_B = RP_A + t, \tag{5.3}$$

where R is an orthonormal *rotation* matrix and t a *translation vector* [36, 37]. As a matter of fact, only the view direction is changed by this change of basis applied by equation (5.3)

[5]Recent GPUs have more than one texture memory such that more slices can be stored in texture memory in the two-dimensional case also

whereas the pose of the point remains unchanged [56]. The rigid registration thus requires two sets of points *with the same dimensionality* and can then transform every point of one set to the coordinate system of the second set.
The translation vector can be expressed in the dimension of the image modality, here, $t = (t_x, t_y, t_z)^T$. The rotation is represented by an orthonormal 3×3 matrix in this work that can be formulated arbitrarily. This rotation describes three degrees of freedom and is most commonly written using *Euler angles*, *unit quaternions* or *rotation axis and angle*. There are much more mathematical representations of rotations like the Pauli spin matrices, Euler parameters (quaternions), or Rodrigues parameters (Gibbs vector). It depends on the purpose of the application the representation is used for, i.e. each representation has its advantages and disadvantages. This project just used Euler angles, which will be described shortly. A three-dimensional rotation can be represented by three angles ϕ, ξ, χ. Each angle is interpreted as a rotation around one of the coordinate system's axis, x-, y-, and z-axis[6]. In decomposed notation, a rotation in 3D space is described by the three 3×3 matrices R_ϕ, R_ξ, R_χ:

$$R_\phi = \begin{pmatrix} 1 & 1 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{pmatrix} \quad \text{(rotation around } x\text{-axis)},$$

$$R_\xi = \begin{pmatrix} cos(\xi) & 0 & -sin(\xi) \\ 0 & 1 & 0 \\ sin(\xi) & 0 & cos(\xi) \end{pmatrix} \quad \text{(rotation around } y\text{-axis)},$$

$$R_\chi = \begin{pmatrix} cos(\chi) & sin(\chi) & 0 \\ -sin(\chi) & cos(\chi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{(rotation around } z\text{-axis)}.$$

The three matrices can be composed by multiplication $R_\phi \cdot R_\xi \cdot R_\chi$, which is a rotation about the z-axis followed by a rotation about the **new** y-axis and one about the **new** x-axis and that results in the following rotation matrix R[7]:

$$R = \begin{pmatrix} cos(\xi)cos(\chi) & cos(\xi)sin(\chi) & -sin(\xi) \\ sin(\phi)sin(\xi)cos(\chi) - cos(\phi)sin(\chi) & sin(\phi)sin(\xi)sin(\chi) + cos(\phi)cos(\chi) & sin(\phi)cos(\xi) \\ cos(\phi)sin(\xi)cos(\chi) + sin(\phi)sin(\chi) & cos(\phi)sin(\xi)sin(\chi) + sin(\phi)cos(\chi) & cos(\phi)cos(\xi) \end{pmatrix}.$$

The term "new" means in this context that after the rotation around one axis (which is actually a change of basis) the coordinate system has changed. Thus, after the first rotation about the z-axis, x- and y-axis are altered to x'- and y'-axis. The following rotation is one about the y'-axis resulting in x''- and z'-axis. The third rotation is one around the x''-axis. In fact, this representation of Euler angles is sometimes referred to as Cardanian angle representation. Euler angle representation actually involves a repetition of rotations about one particular axis. While Cardanian angle representation cover the sequences xyz, xzy, yzx, yxz, zxy, and zyx, Euler angle representation cover xyx, xzx, yxy, yzy, zxz, and zyz. However, rotating around three axis or around two where one is repeated results in the same rotation. It is only

[6] In fact, the rotation is not performed around a coordinate axis unless the object's center is in the origin. Thus, the rotation axis are called *rolls* in some texts. However, we assume that the center is located in the origin and refer to axis.

[7] The order of the factors R_ϕ, R_ξ, and R_χ is arbitrary but has to be kept once it was chosen

important to keep one representation and order for following rotations. According to [60] and [76] there are some disadvantages when calculating with Euler angles. For instance, a singularity occurs if $\xi = \frac{\Pi}{2}$, i.e. rotating about $\frac{\Pi}{2}$ on the y-axis causes the x-axis to turn into the z-axis. Since the new x- and z-axis are parallel, one degree of freedom is lost for further rotations (*Gimbal Lock*). Moreover, problems can arise when interpolating a rotation (which is mostly used in computer animation) because a rotation via Euler angles is not unique. To avoid these disadvantages unit quaternions could be taken. However, the Euler angle notation is minimal, i.e. it expresses the three degrees of freedom of a rotation with three parameters whereas unit quaternions have four parameters. Since a registration just requires one rotation that does not have to be interpolated and, furthermore, as Euler angles are a minimal representation of a rotation, it is satisfactory for rotation notation.

Optimization Given an initial transformation formulated in six parameters $(t_x, t_z, t_y, \phi, \xi, \chi)$, the parameters can be optimized with an iterative algorithm based on an error function evaluating the quality of the parameters. In this project, as already mentioned in section 5.1.3, a gradient difference as well as a pattern intensity similarity measure combined with the (weighted) best neighbor iterative algorithm was used to determine the optimized parameters. The similarity measures will be described in the following.
As the nature of the registration is voxel property based, the similarity measures work on the pixel/voxel intensity. Keep in mind that, for testing the quality of the parameters, two images must be compared. This can only be accomplished when the CT data has been re-rendered into a DRR and a 3D/2D projection with the current six parameters has been performed.
Given two images **A** and **B**, let I_A and I_B be the intensity values of images **A** and **B**. Referring to [47] the gradient difference method first transforms I_A and I_B by differentiation in both directions, x- and y-direction (horizontal and vertical):

$$I_A^{gradhoriz}(i,j) = \frac{dI_A}{di},\ I_A^{gradvert}(i,j) = \frac{dI_A}{dj},$$

and for I_B respectively. This is done by applying a filter kernel to the image, like a Sobel filter. The result can be used to compute magnitude and direction of edges appearing in the images. Now, a difference image is created by subtracting the gradient images of **A** from the gradient images of **B**:

$$I^{diffhoriz}(i,j) = I_A^{gradhoriz}(i,j) - sI_B^{gradhoriz}(i,j),$$
$$I^{diffvert}(i,j) = I_A^{gradvert}(i,j) - sI_B^{gradvert}(i,j),$$

where s is a suitable scale-factor. With this information, the similarity measure $G(s)$ can be computed by using an asymptotic function $1/(1+x^2)$. This should give the measure a certain robustness to thin lines, i.e. if the differences in pixel intensity is large:

$$G(s) = \sum_{i,j} \frac{A_h}{A_h + (I^{diffhoriz}(i,j))^2} + \sum_{i,j} \frac{A_v}{A_v + (I^{diffvert}(i,j))^2},$$

where A_h and A_v are the variances of the gradient images of **A**. The gradient difference similarity measure inherently compares the direction and magnitude of the edges resulting from the differentiation of both images.

The pattern intensity similarity measure also works on a difference image but without determining the gradient before.

$$I^{diff} = I_A - sI_B.$$

Naturally, if the images are exactly aligned, the difference must be zero in all pixels. However, when the alignment is not perfect I^{diff} shows a number of structures or patterns. The number of patterns is determined by regarding each pixel, which belongs to an own structure if it has a significantly different intensity value from its neighboring pixels within a radial distance r. A measuring function will reach its maximum when the number of patterns tend to zero and will itself tend to zero when the number of structures increases. This asymptotic behavior makes the measure robust to large differences in pixel intensity. Moreover, a constant (σ) weights the function in order to filter out small deviations caused by noise. The measuring function is given by

$$\begin{aligned} P_{r,\sigma}(s) &= \sum_{i,j} \sum_{d^2 \leq r^2} \frac{\sigma^2}{\sigma^2 + \left(I^{diff}(i,j) - I^{diff}(v,w)\right)^2}, \\ d^2 &= (i-v)^2 + (j-w)^2. \end{aligned}$$

5.3 Registration of CT Volume Data with 2D C-arm Images

For registration of CT and C-arm data, a suitable user interface had to be conceived in order to allow the user to initialize the registration parameters t_x, t_y, t_z, and ϕ, ξ, χ. Based on amira, a module was programmed that provides viewers for all image modalities, an "overview" viewer showing a segmented CT surface model and the X-ray image as well as suitable sliders and buttons for all required functionality needed for registration. Figure 5.2 shows the three viewers displaying the real X-ray image as recorded by the C-arm in the upper left, the artificial X-ray image (DRR) as constructed by volume rendering out of CT data in the upper right, and a segmented CT surface model plus, again, the real X-ray image (for orientation reason) in the lower viewer. The two upper viewers are initialized with the height angle as depicted in chapter 4. Thus, both images, the real and the artificial X-ray image can be seen from the actual C-arm's perspective. Figure 5.3 shows the user interface for the registration module providing sliders for translation and rotation in each direction as well as a transparency slider to adjust either the DRR or the 3D surface model. Moreover, three action buttons offer functionality for starting the automatic optimization of the registration, back-projecting a point marked in the real X-ray image to the CT volume data, and displaying just the surface model for viewing the actual result – the back-projected stent in the 3D reconstruction. Furthermore, a field is provided where the size of the stent that is to be implanted can be put in and a slider of port **Catheter depth [cm]** can then move the stent after back-projection up and down the aorta. Thus, the binding between the two subsystems for planning and navigation just relies on the computed stent graphic but not on calculated metrics (see section 3.5.3) etc., so coupling is minimized.

Transferring all theoretical background of section 5.2 into the proposed application, the steps that are to be taken for registration were depicted as follows.
First, a DRR is created via direct volume rendering using 2D or 3D texture mapping that displays the CT data in a viewer window (see section 5.3.1). Then, according to the activity

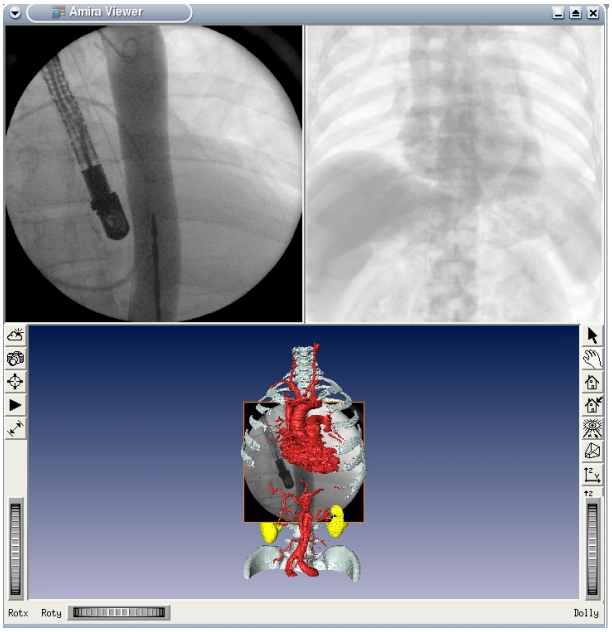


Figure 5.2: Three viewers of registration interface in amira, the upper left shows the real X-ray image from the C-arm, the upper right the artificial X-ray (DRR) and the lower the segmented CT surface model and again the real X-ray image

diagram shown in figure 5.4 an initial transformation is determined by using the real X-ray image as reference and applying transformations to the rendered CT volume. This will be described more thoroughly in section 5.3.2. The result are initial values for t_x, t_y, t_z and ϕ, ξ, χ. Afterwards, a snapshot of the viewer displaying the DRR is taken in order to obtain a 2D image of the artificial X-ray image. The similarity measure is computed as described in 5.2.2 comparing the real X-ray image and the artificial one. If the quality cannot be improved any more the algorithm terminates by returning and applying the optimized parameters of t_x, t_y, t_z and ϕ, ξ, χ (refer to section 5.3.3). Otherwise, the six parameters are altered and another snapshot is taken leading to yet another iteration. After finishing the optimization, the parameters are applied to the segmented 3D surface model, the stent is marked in the real X-ray image and back-projected in the model. This part will be presented in detail in section 5.3.4.

5.3.1 Creating Digitally Reconstructed Radiographs

Fortunately, amira provides a volume rendering module (*HxVoltex*, see section 3.3.1.2) doing direct volume rendering via 2D/3D texture mapping as described in section 5.2.1. Naturally, better results can be achieved when rendering in 3D texture mode, however, since the transformations applied to the DRR in this system do not have to be very fluent and artifacts just emerge when zooming into the volume extensively, the 2D mode can also be worked with. Most important is the application of the right colormap to the volume rendering module in order to adapt the grey scales of the C-arm. A colormap is attached to the volume rendering module mapping the right values to the range of grey values of the CT data via interpola-

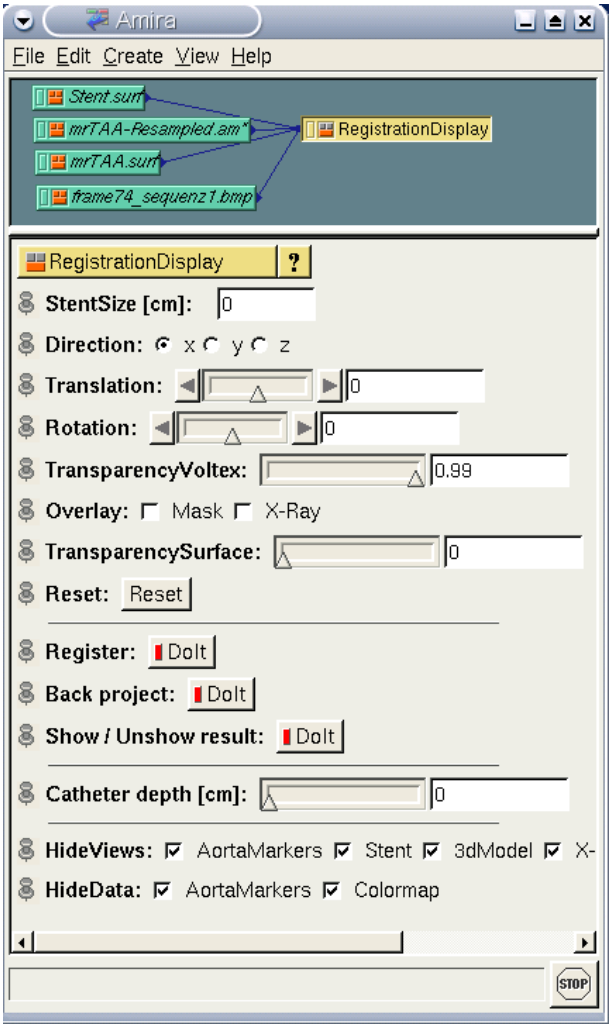


Figure 5.3: The User interface of the registration module

tion. The value range was determined by specifications from the manual of the C-arm GE MS OEC9800 and by talks with radiologists at the DHM.

5.3.2 Interactive Initialization

Referring to figure 5.3 the interactive initialization was accomplished by changing the sliders of the ports **translation** and **rotation** in combination with the radio boxes of port **direction** determining the axis of the transformations. When moving these sliders the transformation is applied to the CT data, i.e. both representations of the data, DRR and 3D surface model are transformed. Actually, the z-axis of the coordinate systems in the viewers directs right to the user, i.e. perpendicular to the user's screen. Thus, when translating in z-direction, the volumes can be scaled or the region of interest may be changed. Furthermore, the transparency of both volume representations can be changed arbitrarily by altering the sliders of the ports **transparencyVoltex** and **transparencySurface**. As all C-arm images have a circular shape

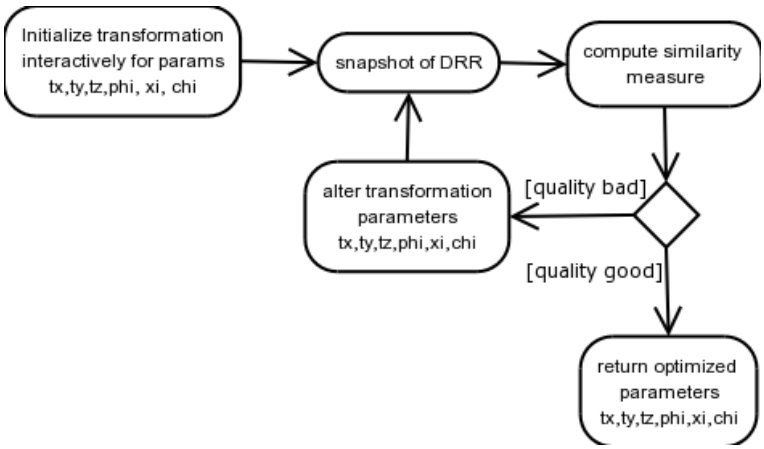


Figure 5.4: Activity diagram of the registration procedure of the CANP system

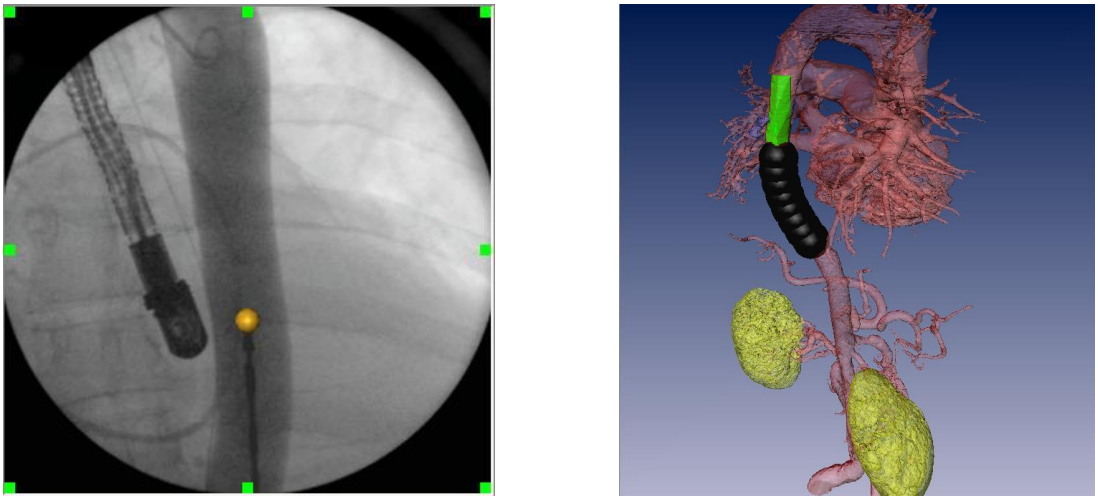
a mask is provided that can be drawn over the DRR viewer resembling the black frame in an X-ray image. Thus, the visual and automatic alignment can be improved. Moreover, for testing the quality of the initial registration visually, the X-ray image could be displayed in the DRR viewer as well, which had a transparent rendering mode and allowed to view DRR and X-ray image overlaid. These two functionalities are offered by the two toggle boxes at port **Overlay**.

5.3.3 Automatic Optimization

The automatic optimization can be initialized by pressing the **Register** action button. As a preliminary step, the region of interest may be restricted by changing the size of a rectangle bounded by green squares as showed in figure 5.5(a). This causes the algorithm to confine to boney structures. Then, the loop as described by figure 5.4 is executed. Naturally, altering the parameters is only applied to the DRR representing the CT data since the similarity measure is evaluated for the DRR, not for the 3D surface model. The algorithm stops at a minimum or maximum of the similarity measures, i.e. when the similarity measure does not decrease – in case of gradient difference – or increase – in case of pattern intensity – by changing any of the parameters t_x, t_y, t_z and ϕ, ξ, χ. The algorithm was implemented using *best neighbor* and *weighted best neighbor* iterative approaches respectively. After termination, the optimized parameters are applied to the 3D surface model, too. For a detailed description of the outcome of this optimization, i.e. the comparison and evaluation of both iteration algorithms with gradient difference and pattern intensity similarity measure respectively, refer to [30].

5.3.4 Backprojection of Stent Location into CT data

When the optimization algorithm is finished, the stent (only visible in the real X-ray image) has to be marked by the user as shown in figure 5.5(a). The size of the landmark with which the stent is marked in the X-ray image can be adjusted such that the desired precision of



(a) (b)

Figure 5.5: Back-projection of the stent; (a) the stent marked in the original X-ray image with a golden sphere. The green squares limit the region of interest; (b) the stent back-projected to the 3D surface model. The green surface shows the planned stent, the black sphere line shows the actual stent

the back-projection is not influenced by visualization. Afterwards, the button **Back project** can be pressed, which causes the back-projection of the marker that was set in 2D space to the CT data in 3D space. Since both representations of the CT data have the same transformations the back-projection is now only applied to the 3D surface model rather than to the DRR. This can be accomplished by creating a line out of two points, the camera center of the C-arm (taken from the viewer on the upper right) and the marker placed on the stent. Then, a collision detection of line and 3D surface model is performed and the intersection point nearest to the aorta is determined. This can only be achieved by measuring and minimizing the distance between the aorta line set given from the planning procedure and the intersection points. It is accomplished by determining the orthogonal projection of each point to each line (see section 4.2.5.2) of the line set and computing the Euclidean distance between point and projection point. Afterwards, the nearest point is displayed as a black sphere with a radius of one centimeter. From there, more spheres are attached in distances of one centimeter to draw the length of the stent according to the value of port **Stentsize [cm]**. Now, the planned as well as the real position of the stent can be visualized as shown in figure 5.5(b). Port **Catheter depth [cm]** offers a slider, which moves the stent along the aorta line set when altered. Thus, by moving it exactly to the planned position of the stent, i.e. overlaying current and planned position interactively, the distance between planned and current stent location can be determined.

6 Conclusion

The system as conceived enables the user, i.e. the physician, to plan a stent operation in advance and use this planning, unlike in the current situation, during actual treatment in the operation theatre. Moreover, it can merge information from two image modalities, which may result in an aggregated and extended knowledge for the clinician such that a faster operation is possible with lower risk for both, clinicians and patients. Result and feedback of the project will be given in the next section, including the benefit that physicians might gain with the new system as well as a review of all requirements evaluated whether they have been met or not. In section 6.2 the implementational problems that were encountered while working on this project are highlighted. All advantages and flaws of the systems are argued critically. In section 6.3 possible extensions and further projects based on or tied with the heARt project will be discussed.

6.1 Result

A system was implemented that could be used as an extension to amira, the graphics engine, which is to be introduced at the DHM. The planning procedure could be realized by using already existent modules, by merging, and extending them. Thus, planning a stent implantation is made rather convenient since all image modalities that are available at this point, i.e. axial, sagittal, and frontal CT slices as well as a three-dimensional reconstruction of a volume from CT, can be included. As markers can be set in one viewer but are shown in all four (three), the surgeon can double-check each entered marker and has full visible information about the augmented region of interest.
When it comes to navigation, the physician can utilize the tool to merge CT- and X-ray-information and can view all data acquired, i.e. planned and current position of the stent as well as anatomical details in the aorta region, in one three-dimensional model, the 3D surface model. An interactive initialization must be performed after which an automatic optimization computes a best alignment. The registration optimization can be conducted via weighted or unweighted best neighbor iterative algorithms evaluating the quality either with gradient difference or with pattern intensity similarity measures. This can be repeated for all following X-ray images shot with the C-arm. Moreover, using the saved metrics of the CT model, the physician is informed about the size of the planned stent and the distance of current and planned stent, which might improve treatment considering time. The image overlay could only be achieved by preliminarily determining the C-arm's physical and geometrical characteristics via calibration. The calibration was quite accurate, although it did not consider the specific setup of a C-arm but simply performed the calibration of a CCD camera using commonly known algorithms for parameter calculations.
The project was presented at the DHM at a meeting where all clinicians, operating physicians as well as radiologists could observe the functionality of the system. The feedback was

rather enthusiastic, encouraging a continuation of the efforts for a full-fledged planning and navigation system.

6.1.1 Benefit

The system was implemented to help and guide physicians through the complex procedure of stent implantations. Moreover, X-ray exposure as well as contrast injections were to be reduced. When used properly, the navigation as well as the planning tool aids physicians to accomplish the detection of an aneurysm, the restriction to the region of interest, the confinement of the stent's boundaries, as well as the run of the catheter through the aorta with little complications. In this respect, there is definitely a benefit for clinicians since the gap between using C-arm systems only during operation and extending the operation with views of the patient's interior in 3D plus all information provided by planning and registration is rather big. Ideally, the system could be used in order to avoid injection of contrast agents entirely and reduce the shooting of image sequences with the C-arm to two *single* images[1] – one at the beginning and one at the end of the operation. In the first image the aorta does not have to be visible necessarily since the registration mechanism was designed to perform alignments not on soft tissue but on boney structures. The second image would then just serve for double-checking, i.e. whether the stent is at the right place at the end of the operation.

6.1.2 Requirements Analysis Review

All requirements that were demanded for the system were included. First, the use of fiducials as common in almost all other clinical applications could be totally avoided. Due to amira, a user interface could be conceived that was rather easy to understand and operate. However, a usability testing like a field test was not performed yet and is to be processed in the future. Metrics, as demanded by surgeons were transferred from CT data directly into the system enabling the calculation and determination of distances or sizes. Although the system was not yet used in the operation theatre, the sterility requirement can be met easily since only a monitor and a PC and some input devices (mouse, keyboard) are required. As mentioned above, all functional requirements could be fulfilled in both, the planning and the navigation part. On the one hand, the planning tool can show CT data in all different modalities and offers a convenient landmark-based method to create an approximate stent graphic. The navigation tool, on the other hand, can merge CT and X-ray data and, by using previously planned information like stent graphic and aorta markers, back-projects all determined features from the X-ray image to the 3D surface model. Additionally to the modules for planning and navigation, an extensible calibration and registration framework was programmed offering a convenient calibration of cameras by only providing a file of point correspondences and a set of images. Moreover, methods for forward- and back-projection are included enabling users to test the quality of the calibration.

[1] not sequences as in angiograms since the abandonment of contrast makes image chains useless

6.2 Problems and Discussion

Despite all positive feedback and approval of the project, the system is still a prototype and was not tested or even used in an operation environment as it is intended. Thus, the acceptance of the tool by physicians is not yet evaluated and could lead to further software (re)engineering. Moreover, radiologists would have to switch from a SIEMENS proprietary software doing segmentation tailored for the CT scanner SOMATOM sensation cardiac to a framework (amira) whose many features include one rather basic segmentation tool for which the right colormaps and mappings from Hounsfield units to grey values have to be determined manually for the CT scanner. However, amira is independent from the SOMATOM sensation cardiac such that a replacement of CT scanners would not require to change the software. Due to the standardized DICOM format that can nowadays be produced by almost every CT machine, the software could be linked to any computed tomography device. Furthermore, amira focuses on the functional part not only on the visual part of image acquisition and processing, i.e. while three-dimensional reconstructions created with the Siemens SOMATOM software have a real fancy look, they cannot be rotated or translated in space and clinicians cannot zoom into the graphic or transform it in any way such that it merely serves as a nice visual effect but does not provide the possibility to plan or navigate. Three-dimensional reconstructions at the DHM are fancied by physicians, but actually they are not used in any form for aiding a stent implantation. As a matter of fact, reconstructions produced by the SOMATOM software are movies in .avi format and mostly show a 360° rotation of the 3D model around one axis.

There are several problems not only concerning medical sterility in the operation theatre[2]. For instance, the image sequence of the C-arm GE MS OEC9800 cannot be accessed and processed by the system. They have to be stored intermediately in a bitmap file, which in turn can be loaded by the system. For successfully applying intraoperative navigation, a communication between system and C-arm has to be established and data has to be extracted and directly processed. This would require a co-operation with GENERAL ELECTRIC, which must provide adequate format descriptions and streaming possibilities[3].

Calibrating the C-arm was rather accurate for images recorded stationary, i.e. the C-arm was not moved between two image acquisition steps. However, as can be seen in [56, 64, 58], when moving the C-arm it also changes its intrinsic parameters since it depends on the external magnetic field, i.e. the earth's magnetic field being different at different locations. This was not considered in the system.

Analyzing registration, the time for the automatic optimization is rather long. It takes between one to four minutes, and, if performed more than once during an operation, turns out to be useless due to time constraints. The interactive initialization is also quite error-prone since even physicians cannot – as has been evaluated – exactly say where they look at the patient when an X-ray image shot with the C-arm is given. In other words, it might be possible that ribs are miscounted and the overlay is performed wrongly leading to bad results even with automatic optimization, where the stent is actually back-projected but at the wrong location, which would be futile for navigation.

[2] even though only PC and monitors are used, they have to be spayed as well

[3] Actually, the C-arm contains a PC, which has a network and COM interface that could be used, but the access to the PC is not possible yet

amira is a very powerful tool for image processing not only in medical imaging. Its generality does not influence its application in the medical sector, especially in CT image processing and visualization. However, when extending amira, the problem is the absence of some crucial header files, which makes it almost impossible to conceive reasonable C++ code. There is the possibility to address modules and their ports via the Tcl interpreter but this is not a satisfactory state for application programmers. A recommendation to INDEED VISUAL CONCEPTS[4] is to publish header files especially for compute modules. The amiraDev extension is apparently meant to enable programmers to conceive their own *single* display or compute modules as well as read or write routines for new file formats but lacks one important paradigm of object-oriented programming – reusability. Thus, systems based on some functionality of amira cannot be programmed consistently if reinventing the wheel for certain computations should be avoided.

6.3 Future Work

As was worked out at the meeting after the end presentation of the system at the DHM, the next step to be taken for this project is to move the system into the operation theatre, i.e. perform some testing and actual application in order to gain feedback for the ease of use as well as additional requirements. It was suggested that at first, the system is to be placed and run in the operation room, however, the actual tools for operation shall not be replaced yet. This means that the operation is still performed guided by angiograms shot with the C-arm but a best image is extracted from the sequence and put into the system for registration. However, this would not affect the operation procedure and would just serve as an additional visualization tool, i.e. no decisions are made without counseling the conventional tools. This is due to the fact that before using the system solely and abandoning all other imaging methodologies that have been established for a stent implantation, an ethic committee would have to approve to the system being used. Before this ethic committee shall be consulted, however, the system is to be made full-fledged via testing, integration of error-handling procedures as well as analysis and implementation of behavior in extreme situations, quality of service and security.

One disadvantage of the system is, as mentioned above, the segmentation step that produces a 3D surface model, which is required for planning as well as for registration. It can be accomplished via the thresholding tool and the segmentation editor of amira, but this requires lots of manual work, which could be automated using other segmentation tools or algorithms as described in section 3.2.2. As amira is focused on generality, i.e. serves as a visualization tool that can be used in different fields, the segmentation procedure could not be specialized on medical image modalities. However, a module could be integrated that offers more segmentation techniques like deformable models, classifiers, or atlas-guided approaches, which could reduce the interactive character of building 3D models out of CT data.

One improvement with mere implementational character could be made for the planning procedure. Right now, only the whole planning network can be stored; when restoring it, the `PlanningDisplay` and all attached modules are loaded again. However, this is not

[4] INDEED VISUAL CONCEPTS GmbH located in Berlin, Germany has developed amira

desired when loading the information acquired via planning for navigation. Here, only the stent surface as well as CT data and surface model are necessary. When saving, the planning procedure could thus create a script that can be executed by amira and loads these three data modules for registration.

6.3.1 Calibration

As it is crucial for the system, the calibration procedure to determine the C-arms characteristics could also be improved. As already mentioned, the intrinsic parameters depend on the earth's magnetic field, i.e. whenever the pose of the C-arm changes, a new calibration has to be performed. In some systems this is accomplished by attaching the calibration body rigidly to the C-arm as described in [57, 7]. This would be a necessary step to make the calibration more accurate. Another approach could be to track the C-arm's pose. This would enable the calibration algorithm to automatically determine the right parameters whenever the C-arm moves. However, tracking the C-arm would require additional hardware, i.e. tracker and adequate markers would have to be installed.

The system did not consider any kind of distortion when calibrating the C-arm. Except radial lens distortion, which is not the main factor distorting the image in X-ray-image-intensifier-systems, no distortion has been included. In the GE MS OEC9800 especially spiral distortion as can be seen in the images in appendix B affects the quality of calibration. This could also be taken into account by using the methods for distortion correction described in [64, 58, 65]. However, this topic should not be focused on too extensively since recent developments in fluoroscopic imaging could avoid distortion at all. This is due to replacing the detector plate – so-called *solid state detectors* are used and can abandon geometric image distortion [43].

6.3.2 Registration

Due to the tight time schedule of this project, some considerations have been left out when it comes to registration. Basically, and most importantly, a gold standard algorithm should be developed that could be taken to evaluate the accuracy of the used registration. Different methods have been tried out, similarity measures have been replaced as well as iterative algorithms, and measurements as described in [30] were compared and evaluated. However, an effective comparison to a method that is commonly used for such registration problems is still missing and should be implemented and integrated into the system. For instance, a method using fiducials that are fixed to a phantom is considered to give best results for alignments in some medical sectors like neurosurgery and could probably be used here as well.

Fiducials could be an aid to the manual alignment too. As mentioned above, the interactive registration could easily lead to wrong results due to the small region that is visible on C-arm images. In this case, even experienced physicians can miscount ribs or boney structures and the registration would be useless. Mounting fiducials on the patient that are recorded during computed tomography and X-raying in the operation theatre could function as landmarks to align the two image modalities more accurate and faster. Actually, three fiducials at the right region could suffice to determine translations and rotations interactively. Naturally, the placements of the fiducials must not be altered between preoperative and intraoperative

image acquisition, a requirement that is not easy to meet since a delay of some days between CT scanning and operation must be considered when attaching markers to patients. The fiducials can be rather basic, e.g. plasters treated with metal that are attached to the patient's skin can be useful for an interactive registration initialization.

The similarity between DRR and real X-ray image is actually not satisfactory since contrast could not be modelled in the desired way. As contrast agents flow through vessels, the liquid does not emit the same intensity in images that are recorded via CT or C-arm respectively. In fact, the aorta could not be seen in the DRR whereas it was quite visible in real X-ray images. However, it is possible to model contrasted images as shown in [27], which could possibly cause an improvement for interactive initialization as well as automatic registration. Then, the alignment does not have to be restricted on boney structures any more. However, since vessels are non-rigid, a different registration approach must then be considered [59].

As already mentioned above, tracking the C-arm would help to make the calibration more accurate. This would also be helpful for registration if the patient is tracked as well. Then, the relative pose of patient to C-arm could be determined and inserted into amira. An initial manual alignment could then be left out or at least the degrees of freedom for manual registration initialization could be reduced. Already in the field of tracking, another useful tracking device could improve the system. Tracking the catheter that is used to advance the stent through the aorta could abandon the slider that is used for distance recognition. The distance could automatically be updated and physicians would always be informed about the actual position of the stent when it was registered previously. Since the catheter has in this respect just one degree of freedom, it would be quite easy to extend the system for stent tracking. Tracking patient and C-arm, however, would require sophisticated tracking systems, e.g. infrared or visual tracking with appropriate markers mounted on both, C-arm and patient. Naturally, the requirement of a "slim" system with little hardware devices[5] would have to be abandoned.

The system actually just needs one real X-ray picture to align CT and C-arm images. Up to now, one image was taken from the sequence shot by the C-arm. This was performed manually due to the inability of transferring data directly from the C-arm to a PC, i.e. the sequence was browsed through and an adequate picture was taken. However, this task could also be accomplished automatically – an algorithm could select the best image that is taken for alignment.

The system laid the basis for a tool aiding clinicians to plan and navigate minimally invasive treatments. Although it is still a prototype and not used in the operation theatre yet, it can be taken as a core and reference system for a full-fledged product that guides physicians through the difficult task of implanting a stent graft. One step could be taken from conventional treatment techniques to modern methodologies entirely relying on medical image processing and fusion. Hopefully, this system can impact medicine to move towards the idea of the vitreous patient.

[5] which have to be made sterile in the first place

And since you know you cannot see yourself,
so well as by reflection, I, your glass,
will modestly discover to yourself,
that of yourself which you yet know not of.

WILLIAM SHAKESPEARE

Appendix

A Requirements Analysis Diagrams

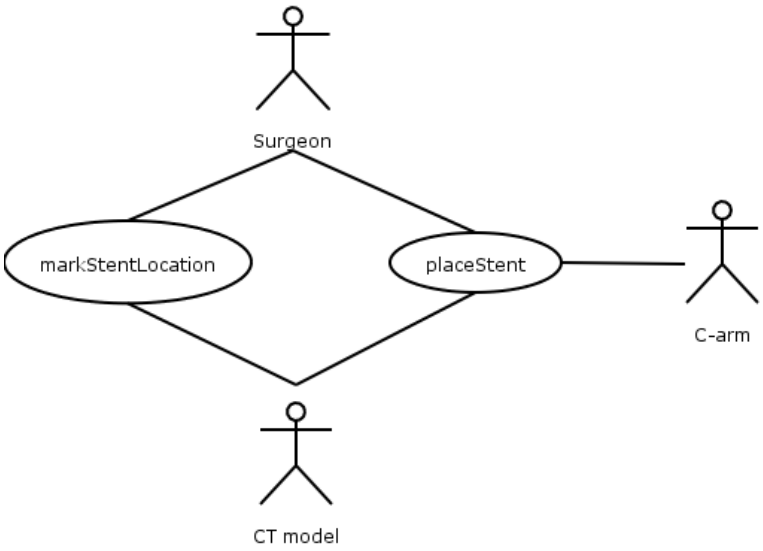


Figure A.1: Use Case diagram of the CANP system

Use Case: **markStentLocation**
Initiated by: `Surgeon`
Communicates with: `CT data`
Flow of Events:

1. The `Surgeon` activates the pre-operational planning mode of the PlanningTool.
2. The `Surgeon` loads the `CT data` of the patient
3. The `Surgeon` examines the `CT data` and plans the location of the Stent.
4. The `Surgeon` marks the position and orientation of the Stent in the displayed 3D model on the screen.
5. The system calculates a graphic of the stent given the formerly defined pose.
6. Now, the `Surgeon` marks the position of the aorta for the registration during the `placeStent` use case.
7. If the planned stent appears to be placed correctly and the aorta markers are set properly, the `Surgeon` confirms them and the additional data is stored in the `CT data`.

Use Case: **placeStent**
Initiated by: `Surgeon`
Communicates with: `C-arm, CT data`
Flow of Events:

1. The `Surgeon` activates the navigation mode of the tool.
2. The `Surgeon` loads the latest fluoroscopic image (given by `C-arm`)
3. After an approximative interactive alignment the Navigation Tool merges the given image with the `CT-data` (registration) and displays the current location of the stent on the one hand and navigational information on the other hand (distance between current and desired stent location).
4. If the current location is not satisfactory, the `Surgeon` starts over with another iteration at step 2.
5. The `Surgeon` can advance the catheter to the position suggested by the navigation tool and can, after testing the location with another X-ray shot of the `C-arm`, expand the stent.
6. The `Surgeon` shuts the system down.

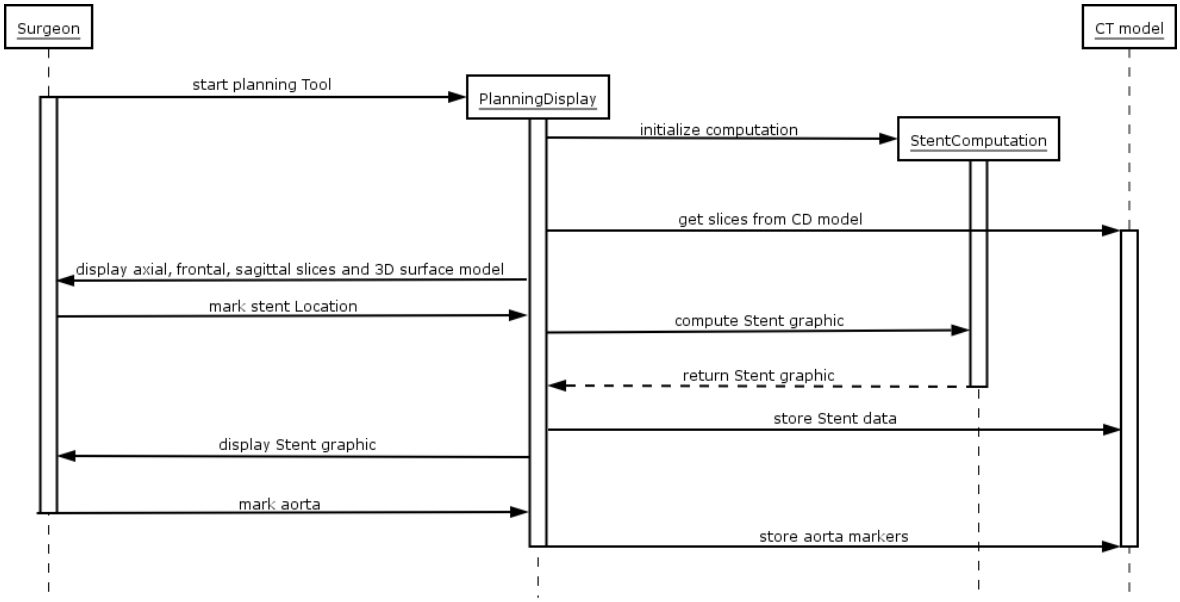


Figure A.2: Sequence Diagram for the Planning system

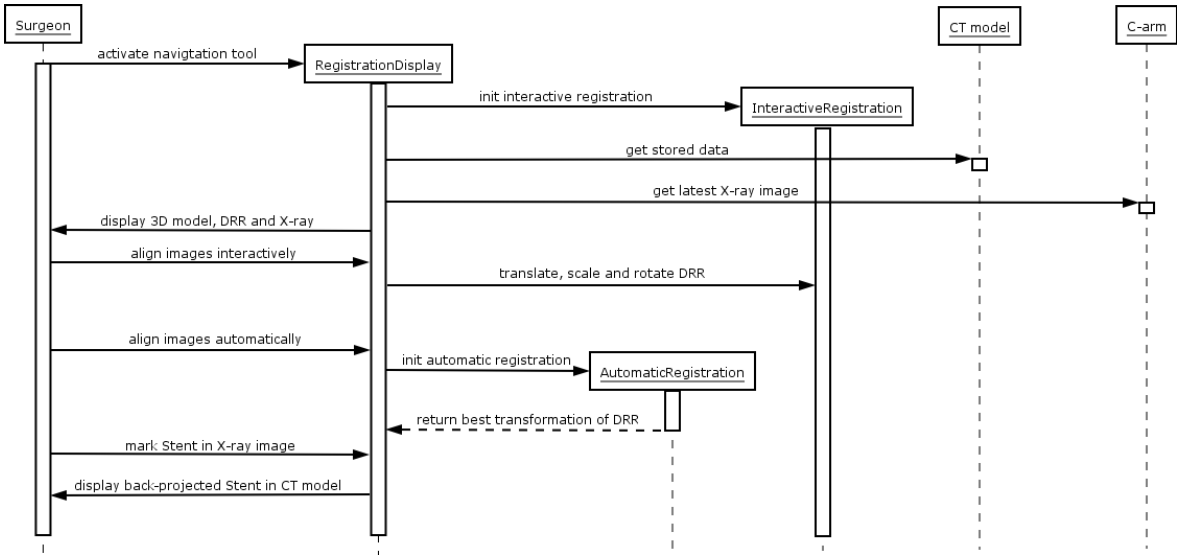


Figure A.3: Sequence Diagram for the Registration system

B Calibration Images and Tables

Intrinsic Parameters of Tsai calibration (not optimized)				
Image	Focal Length f [mm]	(p_x, p_y) [pix,pix]	κ_1 [$\frac{1}{mm^2}$]	intern quality [%]
image 1	12602.00	(562.396, 512.467)	-5.26067e-08	100.0000
image 2	2532.55	(518.809, 628.850)	-8.42893e-08	51.2195
image 4	7.90	(511.324, 478.494)	-5.32897e-06	3.0303
image 5	3921.73	(470.600, 577.050)	-3.90497e-08	65.6250
image 6	4315.10	(518.519, 534.348)	-6.25371e-08	75.6098
image 7	6047.05	(480.036, 713.945)	-9.19423e-08	61.2903
image 8	4211.38	(531.383, 503.437)	-1.08023e-07	58.9744
image 9	3225.17	(816.172, 388.860)	8.26198e-08	96.8750
image 10	4259.88	(517.031, 489.970)	-4.90252e-08	78.1250
image A	3156.06	(592.340, 422.712)	-1.19846e-07	63.6364
image B	3344.52	(749.542, 397.677)	2.84643e-08	90.6250
image F	3905.16	(494.596, 616.884)	-1.1312e-07	50.0000

Table B.1: Intrinsic Parameters calibrated with the Tsai algorithm

Intrinsic Parameters of Tsai calibration (optimized)			
Image	Focal Length f [mm]	(p_x, p_y) [pix,pix]	intern quality [%]
image 1	12646.00	(562.322, 512.501)	100.0000
image 2	16495.00	(465.194, 505.609)	65.8537
image 4	9.75673	(603.975, 375.091)	3.0303
image 5	3951.81	(470.691, 576.972)	65.6250
image 6	4349.98	(518.342, 534.369)	75.6098
image 7	6084.31	(479.855, 713.727)	61.2903
image 8	4248.88	(531.048, 503.308)	58.9744
image 9	4056.96	(818.775, 376.000)	96.8750
image 10	4375.21	(514.207, 490.170)	81.2500
image A	4565.03	(561.893, 425.589)	72.7273
image B	3393.44	(750.249, 396.255)	90.625
image F	4256.84	(489.008, 604.864)	83.3333

Table B.2: Intrinsic Parameters calibrated with the Tsai algorithm and optimized with Weighted Best Neighbor

image001.bmp	(-0.97602, 11.7028, 96.2486)
image002.bmp	(-0.898351, 11.8692, 96.0629)
image004.bmp	(-0.612376, 8.11559, 66.7944)
image005.bmp	(1.99963, 5.81918, 39.3617)
image006.bmp	(1.35593, 7.07312, 59.2942)
image007.bmp	(1.17285, 6.76271, 47.1416)
image008.bmp	(1.73376, 10.1605, 70.5831)
image009.bmp	(2.40311, 9.7521, 85.1956)
image010.bmp	(-5.29069, 9.65666, 61.388)
image00a.bmp	(-0.459797, 11.3921, 95.5382)
image00b.bmp	(0.966861, 8.55749, 84.0719)
image00f.bmp	(2.19817, 6.37961, 37.2605)

Table B.3: Translation vectors calculated with OpenCv

image001.bmp	(-1.77915, -0.02385, 277.856)
image002.bmp	(-0.746358, -2.42704, 56.1281)
image004.bmp	(3.00308, 1.53248, 0.983634)
image005.bmp	(2.49732, 0.422145, 35.571)
image006.bmp	(1.45228, -0.469091, 58.7085)
image007.bmp	(1.67542, -1.19147, 66.2075)
image008.bmp	(1.64734, 1.65489, 68.9734)
image009.bmp	(-3.24704, 1.77062, 62.7747)
image010.bmp	(-6.22142, 3.38413, 45.3796)
image00a.bmp	(-5.33269, 2.23647, 72.8488)
image00b.bmp	(1.13812, -1.27419, 81.7484)
image00f.bmp	(2.45549, 0.904006, 33.7607)

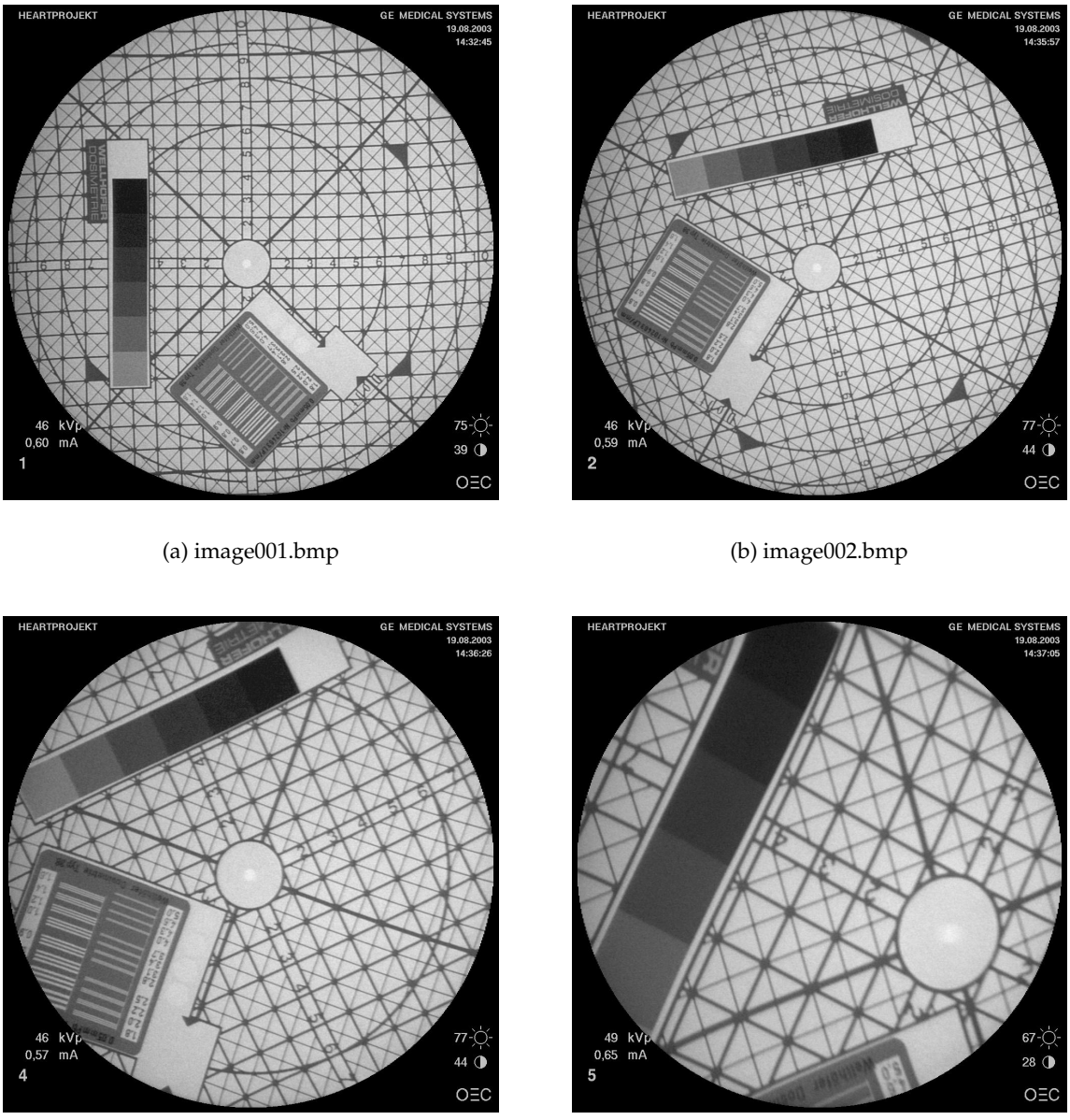
Table B.4: Translation vectors calculated with Tsai

image001.bmp:				**image002.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	-0.00642456	-0.999535	-0.0298107	0.972591	-0.232046	-0.0148583
r_{2j}	-0.999617	0.00561655	0.0271097	-0.232521	-0.970459	-0.0643729
r_{3j}	-0.0269297	0.0299735	-0.999188	0.000518174	0.0660634	-0.997815
image004.bmp:				**image005.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.77651	-0.569701	0.269208	0.414961	-0.806969	0.420248
r_{2j}	-0.576179	-0.814922	-0.0626006	-0.901858	-0.42586	0.0727677
r_{3j}	0.255047	-0.106502	-0.961046	0.120245	-0.4092	-0.904487
image006.bmp:				**image007.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.693588	0.717567	-0.0635096	0.72737	0.652285	-0.213207
r_{2j}	0.666011	-0.672349	-0.323073	0.619825	-0.491111	0.612067
r_{3j}	-0.274527	0.181781	-0.944241	0.294534	-0.577351	-0.761522
image008.bmp:				**image009.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.698053	0.715714	-0.0218099	0.789281	0.501039	0.354959
r_{2j}	0.607626	-0.608197	-0.510771	0.607192	-0.722912	-0.329722
r_{3j}	-0.37883	0.343293	-0.85944	0.091401	0.475771	-0.874807
image010.bmp:				**image00a.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.52071	0.714235	-0.467684	0.837959	0.544342	0.0389479
r_{2j}	0.813799	-0.580834	0.0190324	0.544914	-0.838477	-0.00507037
r_{3j}	-0.258054	-0.390512	-0.883691	0.0298969	0.025472	-0.999228
image00b.bmp:				**image00f.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.494083	0.86815	0.04687	0.673248	0.739103	-0.0215252
r_{2j}	0.65344	-0.406367	0.638656	0.40753	-0.395195	-0.82325
r_{3j}	0.573496	-0.284922	-0.768064	-0.616973	0.545479	-0.567271

Table B.5: Rotation matrices calculated with Tsai

image001.bmp:				**image002.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	-0.00880843	-0.999377	0.0341685	0.972144	-0.232641	-0.0285154
r_{2j}	-0.986444	0.00308497	-0.164069	-0.234288	-0.961089	-0.146347
r_{3j}	0.163861	-0.0351505	-0.985857	0.00664048	0.148951	-0.988822
image004.bmp:				**image005.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.885505	-0.412672	-0.213502	0.410979	-0.802208	0.43308
r_{2j}	-0.443381	-0.887886	-0.122767	-0.875643	-0.479545	-0.0573205
r_{3j}	-0.138903	0.203374	-0.969198	0.253664	-0.355666	-0.899531
image006.bmp:				**image007.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.695719	0.716061	-0.0568528	0.725963	0.653797	-0.21337
r_{2j}	0.626558	-0.643651	-0.439475	0.660432	-0.576193	0.481489
r_{3j}	-0.351284	0.27013	-0.896454	0.191854	-0.490459	-0.850084
image008.bmp:				**image009.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.698123	0.715696	-0.0200562	0.792078	0.526956	0.308107
r_{2j}	0.557549	-0.561006	-0.611891	0.610323	-0.674658	-0.415142
r_{3j}	-0.44918	0.415993	-0.790688	-0.0108947	0.51687	-0.855995
image010.bmp:				**image00a.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.516902	0.708775	-0.480053	0.838369	0.539325	-0.0791595
r_{2j}	0.780595	-0.620454	-0.0755574	0.53588	-0.842048	-0.0615483
r_{3j}	-0.351404	-0.335671	-0.873979	-0.0998505	0.0091802	-0.99496
image00b.bmp:				**image00f.bmp:**		
	r_{i1}	r_{i2}	r_{i3}	r_{i1}	r_{i2}	r_{i3}
r_{1j}	0.494413	0.867968	0.0467602	0.678095	0.734861	-0.0129025
r_{2j}	0.716148	-0.437241	0.544015	0.315745	-0.307118	-0.897766
r_{3j}	0.492633	-0.235481	-0.837772	-0.663696	0.604697	-0.440284

Table B.6: Rotation matrices calculated with OpenCv



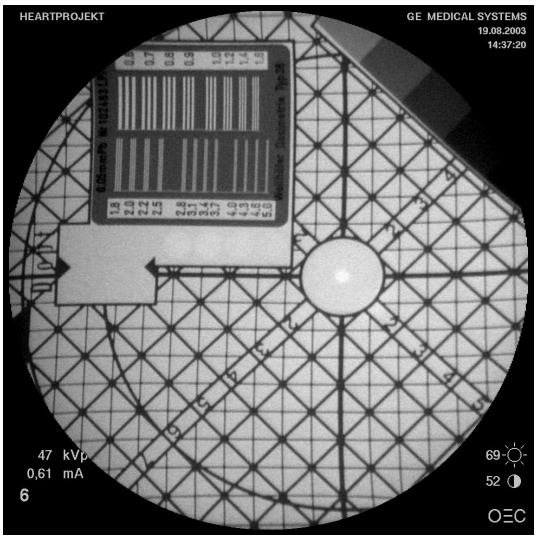
(a) image001.bmp

(b) image002.bmp

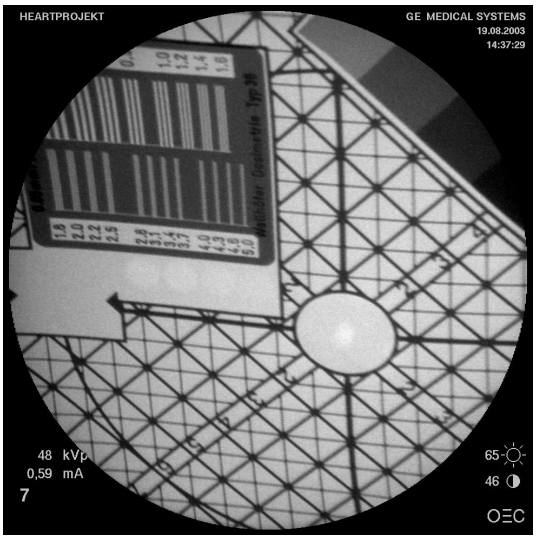
(c) image004.bmp

(d) image005.bmp

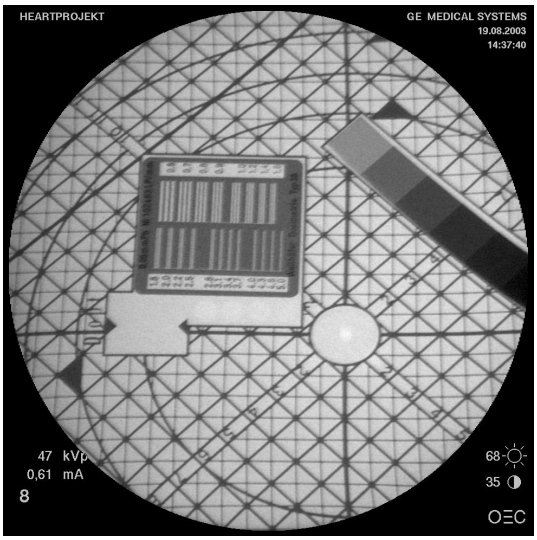
Figure B.1: Calibration plate pictured with the Image Intensifier GE MS OEC9800 - images 1 - 5



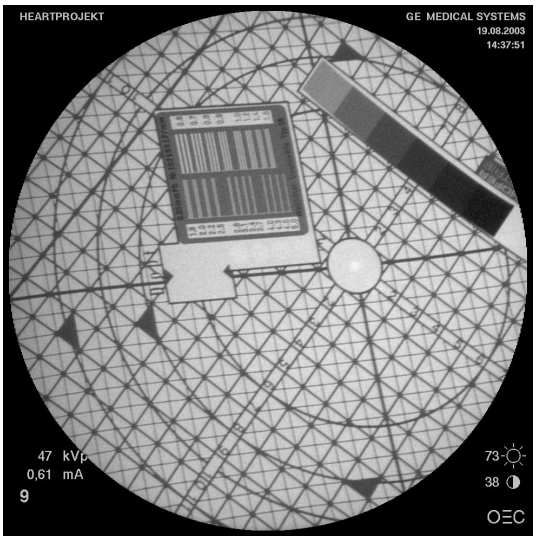
(e) image006.bmp



(f) image007.bmp

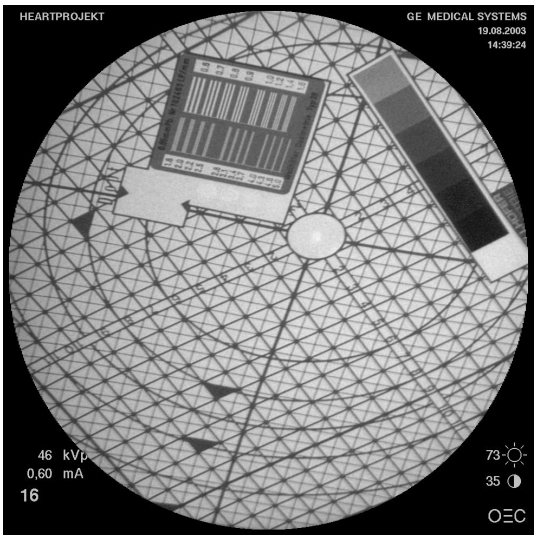


(g) image008.bmp

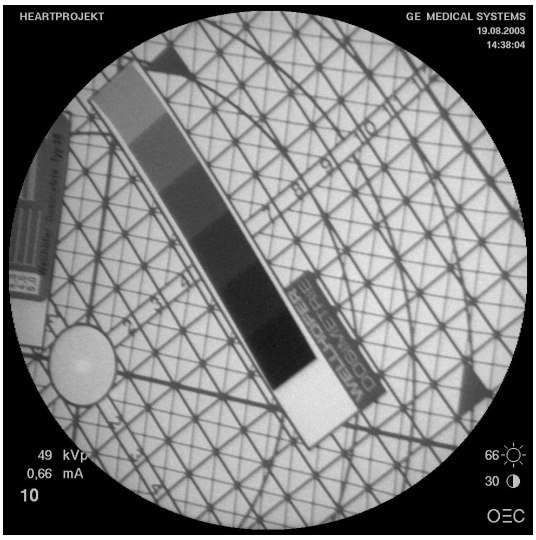


(h) image009.bmp

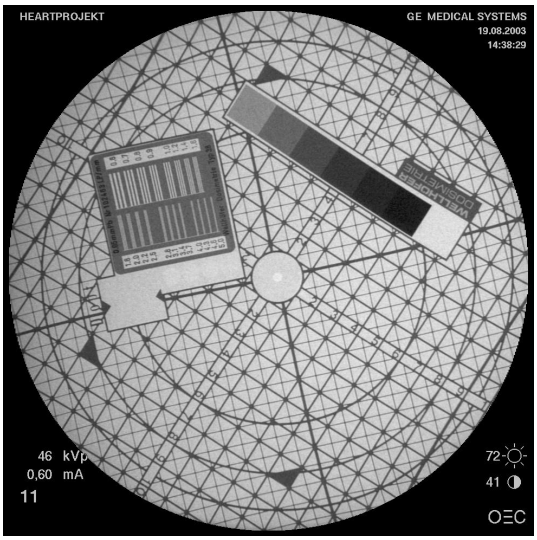
Figure B.2: Calibration plate pictured with the Image Intensifier GE MS OEC9800 - images 6 - 9



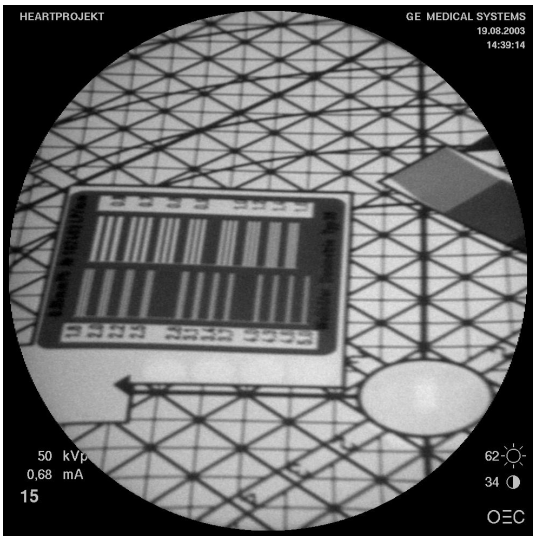
(i) image010.bmp



(j) image00A.bmp



(k) image0B.bmp



(l) image00F.bmp

Figure B.3: Calibration plate pictured with the Image Intensifier GE MS OEC9800 - images 10 - F

C Calibration Manual

Attached to the framework for calibration and registration is a main method with some targets that can be used to execute certain calibration tasks. In the following, the main method's targets and their requirements will shortly be described.

Basically, the framework expects a file where the point correspondences are stored and a number of images the correspondences belong to. The file saving the point correspondences begins with "$\$n\ m\ k$", where n and m are the number of pixels in x- and y-direction respectively and k is the number of images the point correspondences are stored for. Then, the respective correspondences for each image start with "@<*image_name*>" followed by alternately listed 3D points and their corresponding 2D points. At the end of the file a tag "*@end*" must be entered. For instance, a point correspondence file with two $980 \cdot 980$ pixel images each having 3 point correspondences would look like

```
# Example file for point correspondences
$980 980 2

@image001.bmp

1 -1 0
527 466
1 -2 0
572 466
2 -1 0
527 421

@image002.bmp
-8 6 0
62 337
-8 4 0
86 424
-8 2 0
107 512

@end
# This is the end of the file.
```

Naturally, the 3D points are in world coordinates whereas 2D points are in image coordinates, i.e. pixels. Another file is important when calibrating and especially when testing the quality. This file is provided for the intrinsic parameters – determined with Tsai or OpenCv – to be stored. It is found by the main method via the shell variable `CCPARAMFILE`. The file

consists of four numbers indicating focal length (in x- and y-direction) and principal point. For instance, such a file could look like

```
# Intrinsic Parameters
# 4 values: focallengthx, focallengthy, px, py
4288.5
4288.5
516.361
489.992.
```

With the images and the two files all targets of the main method can be executed:

- *Target **tsaicalib***:
  - *arguments*: The file of point correspondences and one image.
  - *description*: The target searches the file for the right point correspondences and stores them in two arrays. It instantiates the class `CcTsaiWrapper` and invokes the method `calibrateCamera()`.
  - *return*: The camera parameters (intrinsic and extrinsic parameters) of the calibration with Tsai / Levenberg-Marquart. The parameters are also stored in a file that is found via the shell variable `CCPARAMFILE`.
  - *call*: `./camcalib tsaicalib <pointsFile> <imageFile>`
- *Target **cvcalib***:
  - *arguments*: The file of point correspondences.
  - *description*: The target gets all point correspondences and stores them in two arrays ordered by images. It instantiates the class `CcOpenCvWrapper` and invokes the method `calibrateCamera()`.
  - *return*: The camera parameters (intrinsic and extrinsic parameters) of the calibration with OpenCv (DLT). The parameters are also stored in a file that is found via the shell variable `CCPARAMFILE`.
  - *call*: `./camcalib cvcalib <pointsFile>`
- *Target **calibtest***:
  - *arguments*: The file of point correspondences.
  - *description*: The target reads all images and displays them augmented with red circles at the pixel locations described in the point correspondences file. Additionally, the first two world coordinates (since it is a planar calibration, z is always zero) are attached to the circles in brackets. With this target the point correspondences can be checked.
  - *return*: A window is returned and displayed showing all images in turn where the red circles can be checked. A keystroke instructs the main method to display the next image. After all images have been investigated, the window is closed.
  - *call*: `./camcalib calibtest <pointsFile>`

- *Target **fp***:
  - *arguments*: The file of point correspondences and an image.
  - *description*: The target performs a forward projection on the image with intrinsic parametes given in the parameter file stated in `CCPARAMFILE`. The extrinsic parameters are deteremined for the image in the argument. Here, the projection matrix with already existent intrinsic and new extrinsic parameters is calculated and applied to all 3D points of the image.
  - *return*: The result is shown as yellow circles indicating the projected points in a window displaying the image with the measured 2D points presented as red circles.
  - *call*: `./camcalib fp <pointsFile> <imageFile>`
- *Target **optintpar***:
  - *arguments*: The file of point correspondences and an image.
  - *description*: The target takes the intrinsic parameters saved in the file specified via the shell variable `CCPARAMFILE` and calculates the extrinsic parameters of the given image. Then, it applies a forward-projection of the 3D points from the point correspondence file of the given image and stores it in a first array. Afterwards, the intrinsic parameters are optimized by iterating via a Weighted Best Neighbor approach over a parameter vector (f_x, f_y, p_x, p_y), which is minimizing the Euclidean distance between measured and forward-projected 2D points. Then, a second array is filled with the optimized forward-projections.
  - *return*: The measured 2D points as well as the (un)optimized forward-projections are plotted on the given image with green, yellow and red circles respectively.
  - *call*: `./camcalib optintpar <pointsFile> <imageFile>`
- *Target **quality***:
  - *arguments*: The file of point correspondences and optionally a deviation in pixels.
  - *description*: The target takes the intrinsic parameters as stored in the parameter file (found, again, by the shell variable `CCPARAMFILE`) and determines for each image the extrinsic parameters and, thus, the projection matrix. Then, a back-projection is performed as described in sections 4.1.4 and 4.2.5.2. The distance (in world coordinate units) of measured and back-projected 3D points is determined and a mean value over all points in one image is calculated. Finally, a mean of means is determined. Moreover, a forward-projection is performed also using intrinsic parameters and the respective extrinsic parameters for each image. The distance (in pixels) is determined and if a deviation is given, it is checked whether the 2D point lies near enough to the measured 2D point. If no deviation is given, deviations from 0 to 10 pixels are computed.
  - *return*: The mean value of the distances between all back-projected 3D points and measured ones for all images and the mean of means. Also, the quality (in percent) of the forward-projections for a given deviation or for deviations from 0 to 10 pixels.
  - *call*: `./camcalib quality <pointsFile> [<deviation>]`

D Glossary

AAA *see* ABDOMINAL AORTIC ANEURYSM

Abdominal Aortic Aneurysm leak in the inner wall of the aorta which results in a blood-streamed lump located in the abdomen

AR *see* AUGMENTED REALITY

Augmented Reality. A technique that uses virtual objects to enhance the user's perception of the real world.

CANP Computer Aided Navigation and Planning

CCD Charged Coupled Device

CT *see* COMPUTER TOMOGRAPHY

Computer Tomography Imaging technique used in medicine based on creating cross-sectional images from projections of patients at multiple angles.

CTA *see* COMPUTER TOMOGRAPHY ANGIOGRAPHY

Computer Tomography Angiography Computer Tomography enhanced by angiography.

DLT Direct Linear Transformation

DICOM Digital Imaging and Communications in Medicine

DRR *see* DIGITALLY RECONSTRUCTED RADIOGRAPH

Digitally Reconstructed Radiograph An artificial X-ray image recomputed from CT data via volume rendering

GPU Graphical Processing Unit

GUI Graphical User Interface

MR *see* MAGNETIC RESONANCE

Magnetic Resonance Imaging method using a strong magnetic field (B0 field) and gradient fields to localize bursts of radiofrequency signals coming from a system of spins consisting of reorienting hydrogen H nuclei (protons) after they have been disturbed by radiofrequency RF pulses

PET *see* POSITRON EMISSION TOMOGRAPHY

Positron Emission Tomography A tomographic nuclear imaging procedure, which uses positrons as radiolabels and positron-electron annihilation reaction-induced gamma rays to locate the radiolabels

TAA *see* THORACIC AORTIC ANEURYSM

TAAA *see* THORACOABDOMINAL AORTIC ANEURYSM

Tcl *see* TOOL COMMAND LANGUAGE

Thoracic Aortic Aneurysm leak in the inner wall of the aorta which results in a blood-streamed lump located in the thorax

Thoracoabdominal Aortic Aneurysm leak in the inner wall of the aorta which results in a blood-streamed lump located in both, abdomen and thorax

TEE *see* TRANSOESOPHAGEAL ECHOCARDIOGRAPHY

Tool Command Language A scripting library implemented in C

Transoesophageal Echocardiography An ultrasonic imaging technique where the space near the gullet can be visualized.

Type-B-Dissection two leaks in the inner aortic wall creating two distinguishable and separated lumen

Bibliography

[1] Y. I. ABDEL-AZIZ and H. M. KARARA, *Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry*, Symposium on Close Range Photogrammetry, 1971, pp. 1–18.

[2] L. ANDERTON, *General Electrics OEC Medical Systems*, Personal Communications. September 2003.

[3] K. S. ARUN, T. S. HUANG, and S. D. BLOSTEIN, *Least-squares fitting of two 3-D point sets*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5), 1987, pp. 698–700.

[4] R. BELL, P. R. TAYLOR, M. AUKETT, T. SABHARWAL, and J. REIDY, *Results of Urgent and Emergency Thoracic Procedures treated by Endoluminal Repair*, European Journal of Endovascular Surgery, 25(6), June 2003, pp. 527–531.

[5] P. BERGERON, T. DE CHAUMARAY, J. GAY, and V. DOUILLEZ, *Endovascular treatment of thoracic aortic aneurysms*, Journal of Cardiovascular Surgery, 44(3), June 2003, pp. 349–361.

[6] J. D. BLANKENSTEIJN, *Imaging Techniques for Endovascular Repair of Abdominal Aortic Aneurysms*, Medica Mundi, 44(2), November 2002.

[7] C. BRACK, M. ROTH, A. CZOPF, J. MOCTEZUMA, F. GTTE, and A. SCHWEIKARD, *Towards Accurate X-Ray-Camera Calibration in Computer-Assisted Robotic Surgery*, Proceedings Computer-Assisted Radiology and Surgery (CARS), 1996, pp. 721 – 728.

[8] A. BROWN, *A survey of Image Registration Techniques*, ACM Computing Surveys, 24, 1992, pp. 326–376.

[9] B. BRUEGGE and A. DUTOIT, *Object-Oriented Software Engineering - Conquering Complex and Changing Systems*, Prentice Hall, 2000.

[10] F. CRIADO, N. CLARK, and M. BARNATAN, *Stent graft repair in the aortic arch and descending thoracic aorta: a 4-year experience*, Journal of Vascular Surgery, 36(6), December 2002, pp. 1121–1128.

[11] B. DREBIN, L. CARPENTER, and P. HANRAHAN, *Volume Rendering*, SIGGRAPH, 1988, pp. 65–74.

[12] K. ENGEL, *Strategien und Algorithmen zur Interaktiven Volumenvisualisierung in Digitalen Dokumenten*, PhD thesis, Universität Stuttgart, 2002.

[13] K. ENGEL, M. KRAUS, and T. ERTL, *High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading*, in Siggraph/Eurographics Workshop on Graphics Hardware, 2001.

[14] O. FAUGERAS, *Three-Dimensional Computer Vision: A geometric Viewpoint*, MIT Press, Cambridge, MA, 1995.

[15] M. FEUERSTEIN, *Design of a Planning Tool for Augmented Reality Supported Placement and Tracking for a Teleoperator in Heart Surgery*, Master's thesis, Technische Universität München, 2003.

[16] J. FOLEY, A. VAN DAM, S. FEINER, and J. HUGHES, *Computer Graphics - Principles and Practice*, Addison-Wesley, 1990.

[17] M. FRANKEN, *Mechanical Dynamical Simulation and Positioning of an Aortic Stent*, Master's thesis, Eindhoven Technical University, 2002.

[18] E. GAMMA, R. HELM, R. JOHNSON, and J. VLISSIDES, *Design Patterns: elements of reusable object-oriented software*, Addison Wesley, 1994.

[19] R. GÖCKE, *Schnelle Verfahren des Volume Rendering für die 2D/3D-Registrierung*, Master's thesis, Universität Rostock, 1997.

[20] J. GOLZARIAN, *Imaging after endovascular repair of abdominal aortic aneurysms*, Abdominal Imaging, 28(2), 2002.

[21] M. GRABENWOGER, T. FLECK, M. CZERNY, D. H. M. EHRLICH, M. SCHODER, J. LAMMER, and E. WOLNER, *Endovascular stent graft placement in patients with acute thoracic aortic syndromes*, European Journal of Cardiothorac Surgery, 23(5), May 2003, pp. 788–793.

[22] A. GUÉZIEC, P. KAZANZIDES, B. WILLIAMSON, and R. TAYLOR, *Anatomy-Based Registration of CT-Scan and Intraoperative X-Ray Images for Guiding a Surgical Robot*, IEEE Transactions on Medical Imaging, 17, 1998, pp. 715–728.

[23] S. HALL, *Endovascular repair of abdominal aortic aneurysms*, AORN Journal, 77(3), March 2003, pp. 631–642.

[24] C. HAMPTON, T. PERSONS, C. WYATT, and Y. ZHANG, *Survey of Image Segmentation*. `http://www.citeseer.nj.nec.com/hampton98survey.html`, 1998. 12. Nov 2003.

[25] R. HARTLEY and A. ZISSERMAN, *Multiple View Geometry in computer vision*, Cambridge University Press, 2000.

[26] P. HECKBERT, *Survey of Texture Mapping*, in IEEE Computer Graphics and Applications, November 1986, pp. 56–67.

[27] H. IMAMURA, N. IDA, N. SUGIMOTO, S. EIHO, S. URAYAMA, K. UENO, and K. INOUE, *Registration of Preoperative CTA and Intraoperative Fluoroscopic Images for Assisting Aortic Stent Grafting*, MICCAI 2002, LNCS2489, 2002, pp. 477–484.

[28] J. KAJIYA and B. VON HERZEN, *Ray Tracing Volume Densities*, Computer Graphics, 18, 1984, pp. 215–237.

[29] M. KASS, A. WITKIN, and D. TERZOPOULOS, *Snakes: Active Contour Models*, International Journal of Computer Vision, 1(4), 1988, pp. 321–331.

[30] P. KEITLER, *Mathematical Methods of Image Processing for Automated Navigation in Endoscopic Treatment of Aortic Aneurysms - Computer Aided Implantation of a Stent Graft*, Master's thesis, Technische Universität München, 2003.

[31] I. KOMPATSIARIS, D. TZOVARAS, and M. G. STRINTZIS, *Deformable Boundary Detection of Stents in Angiographic Images*, IEEE Transactions on Medical Imaging, 19(6), 2000, pp. 652 –662.

[32] KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK BERLIN (ZIB) and INDEED - VISUAL CONCEPTS GMBH, *amira 3.0 - User's Guide and Reference Manual*, TGS Template Graphics Software, Inc., Berlin, Germany, and USA, 2002.

[33] S. LAKARE, *3D Segmentation Techniques for Medical Volumes*. December 2000.

[34] A. LAMOTHE, *Black Art of 3D Game Programming*, Waite Group Press, 1995.

[35] J. LANDRÉ, *Programming with Intel IPP and Intel OpenCV under GNU Linux: a beginners tutorial*. obtained at `http://sourceforge.net/projects/opencvlibrary/`, 2003. 28. Oct 2003.

[36] S. LAVALLÉE and R. SZELISKI, *Recovering the position and orientation of free-form objects*, IEEE Transactions on pattern analysis and machine intelligence, 17, 1995, pp. 378–390.

[37] S. LAVALLÉE, J. TROCCAZ, P. SAUTOT, B. MAZIER, P. CINQUIN, P. MERLOZ, and J. CHIROSSEL, *Computer-integrated surgery, Technology and clinical applications*, MIT Press, Cambridge, MA, 1996, ch. 32, pp. 425–449.

[38] M. LEVOY, *Display of Surfaces from Volume Data*, IEEE Computer Graphics and Applications, 8(5), 1988.

[39] W. LORENSEN and H. CLINE, *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, SIGGRAPH, 1987, pp. 163–169.

[40] J. MAINTZ and M. VIERGEVER, *A Survey of Medical Image Registration*, Medical Image Analysis, 2, 1998, pp. 1–36.

[41] J. MATSUMURA, D. BREWSTER, M. MAKAROUN, and D. NAFTEL, *A multicenter controlled clinical trial of open versus endovascular treatment of abdominal aortic aneurysm*, Journal of Vascular Surgery, 37(2), February 2003, pp. 262–271.

[42] M. MENARD, D. CHEW, R. CHAN, M. CONTE, M. DONALDSON, J. MANNICK, A. WHITTEMORE, and M. BELKIN, *Outcome in patients at high risk after open surgical repair of abdominal aortic aneurysm*, Journal of Vascular Surgery, 37(2), February 2003, pp. 285–292.

[43] B. MUNIER, P. PRIEUR-DREVON, and J. CHABBAL, *Digital radiology with solid state linear x-ray detectors*, in Proceedings of SPIE, vol. 1447, 1991, pp. 44–55.

[44] NATIONAL ELECTRIC MANUFACTURERS ASSOCIATION, *Digital Imaging and Communications in Medicine (DICOM)*. `http://medical.nema.org/dicom/2003.html`, 2003. 22. Oct 2003.

[45] K. OREND, T. KOTSIS, R. SCHARRER-PALMER, X. KAPFER, F. LIEWALD, J. GORICH, and L. SUNDER-PLASSMANN, *Endovascular Repair of Aortic Rupture due to Trauma and Aneurysm*, European Journal of Vascular and Endovascular Surgery, 23(1), Januray 2002, pp. 61–67.

[46] K. OREND, R. SCHARRER-PALMER, X. KAPFER, T. KOTSIS, J. GORICH, and L. SUNDER-PLASSMANN, *Endovascular treatment in diseases of the descending thoracic aorta: 6-year results of a single center*, Journal of Vascular Surgery, 37(1), January 2003, pp. 91–99.

[47] G. P. PENNEY, J. WEESE, J. A. LITTLE, P. DESMEDT, D. L. G. HILL, and D. J. HAWKES, *A Comparison of Similarity Measures for Use in 2-D-3-D Medical Image Registration*, IEEE Transactions of Medical Imaging, 17, 1998, pp. 586–595.

[48] H. PETTERSON, *Amersham Health Medcyclopaedia*. `http://www.amershamhealth.com/medcyclopaedia`. 3rd Dec 2003.

[49] H. PETTERSSON, *Seldinger technique*. `http://www.amershamhealth.com/medcyclopaedia/Volume%20I/Seldinger%20technique.asp#Fig.1`, 2002. 03. July 2003.

[50] H. PFISTER, *Architectures for Real-Time Volume Rendering*, Future Generation Computer Systems, 15, 1999, pp. 1–9.

[51] D. L. PHAM, C. XU, and J. L. PRINCE, *A Survey of Current Methods in Medical Image Segmentation*, tech. rep., John Hopkins University Baltimore, 1998.

[52] J. POWELL and R. GREENHALGH, *Small Abdominal Aortic Aneurysms*, The New England Journal of Medicine, 348(19), 2003.

[53] C. REZK-SALAMA, *Volume Rendering Techniques for General Purpose Graphics Hardware*, PhD thesis, Universität Erlangen-Nürnberg, 2001.

[54] C. REZK-SALAMA, K. ENGEL, M. BAUER, G. GREINER, and T. ERTL, *Interactive Volume on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization*, in Siggraph/Eurographics Workshop on Graphics Hardware, 2000.

[55] M. ROCHAT, *Navigation System for Aortic Stent Placement*, Master's thesis, Swiss Federal Institute of Technology Lausanne, 2002.

[56] M. ROTH, *Intraoperative fluoroskopiebasierte Patientenlageerkennung zur präzisen Unterstützung chirurgischer Eingriffe*, PhD thesis, Technische Universität München, 2000.

[57] M. ROTH, C. BRACK, R. BURGKART, A. CZOPF, H. GÖTTE, and A. SCHWEIKARD, *Multi-view contourless registration of bone structures using a single calibrated X-ray fluoroscope*, Computer Assisted Radiology and Surgery (CARS'99), 1999, pp. 756–761.

[58] S. RUDIN, D. BEDNAREK, and W. WONG, *Accurate Characterization of Image Intensifier Distortion*, Medical Physics, 18, 1991, pp. 1145–1151.

[59] D. RUECKERT, *Non-rigid registration: Techniques and Applications*, in Medical Image Registration, J. V. Hajnal, D. L. G. Hill, and D. J. Hawkes, eds., CRC Press, 2001.

[60] S. LAVALLÉE, R. H. TAYLOR, G. C. BURDEA, and R. MSGES, *Computer-integrated surgery, Technology and clinical applications*, MIT Press, Cambirdge, MA, 1996.

[61] P. SABELLA, *A Rendering Algorithm for Visualizing 3D Scalar Fields*, in International Conference on Computer Graphics and Interactive Techniques, 1988, pp. 51–58.

[62] D. SARRUT and S. MIGUET, *Similarity Measures for Image Registration*, European Workshop on Content-Based Multimedia Indexing, 1999, pp. 263–270.

[63] D. SCHRÖCK, *Erste Erfahrungen mit dem SOMATOM Sensation 16*, Master's thesis, Ausbildungszentrum West für Gesundheitsberufe Innsbruck, 2002.

[64] B. SCHUELER and X. HU, *Correction of Image Intensifier Distortion for Three-dimensional X-ray angiography*, SPIE: Medical Imaging, 2432, 1995, pp. 272–279.

[65] D. SOIMU, C. BADEA, and N. PALLIKARAKIS, *A Novel Approach for Distortion Correction for X-Ray Image Intensifiers*, Computerized Medical Imaging and Graphics, 27, 2003, pp. 79–85.

[66] M. SUBASIC, S. LONCARIC, and E. SORANTIN, *3-D deformable model segmentation of abdominal aortic aneurysm*, in Proceedings of SPIE Medical Imaging, San Diego, USA, 2001, pp. 388–394.

[67] M. SUBASIC, S. LONCARIC, and E. SORANTIN, *3-D image analysis of abdominal aortic aneurysm*, in Proceedings of SPIE Medical Imaging, San Diego, USA, 2002.

[68] P. TAGHIZADEH, *Post-traumatic Aneurysm of the Axillary Artery*. `http://cats.med.uvm.edu/cats_teachingmod/radiology/radiology_html/teaching/radio_adult/axillary_aneurysm/axillary.htm`. 3rd Dec 2003.

[69] G. TEITELBAUM, W. BRADLEY, and B. KLEIN, *MR imaging artifacts, ferrromagnetism, and magnetic torque of intravascular fitlers, stents, and coils*, Radiology, 166, 1998, pp. 657–664.

[70] J. TRAUB, *Design of an Intra-Operative Augmented Reality Navigation Tool for Robotically Assisted Minimally Invasive Cardiovascular Surgery*, Master's thesis, Technische Universität München, 2003.

[71] R. TSAI, *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras*, IEEE Journal of Robotics and Automation, RA-3(4), 1987, pp. 323–344.

[72] M. TUCERYAN, Y. GENC, and N. NAVAB, *Single point active alignment method (SPAAM) for optical see-through HMD calibration for augmented reality*, Proceedings of the IEEE and ACM International Symposium on Augmented Reality, 2000, pp. 149–158.

[73] S. UMEYAMA, *Least-Squares Estimation of Transformation Parameters Between Two Point Patterns*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(4), 1991, pp. 376–380.

[74] P. VAN DEN ELSEN, E.-J. POL, and M. VIERGEVER, *Medical image matching-a review with classification*, IEEE Engineering in Medicine and Biology Magazine, 12, 1993, pp. 26 –39.

[75] V. VENOT, J. LEBRUCHEC, and J. ROUCAYROL, *A New Class Of Similarity Measures For Robust Image Registration*, CVGIP, 28, 1984, pp. 176–184.

[76] A. WATT and M. WATT, *Advanced Animation and Rendering Techniques, Theory and Practice*, Addison-Wesley, 1992.

[77] J. WERNECKE and OPEN INVENTOR ARCHITECTURE GROUP, *The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor*, Addison Wesley, 1994.

[78] R. WESTERMANN and T. ERTL, *Efficiently using Graphics Hardware in Volume Rendering Applications*, in Proceedings of the 25th annual conference on Computer graphics and interactive techniques (ACM SIGGRAPH), 1998, pp. 169 – 177.

[79] J. WILHELMS, *Visualizing Sampled Volume Data*, in Scientific Visualization and Graphics Simulation, N. Magnenat-Thalmann and D. Thalmann, eds., Wiley, 1990.

[80] B. ZITOVÁ and J. FLUSSER, *Image Registration Methods: a Survey*, Image and Vision Computing, 21, 2003, pp. 977–1000.