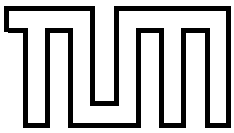
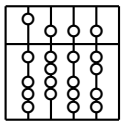


Technische Universität München
Fakultät für Informatik

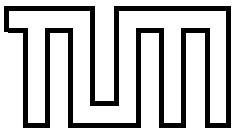


Diplomarbeit

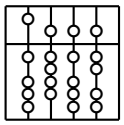
Design of a Planning Tool for Port Placement in Robotically Assisted Minimally Invasive Cardiovascular Surgery

HEART: Heart Surgery Enhanced by Augmented Reality Techniques

Marco Feuerstein



Technische Universität München
Fakultät für Informatik



Diplomarbeit

Design of a Planning Tool for Port Placement in Robotically Assisted Minimally Invasive Cardiovascular Surgery

HEART: Heart Surgery Enhanced by Augmented Reality Techniques

Marco Feuerstein

Aufgabensteller:	Prof. Gudrun Klinker, Ph.D. PD Dr. med. Robert Bauernschmitt
Betreuer:	Dipl.-Inf. Martin Bauer Dipl.-Inf. Hesam Najafi Eva Schirmbeck
Abgabedatum:	23. Dezember 2003

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

München, den 23. Dezember 2003 Marco Feuerstein

Zusammenfassung

In der minimal-invasiven Herzchirurgie werden für eine Operation nur kleine *Inzisionen* (Einstiche) im Rippenbereich gesetzt. Deshalb kann man auf eine Öffnung des Brustkorbes oder einen Einsatz der belastenden Herz-Lungen-Maschine verzichten, was im Vergleich zu herkömmlicher invasiver Chirurgie zu einer Verminderung der Komplikationen führt. Erst sogenannte *Master-Slave* Teleoperator-Systeme ermöglichten eine sinnvolle Realisierung dieser Art von Operationen. Bei einer Teleoperation befindet sich der Chirurg an einer Steuerungseinheit, während der *Slave*-Roboter dessen Hand- und Fingerbewegungen imitiert.

Für die Durchführung eines erfolgreichen Eingriffes mit einem Teleoperator ist es unabdingbar, dass die optimalen Inzisionen bzw. *Ports* für die Teleoperator-Arme gefunden werden. Bisher wurde die Planung der Ports von erfahrenen Chirurgen durchgeführt, die sich allein auf die Interpretation zuvor erstellter Bildsequenzen stützten, welche z.B. von einem Computertomographen stammen.

Diese Arbeit stellt nun ein Planungswerkzeug zur Port-Platzierung vor, welches sich dadurch auszeichnet, dass es dem Arzt sowohl die Simulation des chirurgischen Eingriffes in einer virtuellen Umgebung als auch alle dazu notwendigen Vorbereitungsschritte ermöglicht, wie z.B. die Segmentierung axialer Computertomographie-Schichtbilder und die dreidimensionale Rekonstruktion eines Patientenmodelles. Während der virtuellen Simulation eines Eingriffes können die Teleoperator-Arme am Patientenmodell platziert und interaktiv angepasst werden. Dabei wird sichergestellt, dass sie weder untereinander noch mit lebenswichtigen Organen und Knochen kollidieren. Für den Planungsprozess werden über die graphische Benutzerschnittstelle zahlreiche Visualisierungs-, Planungs- und Validierungseinstellungen bereitgestellt. Die Verifikation der Port-Platzierung erfolgt über Techniken der Kollisionserkennung und des virtuellen Endoskopes. Nach erfolgreichem Abschluss der Port-Platzierung können die Planungsdaten an ein Hilfssystem für die intraoperative Navigation exportiert werden.

Abstract

In minimally invasive cardiovascular surgery only small incisions are used to perform an operation. A split of the breastbone or the use of the straining heart-lung machine can be avoided, resulting in less surgical complications than traditional invasive surgery. A reasonable realization of these operations was facilitated by master-slave systems for teleoperations. Whereas the surgeon is controlling the teleoperator sitting at a master console, the arms of the slave robot are able to imitate the surgeon's actions on the patient.

For a successful teleoperator based intervention it is crucial to find the optimal locations, also referred to as ports, for the incision sites of the teleoperator arms. Traditionally, planning the port placement was dependant on the surgeon's experience in interpreting previously acquired two-dimensional image stacks such as computed tomography slices.

This work introduces a planning tool for port placement that is supporting the surgeon to segment cross-sectional axial slices acquired by imaging modalities, reconstruct a three-dimensional model of the patient from these slices, and simulating the cardiovascular intervention in a virtual environment. For the virtual simulation of an intervention, the teleoperator arms can be interactively placed and moved within the patient's model in such a way that they neither intersect each other nor vitals and bones. Therefore, the planning tool's graphical user interface provides various visualization, planning, and validation options. Collision detection techniques and a virtual endoscope view support the verification of the port placement process. All planned data can be exported to an intra-operative navigation tool.

Preface

About this work This work is the *Diplomarbeit* (comparable to *Master's thesis*) for the studies of *Dipl.-Informatik* (*Computer Science*) and was done at the *Chair for Applied Software Engineering* and the *Deutsches Herzzentrum München*, 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 *STENT* and *PORT*. Two theses concern the planning, guidance, and navigation of the computer supported implantation of stent grafts [30, 37]. This thesis and Jörg Traub's work [79] are part of the latter one, which provides a planning, placement, and tracking environment for the arms of any telemanipulator in minimally invasive cardiovascular surgery.

Acknowledgments First of all, I would like to thank my thesis advisors *PD Dr. med. Robert Bauernschmitt* and *Prof. Gudrun Klinker* for making this thesis possible. I also owe much thanks to my supervisors *Hesam Najafi* and, especially, *Eva Schirmbeck* and *Martin Bauer* for giving me helpful advice, guidance, and support during the last months. Having a stimulating collaboration, many mathematical and medical problems could be tackled, discussed, and solved.

I also owe a great deal to the people proofreading this thesis, namely *Kirk Turner*, *Doro Dening*, and, especially, *Joanne Yap*. They provided me with a lot of helpful and corrective advice.

The *HEART* team supported and encouraged me constantly throughout the whole thesis. I would like to thank *Peter Keitler* and *Martin Groher* a lot for helping me negotiating my way through the amira 'jungle'. Especially, I want to thank *Jörg Traub* very much for jointly writing sections of the thesis and many fruitful discussions on the port planning and guidance system.

Moreover, I owe a lot of thanks to my girlfriend *Ching-Hsin Weng*, who was never rejecting her full support for my questions and problems such as the 'arm problem'. Thank you very much for your patience. Lastly, I owe many thanks to my parents - *Franz* and *Marianne Feuerstein* - for their patience and aid, not only for the thesis, but also for enabling me to study at all.

Contents

1 Introduction

> But, for my own part, it was Greek to me.
>
> William Shakespeare

The rapid development of modern operation tools did not neglect the field of cardiac surgery. The physicians at the *Klinik für Herz- und Gefäßchirurgie* (*Clinic for Cardiovascular Surgery*) at the *Deutsches Herzzentrum München* (*German Heart Center Munich*), for example, have been using a telemanipulator system from *Intuitive Surgical Inc*[1] called *da Vinci* for their minimally invasive cardiac operations since the year 2000.

Minimally invasive surgery (*MIS*) is an operative procedure, i.e. a therapeutic or diagnostic procedure that wounds the surface of the body. It is performed with smaller transections of the patient's skin and less exposure to the patients themselves than a traditional invasive procedure. As the patient's thorax is left intact, the procedure is categorized as closed chest surgery or keyhole surgery. An endoscopic operation is a form of minimally invasive surgery. Compared to a conventional surgical intervention on the heart, the main advantages of a minimally invasive operation are less trauma for the patient, which is accompanied by less post-operative complications and faster recovery, lower infection risk, shorter mean intensive therapy stay, shorter mean hospital stay, and the avoidance of using the heart-lung machine, which is a strain on every patient.

1.1 Motivation

Combining the traditional medical methods with the imminent technical capabilities will provide physicians with an additional source of information, but can never replace them. The general goals of *computer assisted surgery* (*CAS*) are:

- for the patient: the reduction of pain, trauma, and recovery time
- for the physician: the provision of a more moderate access to the patient and an extension to human senses
- for the treatment: a decrease in costs, an increase in the success rates of the treatment, and time efficiency.

The physicians' vision of the transparent patient, that provides them with an inside view of the patient's anatomical and functional structure, is about to materialize. Therefore, minimally invasive and endoscopic surgery has to become more reliable. The desired medical process will be based on stabilized, optimized, and saved technical methods.

[1] `www.intuitivesurgical.com`

1.2 Telepresence Systems in Robotically Assisted Minimally Invasive Surgery

A *telepresence* system is defined as a human-machine interface, in which the human operator of a technical system receives sufficient information about the telemanipulator and the task environment. All information is displayed in a natural way, so the human operator feels physically present at a distant operation site [1]. Therefore, the operator's sensory and manipulatory abilities are extended to perform a certain task remotely.
A *telemanipulator*, also referred to as *teleoperator*, needs to have sensors, arms and/or hands, and multimodal communication channels to and from the operator. Sensing important information on its environment the teleoperator provides the sensed information to the operator in a natural and realistic way. Hence, the operator feels present at a remote location, is telepresent. An ideal way of telepresence is that the operator is entirely immersed in the remote environment, while not loosing any sensory skills.

These are the exact features the *da Vinci* telemanipulator system used by the *Deutsches Herzzentrum München* strives to replicate. It consists of three components: a master console with an integrated stereo viewer, a manipulator with three robot arms, and a vision cart. The physician is seated on the master console and the telemanipulator, representing a slave robot, is located remotely. A typical operation setup can be seen in figure 1.1.

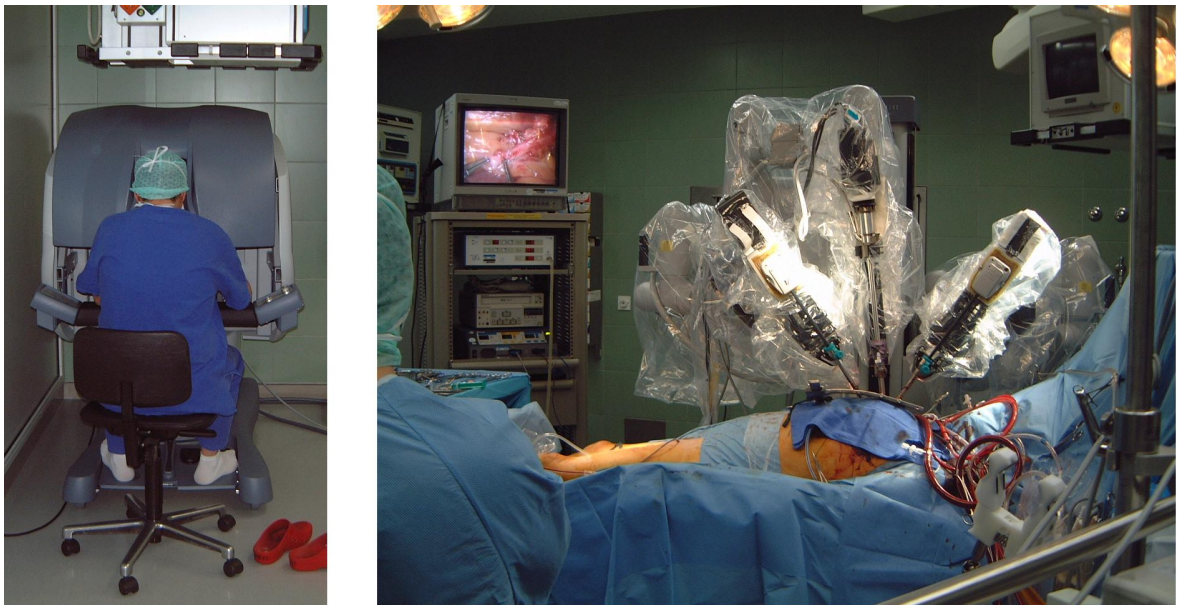


Figure 1.1: A typical setup of robotically assisted minimally invasive surgery; on the left there is a master console, on the right the telemanipulator, and in the middle an external display showing one channel of the stereo picture of the endoscope

The telemanipulator is capable of imitating the physician's actions on the patient. The actions are triggered by moving robotic handles at the master console.
Da Vinci contains one twelve millimeter wide endoscopic and two eight millimeter wide instrumental arms. During a heart operation, these are inserted laterally at the thorax through

their corresponding *ports*. Ports are the entry points in minimally invasive surgery [4, 20].

Physicians can choose from a selection of instruments like needles, scissors, forceps, or hooks, depending on the kind of operation they intend to perform. All instruments can be exchanged during an operation. The instrument tips called *EndoWrist* provide articulated motion with a full seven degrees of freedom inside the abdominal cavity. They can imitate the up and down as well as the side-to-side flexibility of the human wrist. Moreover, they are aligned with the instrument controllers electronically and, hence, provide optimal hand-eye orientation as well as natural operative capability.
The endoscope arm consists of two three chip cameras. Each of them has a separate optical channel and thus a stereo view of the operative images can be provided. These images are transmitted to a high resolution binocular display at the master console [84].

Introducing robots in minimally invasive surgery such as the *da Vinci* telemanipulator or the *ZEUS* surgical system by *Computer Motion*[2] leads to *robotically assisted minimally invasive surgery* (*RMIS*), which enables totally endoscopic surgery. Those robots solve problems of traditional surgery such as missing *degrees of freedom* (*DOF*), handling of imprecise, heavy, delicate, or large instruments, as well as poor visibility [11].

Despite the advantages of robotically assisted minimally invasive surgery, cardiac surgeons need to face new arising problems through the use of telemanipulators. Anatomical structures such as the coronary vessels on the surface of the heart, the configuration of the heart itself, and its position in the chest are different from patient to patient. Hence, the incisions for the insertion of the instruments of the telemanipulator need to be customized to every single patient. Finding an optimal placement for these incisions or ports, respectively, and their operative appliance is essential for a successful and efficient telemanipulator-based operation.

A detailed overview on robots in surgery, including the history, state of the art, and future challenges is provided by *K. Cleary* and *C. Nguyen* [18]. Another survey was published by the robotics and automation society in a special issue of the *IEEE Transactions on Robotics and Automation* [83].

1.3 Computer Technologies in Medicine

The use of and need for computers in medical contexts have increased dramatically over the last decade. As a result, nearly all aspects of medical care have been improved by the introduction of computer-based tools. The application of those tools is in the training and assistance of physicians in diagnoses, planning, execution, and follow-up of surgical procedures. Therefore, several techniques such as image acquisition, processing, segmentation, rendering, reconstruction, and registration are used.

1.3.1 Computer Assisted Medical Imaging

Two-dimensional (*2D*) analysis of *X-ray* images is a traditional way of non-invasive surgical planning, which is capable of providing an inside view of the patient's anatomical structures.

[2] *Intuitive Surgical* and *Computer Motion* (`www.computermotion.com`) merged in June 2003, so there might be a merge of their offered teleoperator systems in the future as well.

The introduction of more sophisticated image acquisition methods in recent years led to an increase in the dimensionality of these images to three (*3D*) and even four dimensions (*4D* - three dimensions and time). *Computed tomography* (*CT*), *magnetic resonance imaging* (*MRI*), *ultrasound* (*US*), *positron emission tomography* (*PET*), and other imaging modalities are techniques to acquire a three-dimensional image based on a scan of multiple two-dimensional slices in one axial direction. To generate a more intuitive representation of the acquired images, a three-dimensional model is reconstructed from segmented two-dimensional slices, as described in the following sections 1.3.1.2 and 1.3.1.3. Figure 1.2 shows the entire medical image acquisition and processing life cycle.

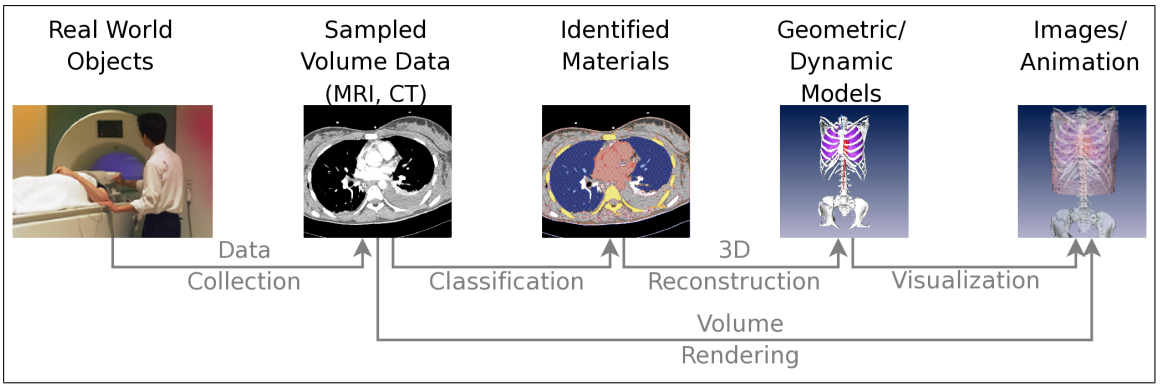


Figure 1.2: The image acquisition and processing life cycle consists of the process of collecting image data from physical objects, their classification using segmentation techniques, and its three-dimensional reconstruction. Visualization techniques are used to render the three-dimensional model onto the image plane.

1.3.1.1 Image Acquisition

Pre-operatively, image data of the patient is collected in order to diagnose, plan, simulate, and assist an intervention. A wide range of imaging modalities, such as computed tomography, magnetic resonance imaging or ultrasound are available to acquire image data. Using different sensor techniques results in different image data. The introduction of contrast agents provides not only anatomical, but also functional image data, for instance blood vessels, bile ducts, or the bowel.
Endoscopes and surgical microscopes gather image data of visible surfaces deep inside the patient's body.

All imaging technologies provide an additional source of data on the anatomical and functional property. Current computed tomography image data has a spatial resolution with accuracy down to a millimeter. The resolution is growing with each successive generation of image acquisition systems. Also, the images are acquired faster and in an improved quality for every generation [31].

The most common standard for acquiring and distributing medical images is called *DICOM* (*Digital Imaging and Communications in Medicine*). It was introduced and is maintained by the *National Electrical Manufacturers Association*[3]. *DICOM* features support for a

[3] `medical.nema.org`

networked environment based on the *Transmission Control Protocol* and the *Internet Protocol* (*TCP/IP*), a file format, semantics of how devices interact with the standard, levels of conformance, and a sub-document structure as proposed by *ISO* (*International Organization for Standardization*). In addition to images and graphics also other objects like printings or reports and their relationship is described and specified. The entire documentation is published in the *DICOM Standard* set [2].

1.3.1.2 Segmentation

Volume *segmentation* is an important task to efficiently analyze and diagnose medical volumes of interest such as anatomical structures and organs. In general, three-dimensional segmentation is used to cluster or isolate similar regions and label them.
Segmentation can be done manually, automatically, and interactively. Whereas manual segmentation is very time-consuming and automatic segmentation often restraining, interactive segmentation is a promising trade-off.
The algorithms performing a segmentation can be categorized into structural, stochastic, and hybrid techniques. Structural techniques utilize information about the structure of the region, which is supposed to be segmented. In contrast, stochastic methods such as thresholding, clustering, and classification are applied on discrete voxels, i.e. on points in three-dimensional space. For every single voxel the decision is made whether it belongs to a certain region or not. Hybrid techniques take into account both the structure of a region and the result of stochastic analysis.
Segmentation is able to provide pre-diagnostic assumptions. A very common problem of all available segmentation algorithms is that they are not applicable to every kind of medical data set and, thus, they are mostly tailored to specific problems [39].

1.3.1.3 Reconstruction and Volume Rendering

Imaging modalities like computed tomography acquire a set of slices, which lacks in high resolution in one dimension, typically the z-axis or sagittal direction. The missing data of this direction needs to be *reconstructed*, i.e. interpolated and tessellated, in order to get three-dimensional image data [3]. The process of reconstructing is also referred to as deconvolving, i.e. combining the slices.
The acquired three-dimensional image data still needs to be represented on a computer screen. Therefore, three-dimensional models such as polygon meshes can be reconstructed and visualized. Alternatively, the technology of *volume rendering* can be used to transform the three-dimensional volume samples into a two-dimensional image, which can be viewed on a screen. Volume rendering comprises various methods for displaying volumetric data directly on the screen, rather than generating and displaying polygons. These methods include the forming of an *RGBA*[4] volume from the previously acquired image data and reconstruction of a continuous function from this discrete image data. Eventually, the rendered volume is projected onto a two-dimensional viewing plane, which displays the output image from a desired point of view. Widely utilized volume rendering methods are ray-casting

[4] An RGBA volume is a four-vector data set composed of three values for the colors red, green, and blue, as well as a fourth alpha component called opacity or transparency.

and splatting [41, 71]. An overview on volume rendering techniques can be found in *Peter Keitler*'s and *Martin Groher*'s master's theses [30, 37].

1.3.2 Virtual Reality

Aukstakalnis and *Blatner* [6] define *virtual reality* (*VR*) as "a way for humans to visualize, manipulate and interact with computers and extremely complex data". Virtual reality allows physicians to perceive and interact in a purely synthetic, computer-generated environment. It is proposed for use in many medical fields such as:

- *Training and teaching:* Medical interventions can be simulated for the training of physicians. A visualization of the patient's anatomy can teach the physician its function and detailed interior. With the increased sophistication of the surgery room, there will be a need for virtual training environments.
- *Surgery planning:* The planning of risky surgeries in complex anatomy like neurosurgery and cardiac surgery can be executed pre-operatively and, hence, injury to organs can be avoided.
- *Telemedicine:* A patient who is physically separated from the physician can be treated or operated by a telemanipulator (see section 1.2). Therefore, visual, acoustic, and haptic feedback may be given by a virtual reality system to assist the physician pre- and intra-operatively. This merging of virtual reality with telepresence gives mixed reality or seamless simulation systems, where computer-generated inputs are merged with telepresence inputs and the physician's view of the real world [35]. For instance, a physician's view of the heart is overlaid with images from earlier computed tomography scans or a reconstructed three-dimensional cardiac model of the patient.
- *Computer assisted surgery:* An obstacle of traditional surgery is that only the skin or surface of a patient is visible to the physician unless it is opened. Using virtual objects the patient's interior such as organs or bones can be visualized, which enables intra-operative navigation. Therefore, augmented reality techniques (see section 1.3.3) may be employed [81].
- *Rehabilitation:* A mental health therapy can be administered to heal phobias and behavioral disorders. Virtual environments support the healing process in orthopedics and neurology and enable Parkinson patients stroke therapy through motor training.

One promising virtual reality application in medicine is the virtual endoscope. A three-dimensional model of the patient is used for a fly-through simulation with a virtual camera. This provides the physician with the expected view of the patient's interior in an entirely virtual environment. This virtual endoscopy technique supports pre-operative training and simulation as well as intra-operative navigation in minimally invasive surgery [5, 9].

1.3.3 Augmented Reality

Augmented reality (*AR*) is a technology, which superimposes virtual (computer-generated) objects on the view of the real world. Its goal is to align the computer-generated objects

perfectly with their real world correspondents in three dimensions in order to provide the user of such a system with useful information. Augmented reality, as defined by *Azuma* [7],

- combines the real world and computer-generated objects in the same environment,
- is interactive and in real time, and
- is registered in three dimensions.

Milgram [53] defines augmented reality in a reality-virtuality continuum (cf. figure 1.3) as a part of *mixed reality* (*MR*), the general field between an entirely real and an entirely virtual environment.

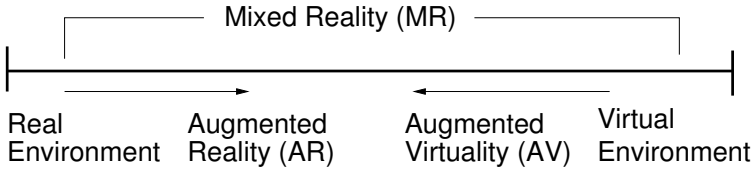


Figure 1.3: *Milgram* defines augmented reality in the reality-virtuality continuum as the part of mixed reality that focuses on the real world superimposed by computer-generated objects. However, the other part that contains real objects in a virtual environment is called augmented virtuality.

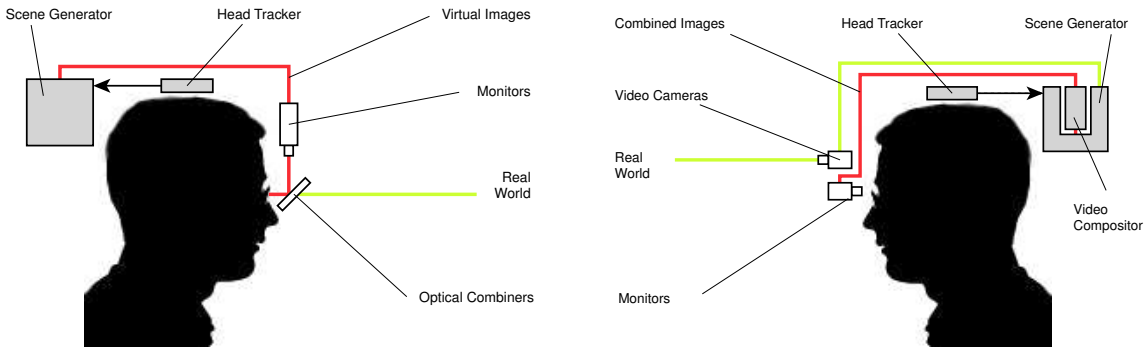
Recent publications on augmented reality show the diversity of the applications and techniques [8, 55, 75]. In the following sections the main components of a typical augmented reality system are described in further detail.

1.3.3.1 System Architecture

In general, an augmented reality application can be decomposed into several subsystems. The most central subsystems are tracking, calibration, and rendering. Promising solutions of augmented reality software architectures focus on multi-modal, distributed, and wearable systems designs. A desired architecture is using the black box pattern. For example, the rendering engine consumes pose data, i.e. position and orientation, of the camera provided by the tracking subsystem. Current studies of the augmented reality research group at the *Technische Universität München* concentrate on the design of a distributed wearable augmented reality framework (*DWARF*) [10].

1.3.3.2 Display Devices

To project the virtual objects in the user's *field of view* (*FOV*) there exist different techniques such as head-mounted, monitor-based, and projective. *Head-mounted displays* (*HMD*s) are traditionally the most commonly used techniques. They offer the user a direct stereo projection in their field of view. The two subcategories of head-mounted displays are optical see-through (see figure 1.4(a)) and video see-through (see figure 1.4(b)). A detailed discussion of optical versus video see-through displays is given by *Azuma* [7].



(a) Functional model of an optical see-through head-mounted display.

(b) Functional model of a video see-through head-mounted display.

Figure 1.4: The real world view through an optical see-through head-mounted display is augmented by projecting the virtual objects onto semi-transparent combiner plates, whereas the video see-through head-mounted display captures the pictures of the real world as a video stream and superimposes the computer-generated objects on the video frames.

However, the joint visualization of reality and virtuality is difficult to achieve in an optical head-mounted display, where the only optical element in the user's view is a simple semi-transparent combiner plate. A combiner plate, also referred to as beam splitter, is an optical component, which combines two optical paths, namely virtuality and reality. It is most likely that the object distance of the virtual scenery reflected on the beam splitter is different from the object distance of the real world scenery. Hence, either the virtual world objects or the real world objects appear unfocused [52].
Video see-through head-mounted displays merge a real-time video and the computer-generated objects on every single frame. The video stream is supposed to capture the real world exactly the way the user will see it. The merged stream is displayed directly in front of the user's eye. Therefore, the view of the real world is totally opaque to the user.
Monitor based augmented reality systems work similar to video see-through devices, but the display and the camera capturing the video stream are not mounted on the user's head. An external monitor is used to display the overlay of the two modalities, reality and virtuality.
A distinctly different approach is the projective augmented reality technique. Using an off-the-shelf video beamer the virtual scene is directly projected onto the real objects. This technique is limited to projections onto the surface of the real world objects and generally used for texture representation [47, 59].

1.3.3.3 Tracking of Real Objects

In an augmented reality system the pose of the user's eyes and all non-rigid objects need to be tracked in the real environment. Only then will the virtual scene be superimposed on the real environment with perfect alignment over time. Tracking is a real-time task that provides

the scene graph with a sufficient update rate of the pose of the real world objects and the viewpoint. There are different kinds of tracking subsystems, each with its own advantages and disadvantages in specific environments and tasks.

In optical tracking systems a camera detects markers to estimate the pose of itself or of an object. Optical tracking systems are very susceptible to light conditions. Therefore, alternative infrared (*IR*) optical tracking systems are introduced using both passive (*IR* reflective) and active (*IR LED*[5]) markers. Commercially available systems are *ARTTrack*[6], *Polaris*[7], and *Flashpoint*[8]. Other indoor tracking methods are magnetic, ultrasound, and inertial tracking. Each of these tracking methods only work in a limited workspace. Outdoor and large environment tracking systems are mostly limited to positional data received from *global positioning systems* (*GPS*)[9] or those of less strict accuracy requirements.

1.3.3.4 Registration of the Real and Virtual Worlds

The registration of the real and virtual worlds, often referred to as *calibration*, estimates the parameters for the transformation and/or projection of one coordinate system onto another. To describe a rigid transformation of one coordinate system onto another, three rotational and three translational parameters need to be computed. In addition, an affine transformation contains three parameters for scaling in three dimensions. Most calibration routines in literature assume that there is no screw of the axis and no distortion. Screw and distortion would introduce additional parameters to the transformation matrix. This parameter space is often referred to as 'camera model'.

In optical tracking systems the projection matrix of the camera needs to be computed, which is referred to as camera calibration. Among others (see [30, 37] for an overview) *Tsai* developed a practical method [80] that decouples the internal and external camera parameters and computes them sequentially.

1.3.3.5 Applications in Medicine

The intra-operative visualization of anatomical and functional structures appears to be one of the most promising applications for augmented reality; namely computer-generated virtual objects are superimposed on the real world view [24]. The vision of augmented reality in a medical context is to allow physicians to see directly into the patient's body, to create a 'transparent patient'. This vision provides new opportunities in minimally invasive and totally endoscopic surgery, where the physician is aided by real-time graphic overlays.
In general, a typical augmented reality application in medicine can be split up into three parts:

1. Planning the minimally invasive intervention

[5] *Infra Red Light Emitting Diode* – A semiconductor device that emits visible infrared light when charged with electricity.
[6] by *Advanced Realtime Tracking GmbH* `www.ar-tracking.de`
[7] by *Northern Digital* `www.ndigital.com/polaris.html`
[8] by *Image Guided* `www.imageguided.com`
[9] *GPS* is a satellite-based radio navigation system such as the American *NAVSTAR*, the Russian Federation *GLONASS*, or – soon – the European *GALILEO*.

2. Spatial registration of the real patient and different virtual modalities
3. Augmented reality supported execution of the surgery using the planned information

Additional fields of simulation and training, which are not considered in this context, are possible, but lean more towards an entirely virtual environment. Nevertheless, those fields can be combined with any augmented reality application.

A scenario for a planning system might be helping the physician to find the exact incisions and to define the volume of interest, e.g. locate the margins of tumors, and other critical structures. This additional information is used to guide the physician intra-operatively.

In order to provide a perfect fusion of two modalities they have to be spatially registered, which allows a powerful navigation interface. The fundamental idea of spatial registration is to perfectly align a model of the patient with the real patient in the spatial domain. For instance, the model can be created by a three-dimensional reconstruction of a computed tomography scan, magnetic resonance imaging [29], ultrasound [62], or a registered mixture of different modalities.

After the registration of all involved modalities, the virtual overlay has to be visualized for the physician. In current scenarios the augmentation is intra-operatively used for the overlay of the anatomical and functional interior of patients onto their surface, for instance in laparoscopic [26] and endoscopic [14] surgery. This task needs to be performed within real-time constraints, which mainly consider the frame rate per second (*fps*) and the latency of the system.

Augmented reality applications for intra-operative guidance and navigation pose additional challenges and limitations compared to industrial augmented reality applications. For instance, a limitation is the acceptance of currently available head-mounted displays by physicians in a futuristic operating theater, originating from their cumbersome and bulky handling. An additional challenge is the spatial registration and tracking of deformable objects. In most cases the transformations are no longer rigid, affine, or projective, but curved and, therefore, more complex and difficult to estimate.

One example of applying augmented reality technology in the surgery room is the *MEDARPA* project (see section 3.3). Another example is the *Varioscope AR* [23, 24], which is a head-mounted microscope modified for augmented reality applications, developed by *AKH*[10] *Universitätsklinikum Wien* and *LifeOptics*[11]. Using image reflection prisms the computer images are merged into the optical path of the microscope. Since the display device is integrated into the head-mounted operating microscope, the physician does not have to deal with any additional instruments.

1.4 Goals and Classification of the Entire System

As described in sections 1.1 and 1.2, a few problems arise when using the *da Vinci* telemanipulator. This project addresses them: optimal ports for placing the teleoperator arms need to be found and applied, supported by a view showing the interior of the patient to locate the

[10] *Allgemeines KrankenHaus*
[11] `www.lifeoptics.com`

teleoperator arms. This work tackles the issue of planning the ports and teleoperator arms in an optimal way, whereas *Jörg Traub*'s work [79] focuses on the intra-operative augmentation of these previously and optimally planned data.

For the joint work methods provided by the mathematic society are used and results fed to the medical society (see figure 1.5). Thereby, the main focus is on applying existing algorithms in computer vision to the medical domain to solve challenges on the way to the operation room of the future.

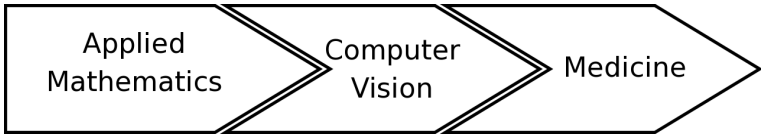


Figure 1.5: Mathematicians provide their field of research as an application that is used by the computer vision society. Moreover, the computer vision society provides applications for the medical community. This work tackles the use of existing computer vision methods and adopts them for the specific medical needs.

1.5 Structure of the Document

The remainder of this document is organized in the following way. Chapter 2 analyzes the current port placement process at the *Deutsches Herzzentrum München* and presents an alternative visionary scenario of this process. It aims to reveal and remedy the current disadvantages and to consolidate the requirements, which are compulsory for a functional planning system.

The very first step towards the development of a new system usually is the search for similar and related work. Chapter 3 presents a collection of work done in the research field of robotically assisted minimally invasive cardiovascular surgery.

Chapter 4 gives an overview on the developed port placement planning system. Its functionality not only covers the port planning process itself and its validation, but also preliminary steps such as image segmentation, three-dimensional reconstruction, and real-time visualization. However, the planning system, which is based on scene graphs, follows the object-oriented programming methodology, which is also depicted.

Image segmentation is a tedious and time-consuming process. If the mathematical and physical background of images is understood, various techniques can be applied to speed up and improve the accuracy of the segmentation process. These techniques are briefly described in chapter 5.

Several aspects such as geometric computations and collisions need to be considered for the planning system. Chapter 6 serves this purpose by providing an insight to implementation details, which are relevant to a system that requires efficiency and real time.

Finally, chapter 7 summarizes and discusses the outcome of this work, along with evolving problems, and makes recommendations on directions and future work.

2 Requirements Analysis

> It requires a very unusual mind to
> undertake the analysis of the obvious.
>
> Alfred North Whitehead

In order to design a planning system for port placement in robotically assisted minimally invasive cardiovascular surgery a selection of requirements for the system need to be defined. It can be found by assessing the shortfalls of the current situation and improving them through a visionary scenario and several use cases, leading to a set of functional, non-functional, and pseudo requirements.

2.1 Current Situation

Currently, the cardiac surgery setup at the *Deutsches Herzzentrum München* involves the use of the *da Vinci* telemanipulator from *Intuitive Surgical*. The placement of the three arms of the telemanipulator (i.e. one camera arm and two instrument arms) into the patient's body is done without knowledge of its precise location in the patient's interior. The only basis, upon which the planning of the port placement is carried out, is provided by computed tomography slices that are studied by the surgeons. Based on the surgeons' memory and imagination in the course of the planning process they place the arms at their desired position and orientation. At present, the telemanipulator itself supports only the operation, but not the precise placement of the ports. Therefore, a system might be helpful to plan and guide the placement process. Furthermore, during the whole operation the surgeon has no precise knowledge of the arms with respect to the patient's interior.

2.2 Proposed Situation

In the following sections, a visionary scenario and use cases are given to describe how the current port planning situation using a *da Vinci* telemanipulator system can be improved by an interactive planning tool.

2.2.1 Visionary Scenario

In the visionary scenario surgeons use an interactive planning tool. Initially, the planning tool supports the surgeons while segmenting the previously acquired computed tomography slices, so anatomical structures such as tissues, bones, and organs can be extracted. Additionally, a three-dimensional model can be reconstructed and visualized. Using this model

of the patient, the planning system proposes an initial fit for the placement of all three teleoperator arms based on a selected region of interest. This initial fit can be manually adjusted by the surgeons.
The placement process in the operation room is supported by augmented reality techniques, which enable the projection of the placement location and orientation through a real-time video stream of the patient's thorax, displayed on an external monitor. Throughout the operation the reconstructed model of the patient's thorax will be augmented by the teleoperator arms and screened on an external monitor in the operation room [79].

2.2.1.1 Iterative Steps

First, a region of interest, i.e. the area, in which the operation will be performed, is marked on the reconstructed model. To reach this region of interest, all teleoperator arms are inserted only one time. Once the region is confirmed, the planning tool computes the best fit for the teleoperator ports. If the supposed fit is not satisfactory, it can be adjusted interactively.

2.2.1.2 Obtaining Best Fit

The decision process of finding the best fit is limited by the distance of the teleoperator arms to organs, the accessibility of the whole region of interest, and the available/allowed poses of the teleoperator arms, limited by the shoulder, rip, and ilium (hip) bones. For any computation all of these constraints need to be fulfilled. An optimal solution will be found consisting of a maximal distance to organs, and a wide margin of the poses of the teleoperator arms as well as the region of interest.
If the fit is adjusted interactively, all constraints will be checked as well. In the case of a violation a warning will appear.

2.2.2 Use Cases

Use cases generally describe the planning system's functionality. Formally, each use case consists of a flow of events and communicates with participating actors, as illustrated in figure 2.1. The use cases Acquire3DModels, PlanPortPlacement, ValidatePortPlanning, and ExportPlannedData, which can be found in appendix A, form all functions of the planning system and interact with human (Surgeon), robotic (TelemanipulatorArms), and electronic (CTData and SceneGraphData) actors.

2.3 Resulting Requirements

As depicted in the aforementioned scenario and use cases, the following requirements can be stated, divided into functional, nonfunctional, and pseudo requirements.

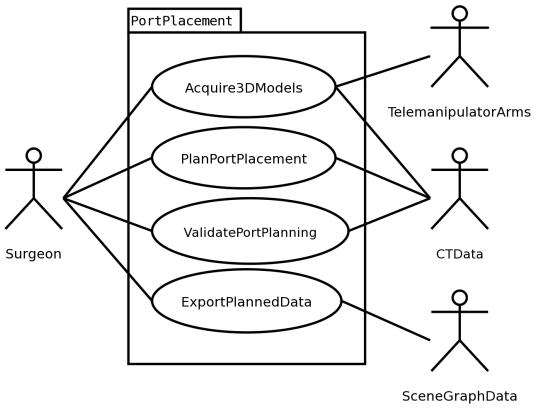


Figure 2.1: Use case diagrams for the port placement process

2.3.1 Functional Requirements

Functional requirements refer to the mutual influence of the planning system and its environment such as the user and any other external systems. All functional requirements can be separated from its implementation.

- The surgeon will be able to reconstruct and visualize a segmented three-dimensional model of the patient.
- The position of the telemanipulator arms and the patient's model will be displayed and aligned with computed tomography data.
- The surgeon will be able to switch between several oriented and scalable views like axial, coronal, and sagittal planes, forming a three-dimensional view of the scene, and a virtual endoscopic view.
- The surgeon will be able to set a *region of interest* (*ROI*) to define the area where the cardiac operation will take place. It is within this region that the tool tips of the teleoperator arms are placed.
- The surgeon will be able to set the pose of the instrument arms and the endoscope of the teleoperator through their corresponding ports.
- The surgeon will be able to validate the planned data to ensure the freedom of collisions between telemanipulator arms and bones or vitals.
- It will be possible to visualize the tracked telemanipulator arms during the operation.

2.3.2 Nonfunctional Requirements

Nonfunctional requirements, which are not related to the functional behavior, refer to facets of the planning system that are obtainable and observable by the users. Facets such as error

handling, extreme conditions, security, and the clinical environment are not considered at that stage of the planning system, since it is in the prototype phase.

- The *user interface* (*UI*) will provide an intuitive way to navigate interactively within the application as well as display the surgeon's requested views of the patient's model.
- The planning will be performed within real-time constraints, leading to a faster response from the planning system to any user commands.
- The developed system will be platform-independent, so both *Windows* (*2000/XP*) and *Linux* (*Suse/Debian*) operating systems can be supported.
- Source code will be commented and documented satisfactorily.

2.3.3 Pseudo Requirements

Pseudo requirements are requested by the client, namely the *Deutsches Herzzentrum München*. Their only pseudo requirement is that the system will be realized using amiraDev[1], which is a three-dimensional visualization and volume modeling tool. It allows the user to add or create new components for visualizing and processing data based on existing modules. In order to extend the amiraDev core, the development will be done using the *C++* programming language. However, for the sole deployment of the system amira is sufficient. The development and deployment will be done on a graphics workstation, which supports hardware-accelerated *OpenGL* three-dimensional graphics as well as hardware texture mapping and features at least a 500 *MHz* (*megahertz*) *Pentium III* processor, 128 *MB* (*megabytes*) of main memory – preferably 512 *MB* or more, and 512 *MB* or, for improved performance, 1 *GB* (*gigabyte*) of virtual (main and swap) memory.

[1] `http://www.amiravis.com`

3 Related Work

> When a man tells you that he got rich through hard work, ask him: 'Whose?'
>
> Don Marquis

This chapter focuses on similar and related work done in the field of robotically assisted minimally invasive and totally endoscopic cardiovascular surgery. Also, applicable techniques of virtual and augmented reality in other medical fields are considered as related work for the application design. As explained previously, the planning system is part of a joint work providing a complete solution for planning ports and teleoperator arms and their intra-operative augmentation. The differences between the related work and our joint work will be discussed in detail at the end of sections 3.1 to 3.5, while section 3.6 characterizes the research project our work is a part of.

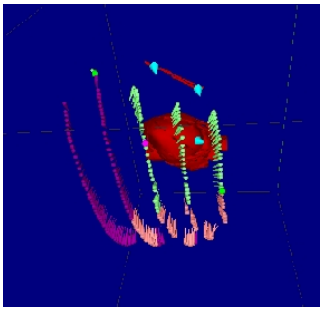
3.1 ChIR team at INRIA Sophia-Antipolis

The research group *ChIR - Surgery, Computer Science and Robotics* at *The French National Institute for Research in Computer Science and Control INRIA* in Sophia-Antipolis recently published parts of their work on robotics surgery [4, 20]. They focus on planning and simulation of robotic procedures, wherein models of deformable organs are created and augmented reality techniques are used to enable real-time and safe integration. The emerging tools are experimentally tested in hospitals in Paris and Leipzig.

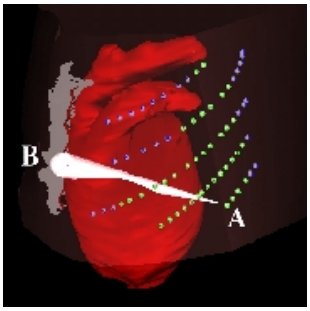
To model deformable organs, bones, and soft tissues several imaging modalities such as X ray, magnetic resonance imaging, and computed tomography are combined to get a four-dimensional model of organs: a three-dimensional model and a further dimension in time, e.g. representing the beating heart.

The planning and simulation process is as follows: At the beginning, three incision sites for the telemanipulator arms are computed. They are chosen from a set of admissible points, depending on factors like optimal accessibility and comfortable position for the surgeon. In addition, target points are chosen, which represent the surgeon's region of interest (see figure 3.1). Next, a corresponding position of the telemanipulator and its arms are found to avoid collisions between the arms. When all positions for the incisions and the robot are identified, the whole surgery can be simulated.

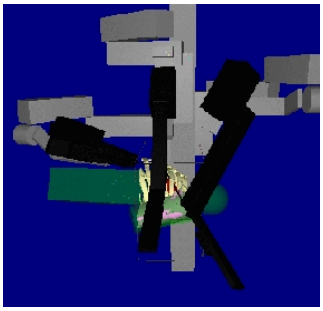
After registering the pre-operative and the robotic data with the operating room environment, the planned surgery is applied to the patient in the real operating theater, supported by augmented reality techniques. Initially, the planned incision sites are back transformed on the patient's skin. Therefore, a three-dimensional reconstruction of the patient's torso is



(a) Planning of the best triplet for the arms. First the optimal pose for the endoscope and then the two instrument arms.



(b) A test series predicts the adequacy of the planned arms.



(c) Another test if the planned arms are suitable for the robot.

Figure 3.1: The planning procedure of the *ChIR* team at *INRIA*.

registered with the patient's actual torso (see picture 3.2). An intra-operative goal is to superimpose the endoscopic view with cardiographic information to improve the process of finding the target points.

ChIR developed the prototypic software *STARS* (*Simulation and Transfer Architecture for Robotic Surgery*) to perform the described planning and simulation process.

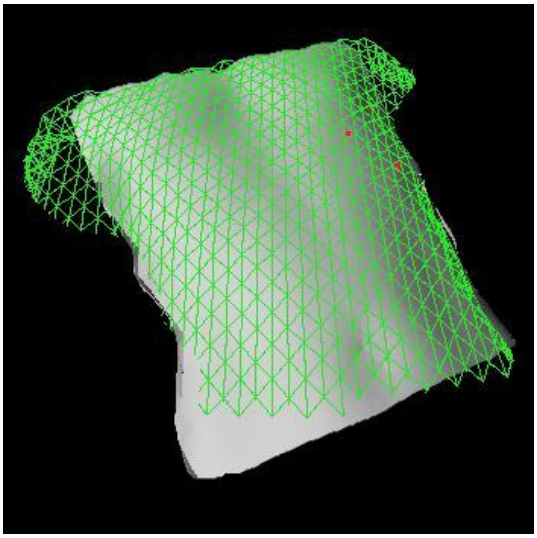
Further details on the project can be found on their project web page[1], a more detailed version of their work in *Louaï Adhami*'s PhD thesis [3].

As an official consultant of *Intuitive Surgical Inc*, the *ChIR* team is able to access the internal parameters of the *da Vinci* telemanipulator via its application programmer's interface. Thus, the exact positions and orientations of the telemanipulator arms and their tool tips can easily be computed using forward computation and be used for the planning process. Hence, collisions between the arms can be detected without any additional effort.
However, compared to our system, *ChIR* is unable to plan any robotic intervention except those that are *da Vinci* based. Our system stays robot independent, even for tracking issues. While our system tracks all instrument arms individually, *ChIR* only needs to track the pose of the robot. Again, the telemanipulator arms and tool tips are determined using forward computations. One big disadvantage of using this system is the tracking of assistant arms, which are not controlled by the robot. In our system we track every single arm using an optical tracking system.

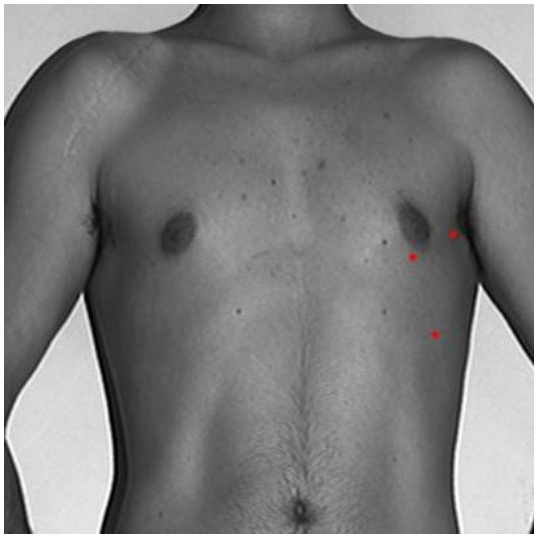
3.2 ORTHODOC

The *ORTHODOC* workstation by *Integrated Surgical Systems*[2] is a pre-operative planning system to simulate orthopedic surgery, which supports the surgeon by selecting and positioning

[1] `www-sop.inria.fr/chir/`
[2] `www.robodoc.com` – *Integrated Surgical Systems* unveiled *ROBODOC* in 1992, being the first robotic system in surgical applications.



(a) The segmented and three-dimensional reconstructed model of image data is registered with the patient using an iterative closest point algorithm.



(b) After aligning the model with the patient the incision points are projected on the patient's skin.

Figure 3.2: The intra-operative model registration of the *ChIR* team at *INRIA*.

an optimal hip or knee implant. It is tightly coupled to the *ROBODOC* surgical robot, which is performing the hip or knee replacement autonomously. *ORTHODOC* is able to display computed tomography or X-ray images along with implant models in three dimensions to the surgeon, who can simulate the whole operation and generate a pre-operative surgical plan. This plan includes all relevant data for the following intervention using *ROBODOC*.

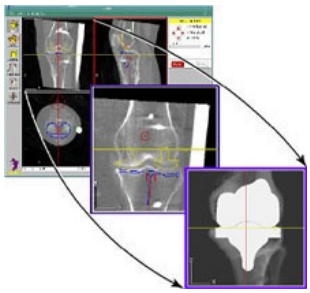
ORTHODOC only aligns slices of computed tomography scans or X-ray images with implant models in a three-dimensional space without first segmenting them and reconstructing a model of specific anatomical structures. The advantage of orthopedic surgery in general is that few critical structures can interfere with the instruments. Nevertheless, this needs to be addressed in our work, since an instrument arm must not collide with any bones or vitals. *ORTHODOC*'s planning system is also limited to the single arm of *ROBODOC*, whereas our system needs to support as many as four telemanipulator arms.

3.3 MEDARPA

The *MEDARPA*[3] project of the *Fraunhofer Institute for Computer Graphics* (*IGD*) in Darmstadt aims to develop a workspace for medical interventions, wherein augmented reality techniques are applied on a so-called *AR-window*. Present augmented reality solutions for medical interventions mainly require a head-mounted display, which usually comes with cameras and wires, and is therefore bulky and awkward in a surgical scenario. The *IGD* research group replaces the head-mounted display by the *AR-window*, a semi-transparent display,

[3] *MEDical Augmented Reality for PAtients* (`www.medarpa.de`)

(a) The *ORTHODOC* pre-operative workstation, which is used to plan the insertion of hip or knee implants.



(b) The graphical user interface of *ORTHODOC*, which is able to display X-ray and computed tomography images. These images are used to position models of implants.

(c) The *ROBODOC* surgical robot, which is actually performing the intervention.

Figure 3.3: The *ORTHODOC* and *ROBODOC* systems by *Integrated Surgical Systems*, which are used in orthopedic surgery.

which is room-mounted on a swivel arm. The wires remain, but they are covered inside the swivel arm. The *AR-window* is able to superimpose the real patient data virtual model, provided that the virtual model is registered with the physical world. Therefore, the pose of the *AR-window* and the head of the surgeon need to be tracked [69].

Data acquisition and visualization methods (see section 1.3.1) are applied to fuse different imaging modalities such as computed tomography and magnetic resonance imaging, combined with the definition of spatial reference points. Techniques of registration and tracking enable a three-dimensional alignment of the previously acquired data with the real patient, whose pose is detected along with the instruments, physician, and display. An intuitive communication and interaction channel to the workstation is provided by multimodal technology, like foot switch or visual and acoustic feedback.

Applications in several medical fields such as puncture, needle biopsy, cancer treatment, brachytherapy, endoscopy, and minimally invasive cardiac surgery are proposed. To evaluate and improve the *MEDARPA* workstation a cooperation with three German hospitals has been established.

In the minimally invasive cardiac surgery scenario, one step is the intra-operative port placement. While looking through the *AR-window*, a so-called Endo-stabilizer, which can be seen as a 'tracked stick', is used to set the ports for the telemanipulator arms. Whenever it touches the patient's skin, a virtual port is set on the Endo-stabilizers' tool tips. A virtual axis corresponding to the telemanipulator arm's axis is aligned with the Endo-stabilizer and extended into a virtual model of the patient. If this axis is hitting a region of interest, it will be highlighted and the information can be saved. Once this procedure is repeated for all ports, the teleoperator-based operation can be conducted after swivelling away the *AR-window*, wherefore the planned axes can be used.

The *MEDARPA* system performs the port placement process intra-operatively, i.e. while

(a) The hardware setup of the *MEDARPA* system using the swivel arm in a real operating theater environment. The surgeon is holding an Endo-stabilizer.

(b) The *AR-window* is a semitransparent combiner plate. This enables the projection of virtual objects on top of the real world view. While looking through this window, the ports can be planned intra-operatively, wherefore the Endo-stabilizer's axis (blue) must intersect the region of interest (red).

Figure 3.4: The operation room setup of the *MEDARPA* project by the *IGD* group.

the patient is in the operating room. The advantage of our approach to plan the port placement pre-operatively is the decrease in total operation time. Additionally, *MEDARPA* does not consider intersection tests to detect collisions between critical organs or bones and robot arms, whose dimensions are disregarded, since they are only represented as lines in the system. However, our system is capable of reproducing the exact robot arm dimensions, along with computational collision detection techniques.

3.4 BrainLAB

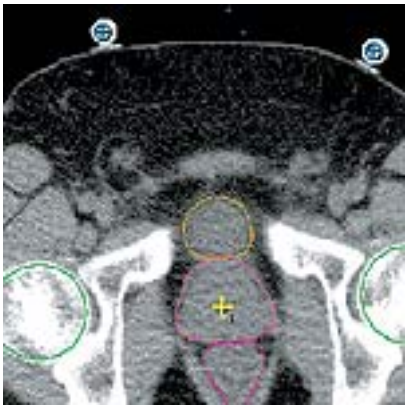
A commercially available product in virtual simulation and spatial registration of patients with their image data, acquired by a computed tomography scan, is the patient positioning system *ExecTrac* by *BrainLAB*[4]. Adhesive skin markers are attached to the patient's body, which are visible in both the real and virtual modalities. The markers can be detected in computed tomography scans by image processing techniques and in the real world modality by an infrared tracking system. *BrainLAB* proposes an application in combination with an optimal cancer treatment planning tool (see figure 3.5) to achieve high accuracy in tumor irradiation. Practical experience with this system was reported by the *Department of Radiotherapy and Urology* at the *Academic Hospital, Free University of Brussels* in Belgium [73, 72].

The virtual simulation and patient positioning system of *BrainLAB* is currently used for radiotherapy in cancer treatment and trauma surgery. There is no described scenario in the cardiovascular or any other strongly deformable surgical field. However, *BrainLAB*'s virtual simulation system only needs to allow for the definition of a treatment position aligned

[4] `www.brainlab.com`

(a) Markers placed on the patient during computed tomography.

(b) The markers are visible in the computed tomography images.



(c) After the patient positioning and planning, the patient gets an optimal cancer treatment.

Figure 3.5: Components of the *ExecTrac* system by *BrainLAB*, a virtual simulation and patient positioning system.

with critical structures and tumor volumes. Since our operation scenario is based on solid minimally invasive instruments and not on beams, we additionally need to consider physical constraints such as possible collisions. Most steps conducted prior to the actual planning, such as image acquisition and three-dimensional reconstruction, are similar to our system.

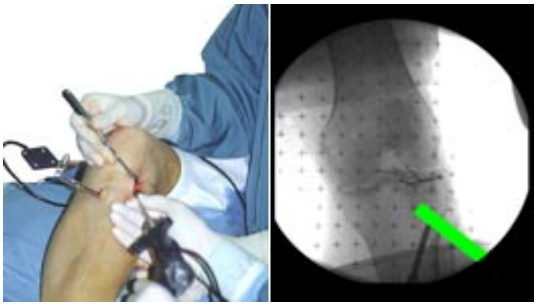
3.5 Medivision

Another commercially available application, which is solely for intra-operative navigation, is provided by the Swiss company *Praxim-Medivision AG*[5]. Its focus is on orthopedic surgery, where solutions for head [56], spine [82], pelvis [40], hip, and knee [67] surgery are proposed. Throughout the entire application domain an infrared tracking system is used for the computation of the pose of any instrument. A reference coordinate frame is rigidly fixed to the region of interest, for instance a certain bone. Two different processes for intra-operative navigation are addressed.

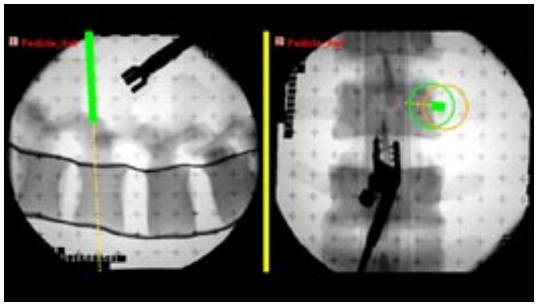
The first method consists of pre-operative data acquisition by computed tomography or magnetic resonance imaging modalities, transfer of the data to the navigation system, preparation of the patient by adding a reference coordinate frame to the region of interest, matching of the image data to the patient's anatomy, which is also referred to as spatial registration, and, finally, performance of the image guided surgery (see figure 3.6).

The second method does not require the matching of the previously acquired image data to the physical space of the patient's anatomy. Since no additional registration is required for the matching, it is also called registration-free navigation. Image acquisition is done by a tracked *C-Arm* and, therefore, a pre-registered three-dimensional model can be obtained.

[5] `www.medivision.ch`



(a) The *Medivision* system for knee surgery. On screens the physician is guided by augmenting the image data with the visualized real-time pose data of the tools, e.g. a drill.



(b) Screen shots of the *Medivision* system for spine surgery. Different views are provided for the physician.

Figure 3.6: The visualization subsystem of the *Medivision* system provides different views for physicians. The tools are tracked in real time and superimposed (green line) into the two-dimensional image data of the patient.

Overall, four steps are traversed. Firstly, the patient is prepared in such a way that a reference base is attached rigidly to the region of interest (e.g. a screw into the bones). Hence, the reference coordinate system is relative to the region of interest. The pose of all instruments is measured with respect to this reference. Secondly, the image data is acquired by a three-dimensional C-Arm. Infrared markers are attached to the C-Arm in order to estimate the external camera parameters in the base coordinate system. Thirdly, the images are downloaded from the C-Arm to the *Medivision* guidance system. Lastly, the operation is guided.

All described scenarios are mainly for rigid structures in the field of trauma and neurosurgery. The extremely deformable thorax is not addressed in their application domain. Being similar to our complete system, the planning part is omitted in the *Medivision* system. Only a two-dimensional view is available during the intervention. Taking a modified version of our planning system and adding it to the first scenario of intra-operative navigation would enrich and simplify their navigation process by augmenting a three-dimensional reconstructed model of the patient and all planned data.

3.6 High-Fidelity Telepresence and Teleaction (SFB 453)

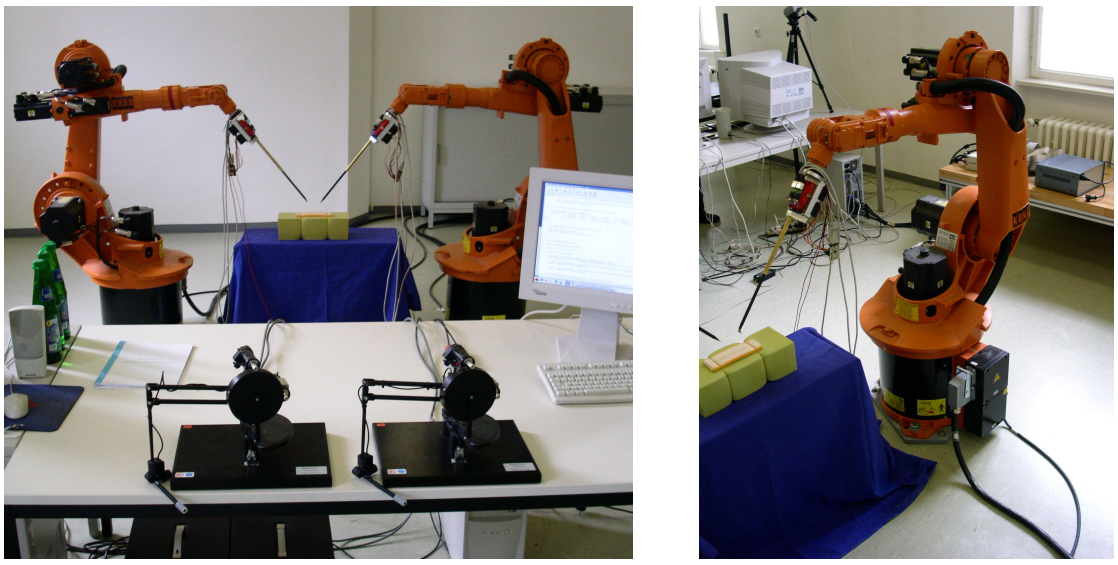
The presented work is part of the research projects of the *Sonderforschungsbereich* (*Collaborative Research Centre*) *SFB 453*[6], promoted by the *DFG - Deutsche Forschungsgemeinschaft* (*German Research Foundation*). All projects take on issues and problems of telepresence systems (see 1.2) and involve several research institutes, namely the *Technische Universität München*, the *Deutsches Zentrum für Luft- und Raumfahrt* (*German Aerospace Center*), the *Universität der Bundeswehr München* (*University of the Bundeswehr Munich*), and the *Technische Universität*

[6] `www.lsr.e-technik.tu-muenchen.de/sfb453/`

Berlin (*Technical University of Berlin*).
Their main focus is on coping with barriers between the operator and the remote environment that evolve from spatial distance, inaccessible environments, and scaling tasks. These barriers need to be bridged to get realistic impressions of the distant environment for the operator, which is done through modeling.
However, in order to connect the remote environment to the operator, not only are visual and auditory modalities important, but also haptic information such as kinesthetic (proprioception, forces) and tactile (pressure, temperature, roughness, and vibration) properties. To provide a way to interact via these haptic modalities, sensors and actuators are developed.

Last but not least, telecommunication and human-machine interaction focuses on the impacts of transmission of modalities and looks at the requirements of man for telepresence systems.

Two projects concerning robotically assisted minimally invasive heart surgery are worked on at the *Institute of Robotics and Mechatronics* of the *Deutsches Zentrum für Luft- und Raumfahrt* and the *Realtime Systems and Robotics* chair of the *Technische Universität München*. Their common goal is to develop partially autonomous functions for the teleoperators, so they can perform elementary actions commanded by the operator in a fast, though robust and reliable, way.



(a) The whole teleoperator system having two instrument arms (in the back) provides force feedback, which is achieved by two *PHANTOM Premium 1.5 A* devices (in the front), one for each arm.

(b) For now, only standard industrial robots are used. However, later on specific robots complying with medical standards can be developed.

Figure 3.7: The prototypic teleoperator system developed by the *Realtime Systems and Robotics* chair of the *Technische Universität München* aims to provide partial autonomy and haptic feedback.

The former project concentrates on the need for acquiring knowledge about the region of

interest where the operation is performed, namely the moving and flexible surface of the beating heart. These movements are captured using stereo image processing techniques. Interferences of the captured images, for example caused by occlusion, need to be dealt with using motion estimation algorithms. The outcome of this project will be a haptic training system, which can be used to instruct cardiac surgeons. It is supposed to provide the surgeons with three-dimensional video images and time synchronous haptic information that is gathered from operations and animal experiments. An overall goal for the project is to prove the assumption that the training system, which combines visual and haptic modalities, will improve the quality of the surgeons' training.

The latter project centers the methods to get partially autonomous teleoperators. Partial autonomy is reached by combining atomic, concatenated, sensor-based, and parameterized action primitives such as gripping, cooperative sustaining, force controlled displacement, suturing, and knot tying, wherefore haptic feedback is compulsory.[7] Skill-transfer-techniques will be used to transfer these primitives from the human to the teleoperator, i.e. the teleoperator will be capable of partially imitating human motion and operation sequences. Additionally, the teleoperator will try to predict any further steps without breaching the surgeon's ability to intervene in the actions. All developed methods and functions will be tested experimentally, particularly in the minimally invasive operating scenario.

[7] Haptic feedback is provided by two *PHANTOM Premium 1.5 A* devices, which are produced by *SensAble Technologies* (`www.sensable.com`).

4 Fundamentals and Characterization of the Planning System

> The whole is more than the sum of its parts.
>
> Aristotle

The following chapter serves two purposes, namely to describe the port planning process as well as to illustrate the underlying structure of the planning system. As such, the functionality and use of amira is presented in section 4.1, which is the system's graphical core, extended by the port planning tool. It is very important to understand all concepts of amira, since it affects all further design steps of the planning system. Thus, the first part characterizing the planning system is devoted to amira. Having established a general understanding of amira, the developed planning tool can be used to conduct all compulsory planning steps (cf. section 4.2) to perform a successful port placement. More specifically, the required steps are perception (cf. section 4.3), port planning (cf. section 4.4), validation (cf. section 4.5), and export (cf. section 4.6).

4.1 Basics of amira

As defined earlier in section 2.3.3, amira (version 3.0) is chosen to be the graphical core for the implementation of the planning tool. It is able to visualize and reconstruct three-dimensional data that comes from image acquisition methods such as computed tomography, magnetic resonance imaging, and ultrasound scans, which are 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. Additionally, it is capable of importing and exporting three-dimensional file formats like *Open Inventor* or *Virtual Reality Modeling Language* (*VRML*). Unusually large data sets are visualized with satisfactory speed. Unfortunately, loading these sets requires an enormous amount of *RAM* (*Random Access Memory*), leading to unforeseen problems.[1]

As illustrated on figure 4.1, amira's basic user interface consists of three windows. The main window consists of the context sensitive menu structure, a so-called *object pool* (cf. section 4.1.1), and a visual representation of all parameters and *ports* (cf. section 4.1.2), which can be made available and visible by selecting a certain *module* or *data object* (cf. section 4.1.1). A viewer window is capable of visualizing three-dimensional graphical data, which

[1] In the planning system amira was sometimes crashing 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*.

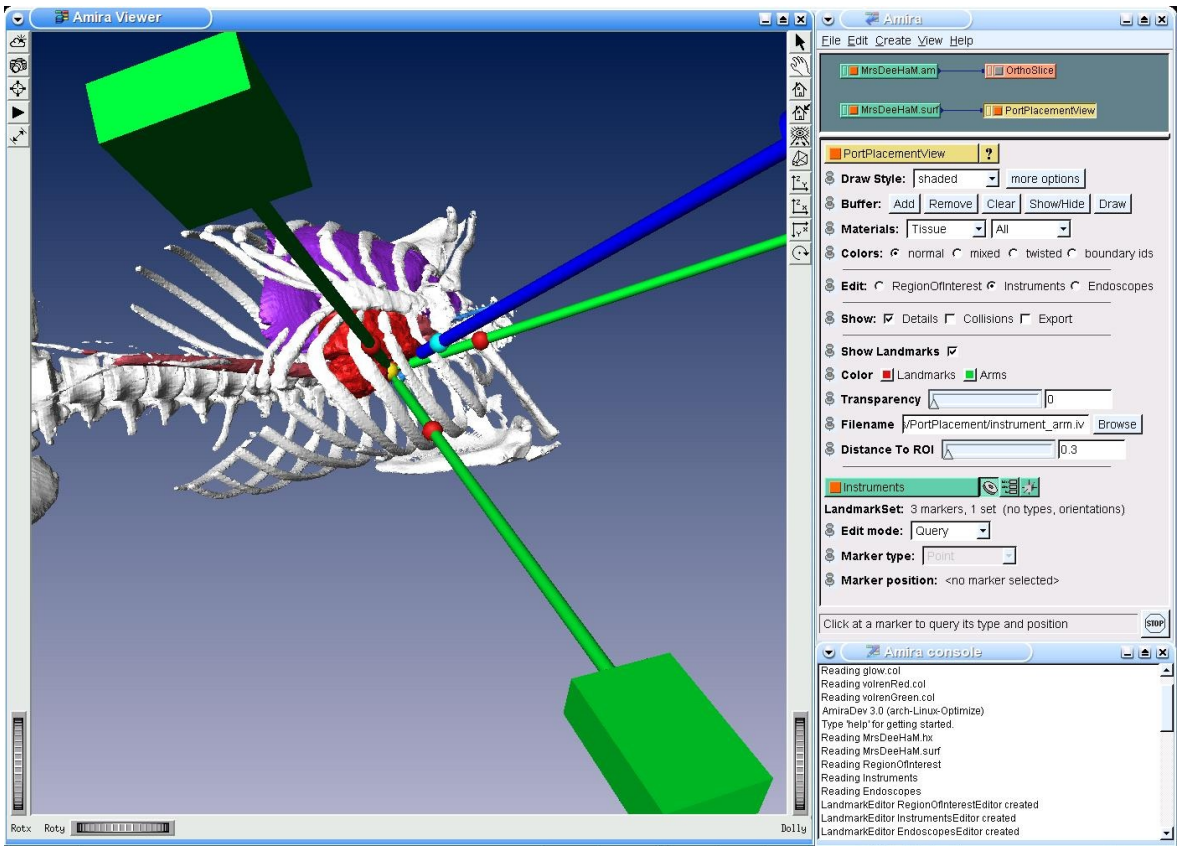


Figure 4.1: amira's graphical user interface consisting of a main window (top right), a viewer window (left), and a console window (bottom right).

can be divided into sub windows or extended to additional viewer windows. The viewer window provides two modes, namely an examiner mode for navigating within the currently displayed scene and an interactive mode for manipulating the scene. The initially smallest window is representing a console window serving as an input/output *Tcl* interface. The whole functionality of amira along with its structure is described in the following sections. An extensive description can be found in amira's *User's Guide and Reference Manual* [38].

4.1.1 The Modular Philosophy of 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 the data some modules are designed to perform computations on the data.
In order to make use of these modules and data objects, they are generated and placed into the object pool, their central management repository. Inside the object pool they are represented by separately colored icons. Data objects are green, display modules are yellow or orange, and compute modules are red. If modules or data objects are interrelated in any way, their corresponding icons are connected by blue lines via *ports* (see section 4.1.2).
In the following sections a selection of modules and data that were crucial to the system is worked out in detail.

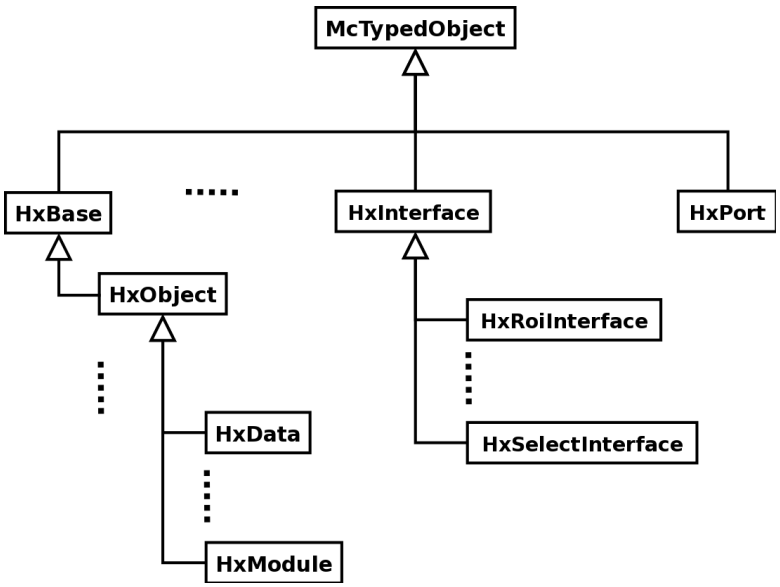


Figure 4.2: The top layers of amira's class hierarchy.

Figure 4.2 shows the top of amira's class hierarchy. The class `McTypedObject` provides a typing facility, which, in particular, enables programmers to perform type-checking during runtime. Two other important classes are the directly derived `HxBase` and `HxPort`. `HxBase` is the base class of all objects in amira, except for the user interaction classes, which are all derived from `HxPort`. Apart from this hierarchical object-oriented approach, amira also offers some global objects for performance reasons.[2] For printing to a *Tcl* console provided by amira, the global variable `theMsg` instantiated from `HxMessage` can be used offering a *C*-like `printf` output syntax. In contrast, for executing, adding, and removing new *Tcl* commands, or setting, getting, and linking *Tcl* variables the global pointer `theInterpreter` can be accessed. 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`. The class `HxWorkArea` along with its global instance `theWorkArea` grants manipulative access to the section in the amira window where all ports and progress bars are displayed. Most importantly, controlling amira viewers and thus scene graphs is provided by `theController`, an object of `HxController`. With this instance, new viewers can be created, and scene graphs can be edited, manipulated and saved. Besides these global objects, amira adheres strictly to the object-oriented paradigm, which will be described in the following.

4.1.1.1 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 range from *JPEG* (*Joint Photographic Experts Group*), *GIF*

[2] Only the global objects that are relevant to the system are introduced.

(*Graphic Interchange Format*), or *BMP* (*Bitmap*) to standard formats found in different scientific areas such as the medical *DICOM* format, the proprietary *Bio-Rad Confocal Format* for biological microscopy, or the *DXF* (*Drawing Exchange 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 are derived from the superclass `HxData`, which implements generic load and save functions. Three-dimensional data are derived from a subclass of `HxData`, `HxSpatialData`, which are divided mainly into fields (like vector or scalar fields) and point sets (representing a set of three-dimensional points in space that can be manipulated). The two classes `HxField3` and `HxVertexSet` reflect this distinction. Figure 4.3 shows the classes of interest inheriting from `HxData`, `HxField3`, or `HxSpatialData` respectively. These are highlighted briefly below.

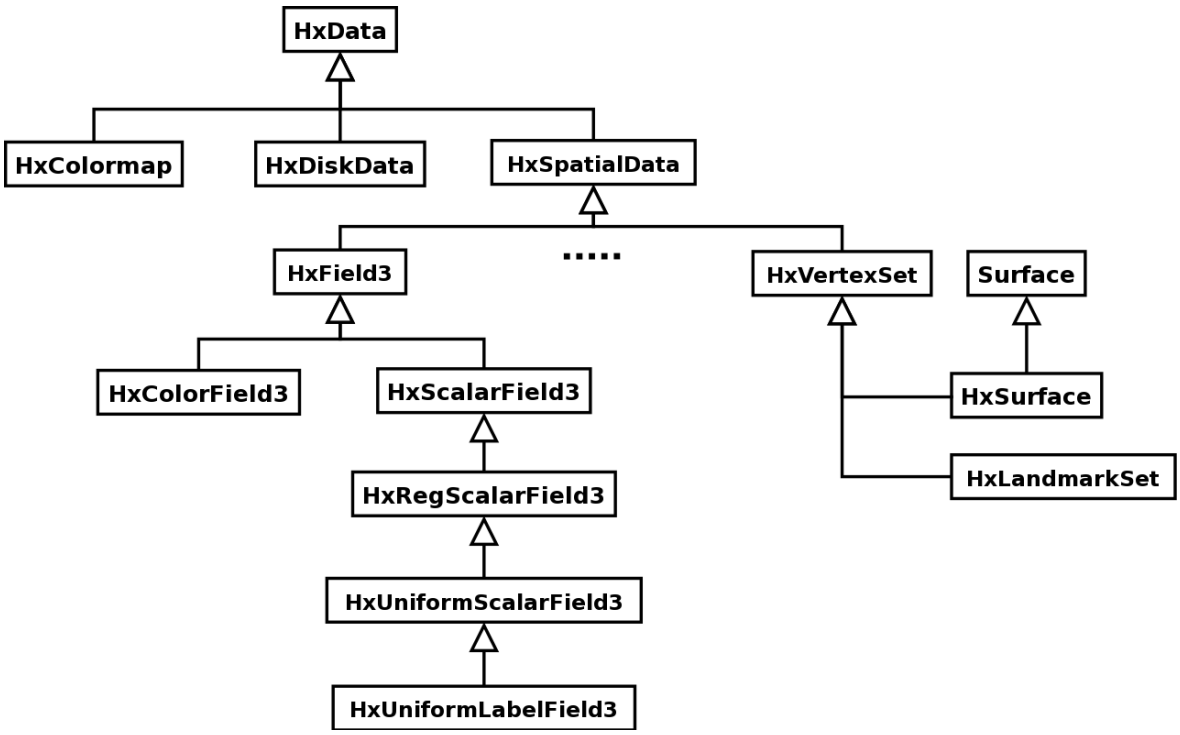


Figure 4.3: The class `HxData` and its derived classes.

- The *ColorField3* data type is a three-dimensional field with an *RGBA* value 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* described in the next section.
- A *Colormap* stores 256 different *RGBA*-tuples 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 (which are mostly much more than 256) to 256 gray 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 three-dimensional space in a uniform lattice. A field can be accessed at any point in a three-dimensional domain unlike discrete scalar arrays.
- Labeled 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 three-dimensional space. This is actually most important for aligning (registering) data. amira already provides methods for registering two three-dimensional data sets, like two surfaces by setting landmarks.
- The *LargeDiskData* type allows the loading of 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 three-dimensional coordinates identifying the triangles' vertices, additional information can be stored, like materials, patches, contours, or edges. 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 used 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 color maps. 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 the *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 criteria 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 exported 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.

4.1.1.2 Display Modules

Display modules visualize data given by data objects that encapsulate any files, which 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. There is a tight correlation between the scene graph methodology described in 4.4.1.2 and the display modules. Figure 4.4 shows the common superclass of all (display) modules, `HxModule`. Only those classes that are presented below are charted.

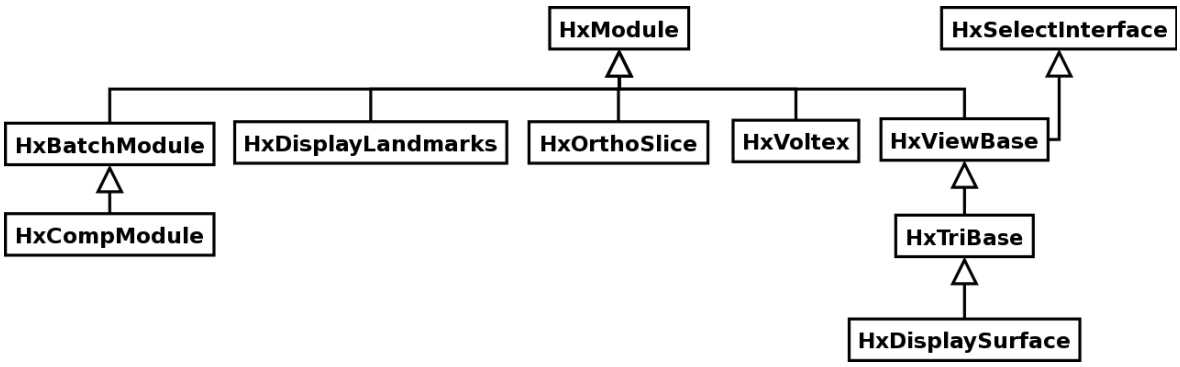


Figure 4.4: The fundamental class `HxModule`, which is most important for developing own modules.

- *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 degenerating.
  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*. Apparently, 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 three-dimensional 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 suggests, the cuts are orthogonal, i.e. three cuts for x-, y- and z-axis 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 in each direction, i.e. the adjacent slices are shown when moving a slider.
- The *HxVoltex* module creates an object in three-dimensional space from volumetric data using direct volume rendering. It's performance is quite good, if two-dimensional, ideally three-dimensional 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.

4.1.1.3 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 it can be extracted from figure 4.4, 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 the 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 4.5 enlists all the compute modules that were consulted in the system.

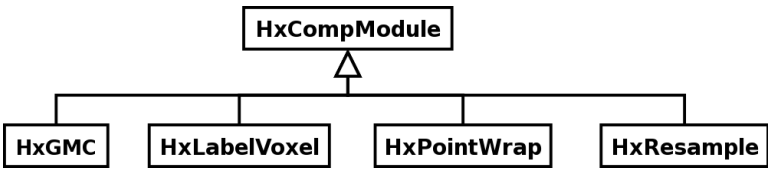


Figure 4.5: The basic class `HxCompModule` and a selection of important compute modules. `HxGMC` is the name of the *SurfaceGen* module.

- 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 gray scale.

- A very useful module is the *Resample* module that allows the up- or downsample of three-dimensional fields. Often, 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, but 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 form a triangle. The *PointWrap* module meets these requirements by creating a sphere of an arbitrary radius, 'dropping' it down an axis 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.

4.1.2 Communication Ports

In amira , graphical interaction between users and objects is enabled by 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 via a *HxConnection*, which is a port. The general design and some important ports will be presented in the following.

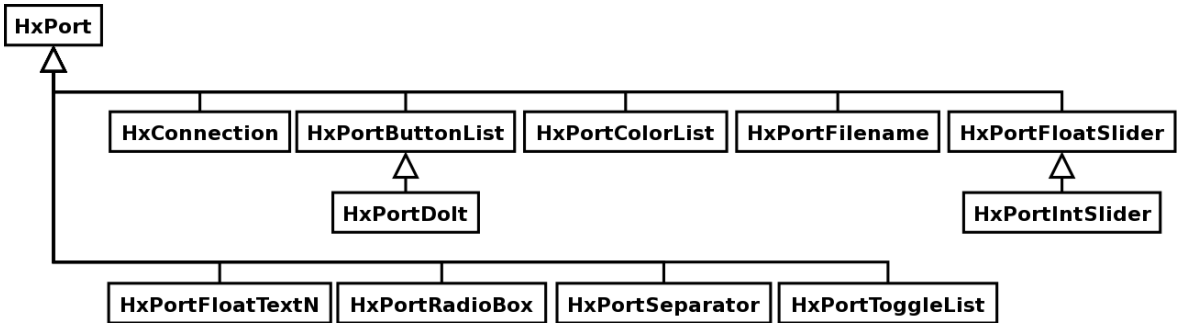


Figure 4.6: The parent class `HxPort` and its subclasses providing several ways for interaction.

- *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 by its graphical user interface, like button lists, radio and toggle boxes, or integer and float sliders as shown in figure 4.6.

4.1.3 Development with amiraDev

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* (*Tool Command Language*) 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 6.1.2). amiraDev is not only using *Tcl*, but also other platform independent industry standards such as *Open Inventor*, *OpenGL*, and *Qt*. 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. It provides a three-dimensional scene database, node kits, manipulators, and a three-dimensional interchange file format (see section 4.4.1.2). *Qt* is another toolkit for the *C++* development of graphical user interfaces 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 and deployment 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.

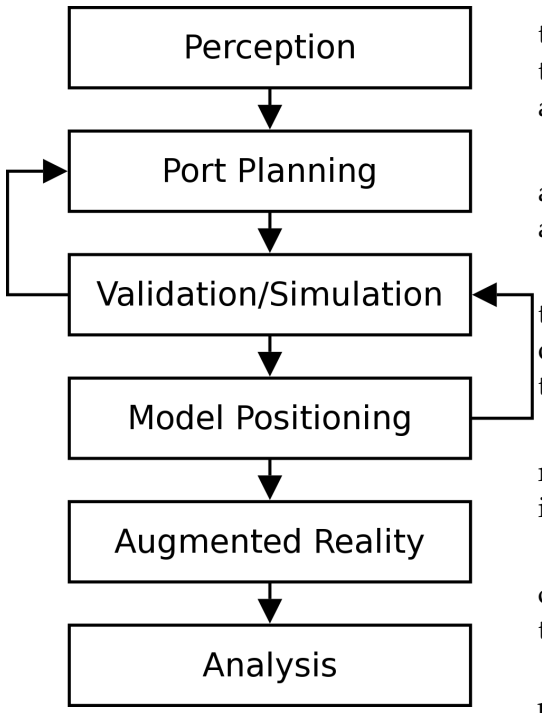
By completing all steps demanded by the development wizard the framework for a new module is automatically created, which is ready for extension and compilation.

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`.
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 initializing 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 not stored when saving a network, unless an extra *Tcl* function is provided. Hence, an amira developer must either extend amira's *Tcl* interface or avoid using parameters without ports.

4.2 System Overview

The whole system supports planning, placing, tracking, and visualizing the teleoperator arms. To achieve optimum improvement in robotically assisted minimally invasive cardiac surgery the following process will be traversed, as illustrated below.



Perception deals with the creation of a virtual three-dimensional model of the patient and the telemanipulator arms and its visualization on any display (see section 4.3).

Port planning concerns the process of finding adequate incision sites for the telemanipulator arms (see section 4.4).

Validation and simulation takes into account techniques to validate the planned incision sites or the quality of positioning the model, respectively (see section 4.5).

Model positioning is applied when the virtual model containing all planned data is positioned in the real world's operating theater.

Augmented reality techniques are deployed to combine the virtual model with tracked data of the real world.

Finally, *analysis* tries to improve the process by identifying, rating, and evaluating certain factors (see section 7.2).

This work concentrates on perception, port planning, and its validation and simulation, whereas *Jörg Traub*'s work [79] deals with model positioning, patient registration, and the appliance of augmented reality techniques.

4.3 Perception

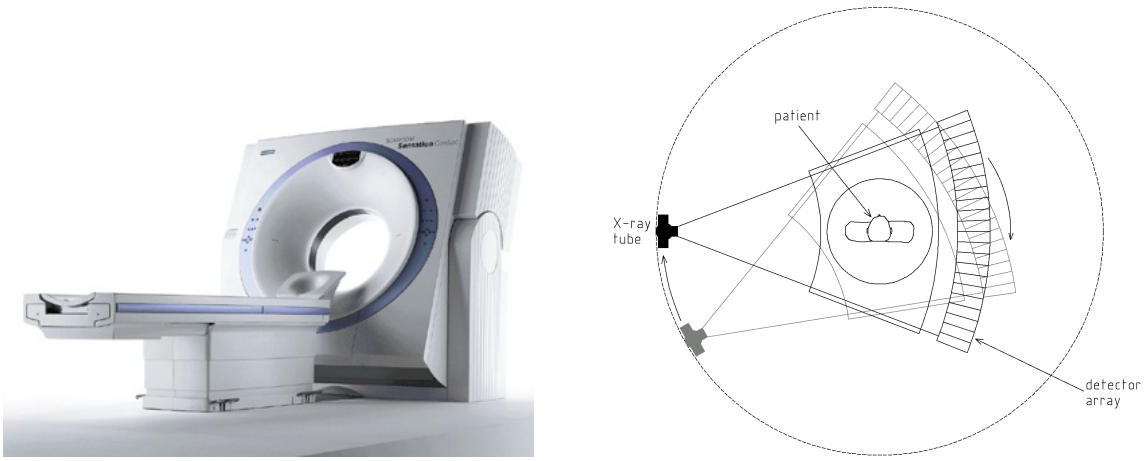
Before performing an operation, a surgeon wants to non-invasively gather as much information about the patient's interior as possible, so a region of interest among anatomic structures such as bones, organs, and tissues can be identified and the surgical intervention can be planned. Unfortunately, it is impossible to obtain an identical representation of the patient's tissues and interior at once. First, the cycle illustrated in figure 1.2 needs to be passed through. It shows the medical imaging techniques data collection, classification or segmentation, three-dimensional reconstruction, and visualization, which are all applied to extract the interior of a patient.

For planning teleoperator-based operations additional knowledge about the dimensions of the telemanipulator is required, as both a three-dimensional model of the patient and one of the teleoperator arms are used in the planning process. However, obtaining all details

about the robot arms is easily accomplished by measuring their dimensions. Afterwards, a three-dimensional model of the telemanipulator arms can be constructed easily.

4.3.1 Data collection

Data collection means the generation of a digital representation of the subject that needs to be worked on, i.e. the patient. Since the desired kind of data is visual, its best representation is a three-dimensional geometric model, which will be generated at a later stage (see sections 4.3.2 and 4.3.3) from the acquired image data. As mentioned in section 1.3.1.1, a variety of methods such as computed tomography, magnetic resonance imaging, and ultrasound support the process of data collection. Unfortunately, the promising technique of magnetic resonance imaging is very expensive and calcium is not visible on the images, and therefore conditions such as coronary calcification cannot be recognized. Ultrasound images on the other hand are only two-dimensional in nature. Hence, for acquiring sampled volume data of the patient computed tomography methods are used in the planning system.



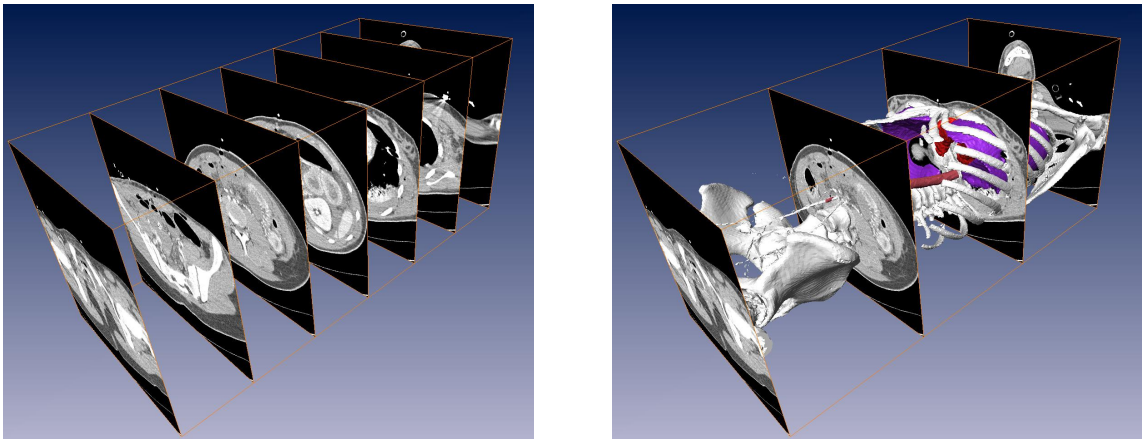
(a) The computed tomography scanner *SOMATOM Sensation Cardiac* by *Siemens*, which is used for the image acquisition process of the planning system.

(b) The functionality of a third-generation fan-beam scanner, using a rotating source and detector array.

Figure 4.7: Illustration and functionality of a computed tomography scanner.

Since the late 1960s and early 1970s, when computed tomography was invented by the British engineer *Hounsfield* and the American physicist *Cormack*, several generations of computed tomography systems have evolved. Widespread and commercially available systems are based on the third-generation technology. A third-generation computed tomography scanner is capable of producing several image slices of the patient, which are all lying on coplanar planes. For a single plane, the patient is exposed to a series of X rays with origin at various points around a 360-degree arc. As figure 4.7(b) shows, the X rays that form a fan beam are emanated by an X-ray tube and detected by a detector array, which consists of a few hundred (or thousand) contiguous elements. While permeating the patient's body, the radiation intensity of the X-ray beams is diluted. Applying an algorithm to the intensity

measurements made in the array of detectors, an image of absorption values at each point in the plane can be reconstructed. Each different absorption value is represented by a different gray scaled color value. Repeating the process of beaming X rays for each plane to be imaged gives a stack of two-dimensional images, also called cross-sectional axial slices, which usually have a distance of 0.75 to 10 millimeters from each other. Figure 4.8(a) clarifies these terms. [48]



(a) A stack of two-dimensional axial slices, acquired by computed tomography techniques.

(b) The process of reconstruction can be clarified by combining a three-dimensional model with a few original computed tomography slices, which were used to build the model.

Figure 4.8: Image acquisition and model reconstruction based on computed tomography techniques.

Specifically, the *SOMATOM Sensation Cardiac* by *Siemens* is used for the image acquisition process of the planning tool (cf. figure 4.7(a)). It is a computed tomography system for multislice spiral scanning. Multislice spiral scanning has the capability to dramatically increase the scan speed and lower the radiation dose compared to conventional computed tomography techniques. Instead of acquiring stacks of parallel images, it is able to simultaneously translate the patient while rotating the X-ray tube for scanning, resulting in spiral or *helical* scans, respectively. Given the above, the accuracy of imaging is increased and any position within the acquired volume can be visualized. However, the scanning system internally converts the spiral images to cross-sectional axial slices, which can be saved, thus conforming to the *DICOM* standard (cf. section 1.3.1.1).

4.3.2 Classification/Segmentation

Each axial slice, which has been acquired before, consists of a gray scaled image, where the X-ray absorption values are represented by the gray scaled color values. The sagittal and coronal slices can be reconstructed from the axial data by interpolation, based on three-dimensional volume data consisting of voxels. Applying volume segmentation techniques to this data enables the classification of anatomical structures such as bones, tissues, and

organs.

However, segmentation in medical context is considered to be very difficult to implement. Difficulties arise, when very complex and variable anatomic structures need to be extracted from large image data. Image data is usually affected by errors of imaging modalities such as sampling artifacts, noise, and low contrast, generally resulting in imprecise, disconnected, and indistinct boundaries of organs and tissues. Hence, a major challenge of all segmentation techniques is to extract the proper boundaries of anatomical structures.

Because currently available segmentation techniques do not fit every purpose, they cannot provide acceptable results for an automatic segmentation of every type of medical image data. Thus, very often interactive techniques are used, which combine automatic and manual techniques and complement structural techniques with stochastic techniques [39, 65].

The presented planning system is based on precisely these complementing components. First, a basic stochastic thresholding technique is applied to the image data. Later on, a combination of edge detection, morphological, deformable models, and region growing techniques is used to classify anatomical structures.[3] Although at this point of the work only segmentation techniques are outlined, which are used in the planning system, a more comprehensive overview on commonly used segmentation techniques can be found in chapter 5. Getting to know the theory behind these techniques is an essential step to perform a successful segmentation.

4.3.2.1 Multi-Thresholding

For *thresholding*, the intensity of all voxels that are occurring in a volume data is measured, where a so-called histogram is created (see figure 4.9) to visualize intensity and occurrence. Multiple regions are specified interactively by setting multiple thresholds to partition voxel intensities. Voxel intensities in between two thresholds are grouped together into one region. Usually local minima are selected as thresholds.

Thresholding is very sensitive to noise and inhomogeneity with regards to its voxel intensity. Volume data acquired by magnetic resonance imaging and ultrasound has exactly these properties, but not volume data acquired by computed tomography. If the volume data has a very good contrast between regions, thresholding will be a very effective technique.

4.3.2.2 Additional Manual Segmentation Techniques

Once thresholds are applied to the volume data, a set of additional, mainly manual segmentation methods based on edge detection, morphological techniques, deformable models, and region growing is provided. Figure 4.10 shows the component of the planning system, which is able to further segment the volume data.

Edge detection techniques try to identify local edges, which arise from two intersecting regions with different voxel intensities, by applying differentiation. It is very likely that local minima or maxima of the first derivative of voxel intensities are edges, since the image gradient is optimal for decreasing or increasing intensities. Furthermore, if these local optima

[3] Additionally, a segmentation of markers may be performed, which is helpful for registering the patient intraoperatively [79].

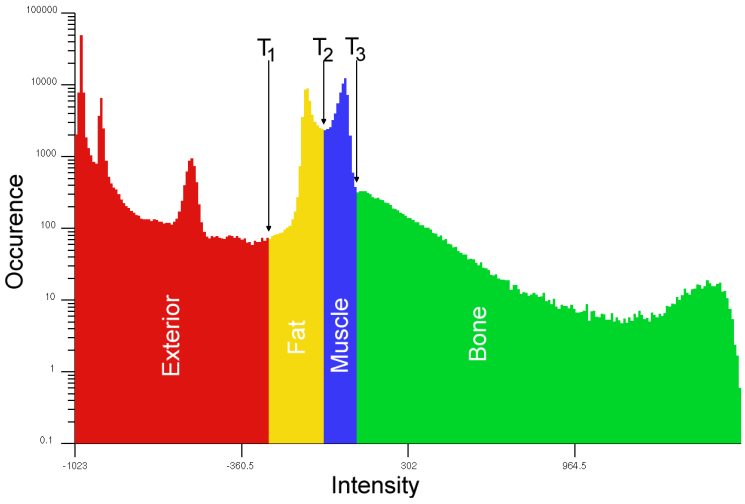


Figure 4.9: Histogram of volume data showing the number of occurrences of voxel intensities. Three thresholds are set to separate the volume data into the four regions exterior, fat, muscle, and bone.

are combined and grouped together, boundary contours can be identified to separate region voxels from remaining voxels. For approximating the detection of edges, algorithms based on *Robert's Cross*, *Sobel*, and *Prewitt*, or the *Laplacian of Gaussian* can be used [22].

Morphological techniques rely on set transformations for image analysis. Making use of morphological techniques enables the user to fill holes in regions, which can arise when applying automatic segmentation techniques such as thresholding. Therefore, the transformations of dilation and erosion are applied to expand regions and shrink regions, respectively. A series of dilations followed by erosions fills holes of certain regions, whereas a series of erode operations followed by dilations introduces holes [70].

Deformable models such as curves, surfaces, or solids are influenced by internal and external forces and, hence, deformable. In physical context, image data applies external forces to a deformable model, causing the model to move towards the image data and its boundary. The internal forces of a model itself prevent it from an abnormal deformation, so the model stays smooth. The most popular and widely used form of deformable models is called *snakes*. Snakes are planar deformable contours using an energy formulation, which tries to minimize the weighted sum of internal and potential energy [36, 76].

Also *region growing* is capable of segmenting image regions. It relies on previously defined connecting criteria, for instance edges or voxel intensity. An algorithm takes a manually planted seed point of the image as input data. Starting with this seed, the algorithm grows till the connecting criteria is violated. The algorithm can grow not only into two, but also three dimensions [89].

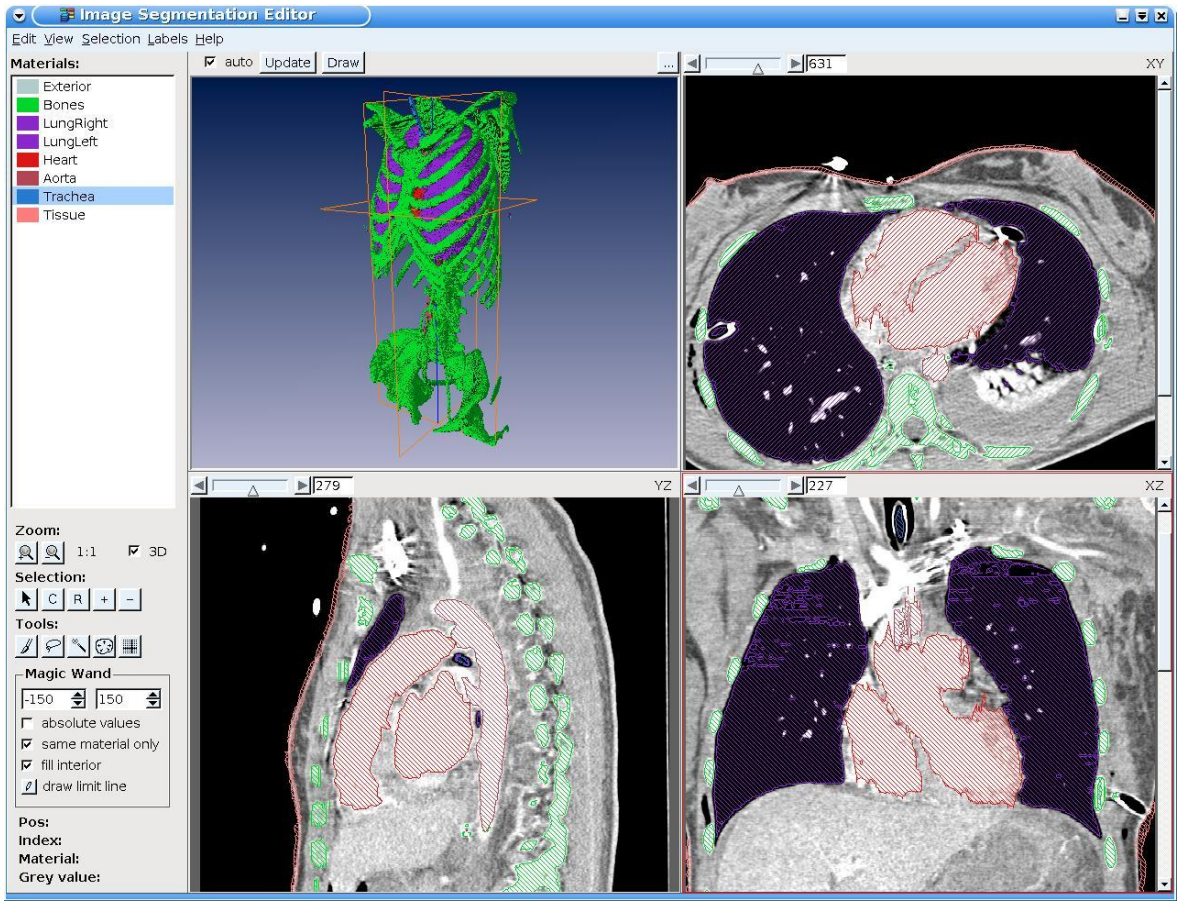


Figure 4.10: The planning system's component providing several segmentation techniques.

4.3.3 Three-Dimensional Reconstruction

The port planning process (see section 4.4) is executed in a purely virtual representation of the real surgery scenario, wherefore both the patient and the telemanipulator arms are geometric models. Geometric models can be divided into *polygonal* and *non-polygonal* objects such as *splines*, *algebraic surfaces*, *deformable bodies*, and *constructive solid geometry* (*CSG*) models.

Heart surgery requires a very precise planning process. Therefore, the geometric model that is constructed from the patient needs to be extremely diverse and fine-grained. To get an adequate representation of the patient and the robot arms, which can be displayed at a reasonable speed, a model based on polygon meshes is constructed. A polygon mesh consists of edges, vertices, and polygons connected such that each edge is shared by at most two polygons. In the presented planning system only triangles are used as polygons.

In order to get triangle meshes from segmented volume data the process of triangulation needs to be carried out. Triangulation is a special case of tessellation, i.e. meshes are formed by small triangles in a mosaic pattern.
The boundary voxels of the sections that were previously identified by segmentation techniques can be used to compute these triangle meshes. However, the computations are com-

monly based on *Delaunay triangulation* and *Voronoi cells* [15, 28].

Unlike that explained in section 1.3.1.3, the planning system does not use volume rendering techniques, which do not require reconstruction and are capable of directly displaying the volume data on the screen. Besides still being severely limited by their incompatible graphics display demand, volume rendering techniques lack the support for detecting collisions between the rendered volume and other models and, thus, are not chosen for this system. In further systems they might be considered as an alternative to the previous tessellation of a polygonal mesh.

4.3.4 Visualization

Planning the placement of the robot arms is done on a graphics workstation, which is attached to two flat screen devices for visualizing the planning process. Most devices that are used to display three-dimensional triangle meshes, such as monitors or flat screens, are only two-dimensional. Hence, a reduction of dimensionality needs to be accomplished, which is done by projections that transform three-dimensional objects onto a two-dimensional projection plane.

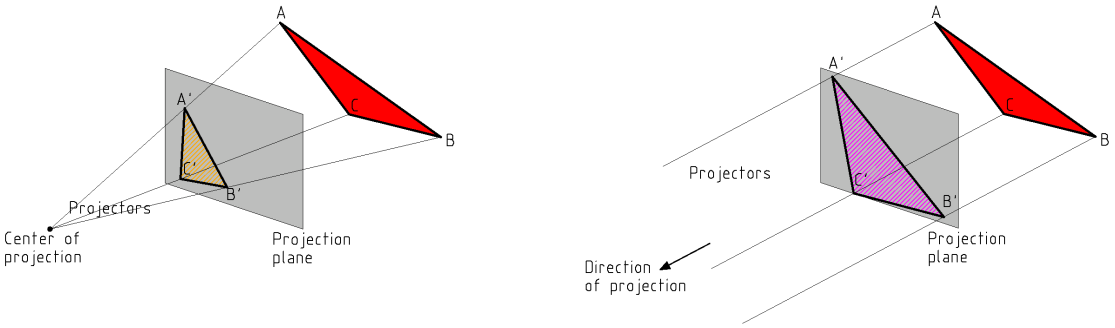
However, projections are only part of a conceptual model of viewing three-dimensional objects. Before the projection is executed, all objects in the three-dimensional world are clipped against a three-dimensional view volume in the world. After projecting the clipped world coordinates onto the projection plane, also referred to as window, the contents of the projections are transformed into the viewport in two-dimensional device coordinates for display.

In detail, a projection from three dimensions to two dimensions can be described as follows. Straight projection rays, also called projectors, are emitted from a center of projection. These projectors pass through each point of the object, intersecting the projection plane. Since the projection is onto a plane, it is also referred to as planar geometric projection.

Generally, the planning system provides two kinds of projection, namely the *perspective* and the *parallel projection*. If the distance of the center of the projection to the projection plane is finite, the projection will be perspective, if the distance is infinite, it will be parallel and a direction of the projection will be given instead of a center of projection. Figure 4.11 illustrates this distinction [25].

4.4 Port Planning

Generally speaking, the planning process is responsible for defining the region of interest, where the operation takes place, and finding the best possible positions of the ports for inserting the robot arms. However, the task of port planning can not be successfully accomplished until an appropriate virtual environment is set up that can be manipulated and interacted with (see section 4.4.1). Within this virtual environment the port planning process can be simulated and various configurations of the telemanipulator arms can be tested.



(a) Triangle ABC and its perspective projection $A'B'C'$.

(b) Triangle ABC and its parallel projection $A'B'C'$. The projectors AA', BB', and CC' are parallel.

Figure 4.11: Difference between perspective and parallel projection.

4.4.1 Setup of the Virtual Planning Environment

As mentioned before, planning of the ports takes place in an entirely virtual, simulated environment. A brief overview on its prerequisites and properties is given in the following sections, for instance, it is required to define and describe virtual models and arrange them within the virtual environment, so they form a reliable, fast, and reasonable virtual planning simulator.

4.4.1.1 Characteristics of Virtual Geometric Models

The geometric models acquired in section 4.3.3 can be stored easily as a collection of points and triangles. This stored data needs to consider several characteristics of geometric models:

- A *spatial layout* defines how geometric models are transformed. Transformations such as scalings, rotations, and translations enable the planning system to align all models that have their own, individual local coordinate systems to a general global coordinate system, also referred to as scene universe. Examples for rotations and translations can be found in section 6.1.1.
- Even though for triangle meshes their *shape* is obvious, sometimes it is more efficient to describe the shape of geometric models as primitives such as cones, cuboids, cylinders, or spheres. For instance, a good approximation of the dimensions of the telemanipulator arms can be achieved by a set of cylinders and cuboids.
- In order to identify and visualize different organs or distinguish between endoscope and instrument arms, certain *attributes* are assigned to geometric models. Attributes describe the appearance of models like transparency, color, textures, or material.
- Very often it is preliminary to *connect* or *group* single geometric models to one bigger model, as described in the example of a telemanipulator arm. The components of a bigger model then gain a topology, which is most commonly obtained by treelike structures or graphs.

- Finally, users of the planning tool want to add and remove, change, or *interact* with geometric models. For instance, if they pick a displayed geometric model with a mouse pointer, a callback function will be provided that responds to the users in any form.

4.4.1.2 Scene Graphs

All aforementioned characteristics can be put together to so-called hierarchical scene graphs. A scene graph consists of one or several nodes, each of which embodies a certain characteristic of a geometric model. There exist grouping, shape, spatial layout, attribute, or callback nodes. Graphical icons showing the different types of nodes can be seen on figure 4.12.

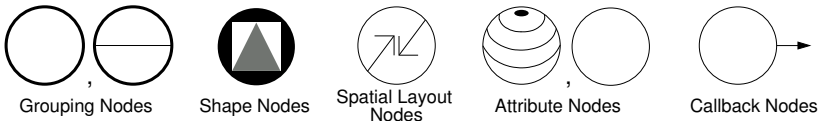


Figure 4.12: Some common nodes of a scene graph and their graphical representation.

These nodes can be composed to scene graphs in such a way that children nodes are attached to grouping nodes, which leads to a hierarchical, directed acyclic graph. Such a graph is illustrated on figure 4.13. At the same time, this graph is also showing a part of the design that underlies the planning system, as described in the following section.

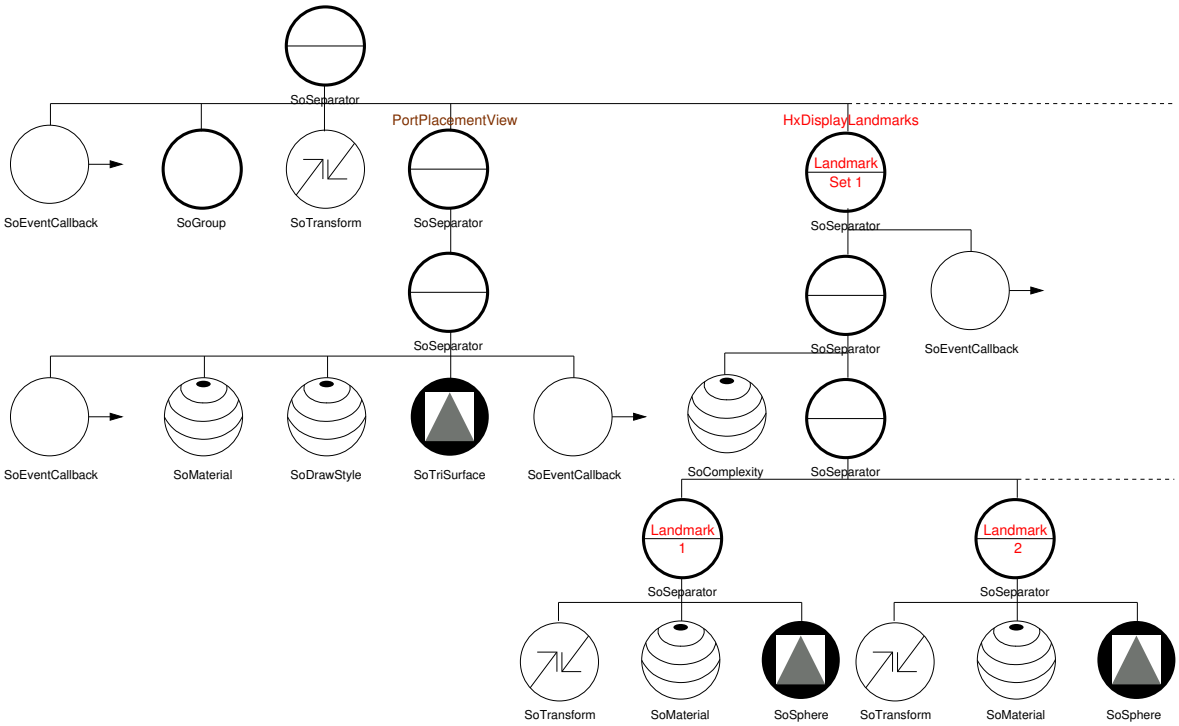


Figure 4.13: The basic scene graph used by the planning system.

Actually the nodes are objects fulfilling many object-oriented paradigms such as abstraction, encapsulation, modularity, and inheritance. All of these objects are stored as scene

graphs in a three-dimensional scene database, which is provided by *Open Inventor*, as mentioned in section 4.1.3. Taking all objects as entities of a scene database, their characteristics can be changed by modifying the database and, hence, its containing scene graphs. The objects of a scene graph are visualized by traversing the scene graph, beginning at the root node, from left to right and from top to bottom. Nodes to the right or down, respectively, in the scene graph derive the traversal state set from nodes to the left or above, respectively. In figure 4.13, for instance, the spatial layout node `SoTransform` as well as the attribute node `SoMaterial` are both applied to a `SoSphere` shape node before its visualization, because they are situated left of it. Whenever an object of the scene graph gets modified, all following objects get updated as well [88].

4.4.2 The Planning Process

Resting upon the manipulation of scene graphs the planning process can now commence. As stated in the use case on planning the port placement in section 2.2.2, first a three-dimensional model of the patient is displayed to the user of the planning system. All necessary planning steps are done on this model's triangle mesh that is shaping a surface, in figure 4.13 referred to as `SoTriSurface`.

Inside amira's object pool users can attach the `PortPlacementView` module to an `HxSurface` data object, which encapsulates a `SoTriSurface` scene node. They can interact with this module via the graphical user interface of amira that shows all available ports. Actually, the planning module provided by `PortPlacementView` is the core of the whole planning system.

`PortPlacementView` is derived from `HxDisplaySurface` and, thus, all its functionality can be used while displaying large Surface data. For instance, certain previously segmented materials such as organs, tissue, and bones, or only specific triangles can be hidden or shown, respectively. Additionally, the draw style can be adjusted, so all triangles of the mesh are filled with an opaque or transparent color. The users are also able to adjust their viewpoint on the surface mesh by moving the scene's projection center, being equal to a rotation or translation of a camera. However, the functionality and structure of the `PortPlacementView` module as well as its underlying scene graph get clarified throughout the planning steps, described in the following sections.

A general overview on the class hierarchy of the module is shown on figure 4.14. It can be seen that a `PortPlacementView` contains three landmark sets for defining landmarks (see section 4.4.2.1) and three `HxDisplayLandmarks` display modules for visualizing the landmark sets in the scene. The latter as well as a `PortPlacementView` display module each create the marked branch of the scene graph seen in figure 4.13, whereas an instance of `RobotArm` is created and attached to the scene graph (see 4.4.2.2) for each instrument arm that is planned. `HxLandmarkSet` and `HxDisplayLandmarks` are in amira's `hxlandmark` package, `HxDisplayLandmarks` and `RobotArm` form their own amira package called `PortPlacement`.

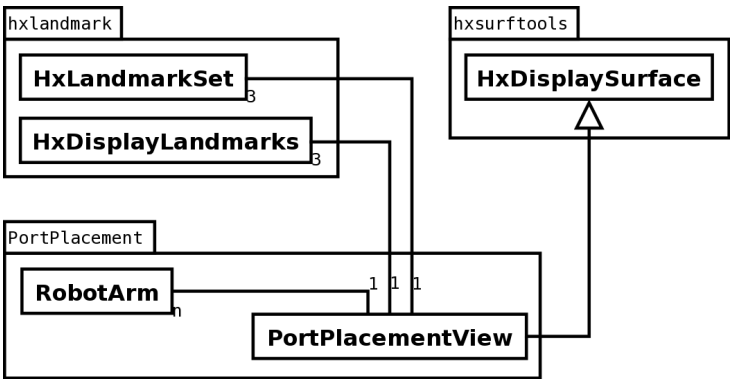


Figure 4.14: The amira module `PortPlacementView` is the core for the port planning process. It encapsulates three `HxLandmarkSet` data objects and three `HxDisplayLandmarks` display modules. Additionally, depending on the number n of instrument arms that are planned, `RobotArm` is instantiated n times.

4.4.2.1 Placing of Landmarks

A user of the planning tool places so-called landmarks on the triangle meshes of the model. First, a *region of interest* is defined by one or more landmarks to specify the region where the operation will take place (cf. figure 4.15(a)). Next, the landmarks for an endoscope arm and instrument arms are placed, which actually represent the *ports* (cf. figure 4.15(b)).

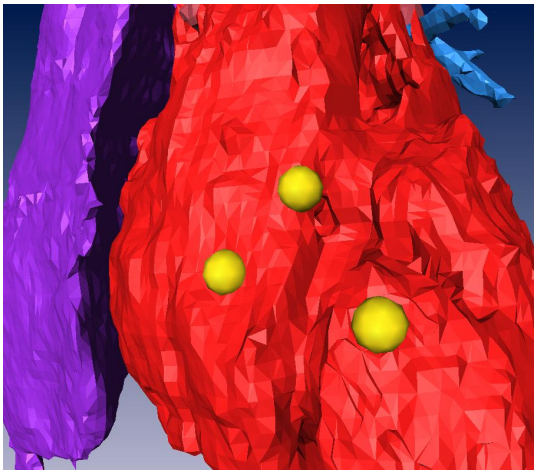
Therefore, three sets of landmarks are generated, one `HxLandmarkSet` per region of interest and per set of instrument arms and endoscope arms of the telemanipulator. Although the amount of landmarks in each set is unlimited, visual limitations occur due to the shape of a landmark, which is visualized by a `HxDisplayLandmarks` module as a small sphere (`SoSphere`), as defined in figure 4.13.
Besides having the possibility to adjust or query the position and appearance of landmarks interactively they can also be deleted.

The placement of landmarks is performed by selecting a pixel (a two-dimensional point) of amira's viewer window, e.g. by using a pointing device and clicking on the pixel. The viewer window can be seen as aforementioned viewport (see section 4.3.4), situated in the visualization process. From the center of projection rays that are staying within a certain radius on the viewport are emanated, so they form a cone. A landmark is placed on the closest of all intersection points between a ray within this cone and a triangle of the mesh.

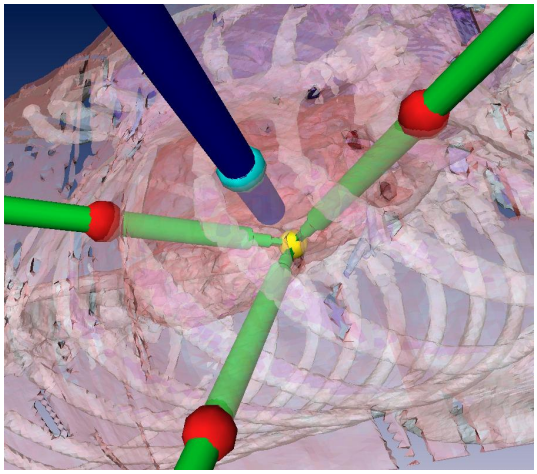
4.4.2.2 Visualization of Telemanipulator Arms

For each landmark that is in the set of instrument arms or endoscope arms, a corresponding telemanipulator arm is generated and visualized by creating an instance of the `RobotArm` node. Depending on the type of the arm, either an endoscope arm or an instrument arm can be instantiated.

Any telemanipulator arm is placed such that its altitude is intersecting with the corresponding landmark's center. It is directed towards the region of interest, which is defined



(a) First, the *region of interest* is defined by placing one or more landmarks. It describes the region where the operation will take place.



(b) Next, the landmarks for an endoscope arm (blue) and instrument arms (red) are placed, which actually represent *ports*. As soon as a corresponding landmark is set the teleoperator arms are visualized automatically in such a way that they do not intersect each other.

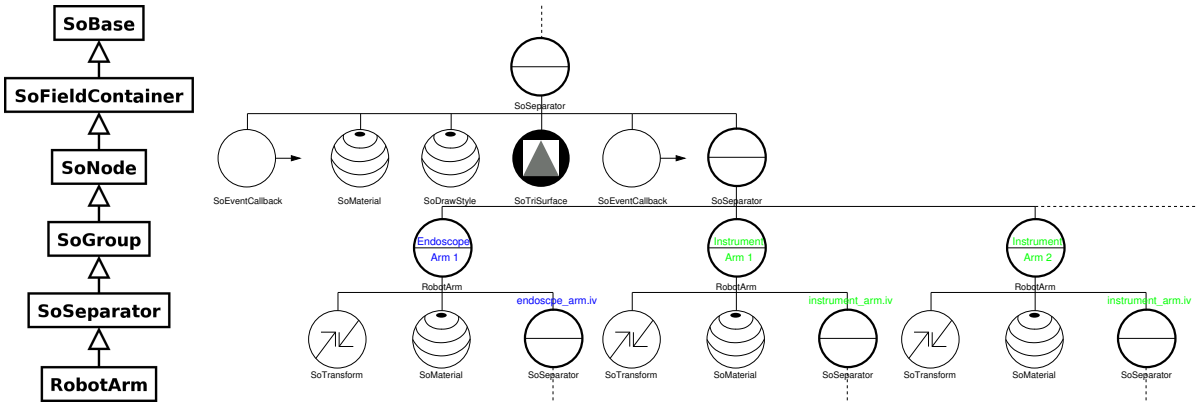
Figure 4.15: The process of placing landmarks, which are visualized as small spheres. The position of all landmarks can be adjusted interactively.

as the center of the bounding box that all landmarks are forming. A certain distance to the region of interest can be specified for all telemanipulator arms, as well as a specific color, transparency, and *Open Inventor* file.

In this *Open Inventor* file the geometric model of a telemanipulator arm is defined, containing a scene graph that describes all nodes. For instance, a simple telemanipulator arm having all previously measured dimensions can be constructed from cuboids and spheres. Two example files, one representing a *da Vinci* endoscope arm and another an instrument arm, can be found in appendix B. This file, which is representing the arm, is automatically loaded and its content inserted into the basic scene graph, when an instance of `RobotArm` is created. A `RobotArm` node is not only responsible for defining the shape of the telemanipulator arm, but also for performing the necessary transformations within the scene as well as attaching a certain colored and transparent or opaque material to the arm. Therefore, the `RobotArm` serves as an *Open Inventor* group node to connect all those characteristics. Thus, its functionality is derived from `SoSeparator` that itself inherits from `SoGroup` (see 4.16(a)).

Figure 4.16(b) shows a part of the basic scene graph that gets extended by one endoscope arm node and two instrument arm nodes, which is very common during a standard planning procedure.

When creating virtual telemanipulator arms, physical constraints of the real world need to be considered. For instance, two or more telemanipulator arms cannot intersect with each other, since they are made of solid steel. They cannot penetrate any rigid anatomical struc-



(a) (b)

Figure 4.16: The `RobotArm` class, situated in a hierarchy of *Open Inventor* nodes (a) and its multiple use as a set of scene graph nodes (b).

tures such as bones. The former problem is already addressed during the visualization of the robot arms, the latter problem is solved using collision detection techniques, as described in section 4.5.2. Because of these collision detection tests the telemanipulator arm nodes are all attached to a node within the triangle mesh's branch of the scene graph, so only this branch needs be tested for intersections, which usually is a very time-consuming process.

4.5 Validation/Simulation

Once all landmarks for the region of interest and the ports are proposed by a user, the validation phase can be started to verify the planned data. Therefore, two kinds of strategies can be adopted, namely a visually based and a collision detection based approach. Whenever any type of verification detects inadequate planning data, it can be re-adjusted by moving, transforming, deleting, or adding landmarks.

4.5.1 Visual Validation

Visual validation modes are dependent on the users' visual skills to observe the planned scene. However, auxiliary means are provided by the planning tool.

Making use of the *virtual endoscope* a view of the camera of the telemanipulator's endoscope arm can be simulated, whereby a specific distance of the camera to the region of interest can be selected and modified in real time. The virtual endoscope view provides the surgeons with a realistic operative view similar to the one acquired by *da Vinci*, which they are very accustomed to (cf. figure 4.18(a)).

Additionally, the original computed tomography data can be visualized attaching one or more `HxOrthoSlice` modules to it, for instance one per sagittal, coronal, and frontal orientation. Merging the three-dimensional reconstructed model with computed tomography

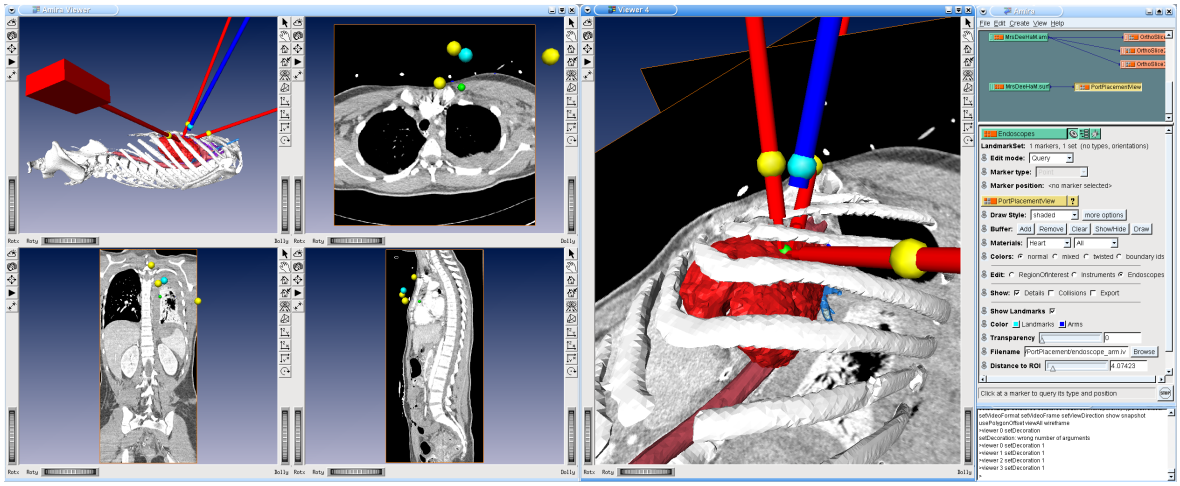


Figure 4.17: The planning tool allows for several views to show and validate the planned data. The planned data, the axial, sagittal, and coronal computed tomography slices, and the reconstructed model of the patient are all aligned to each other.

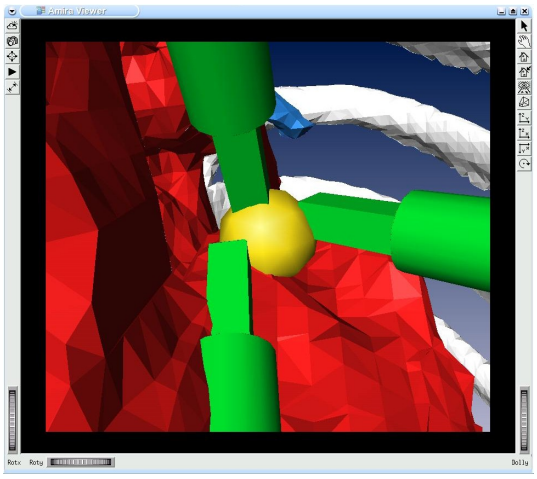
slices enables a surgeon, who is very accustomed to two-dimensional views, an intuitive way of validating the planned landmarks (cf. figure 4.17).

4.5.2 Collision Detection

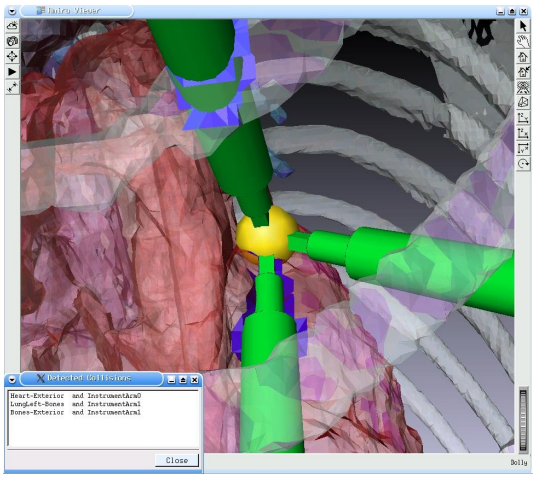
Imagine, a robot arm equipped with a scalpel is inserted into the patient's body through its corresponding port to perform a heart operation. However, the insertion is done blindly, since the arm is as yet not in the field of view of another endoscope arm. Unfortunately, after inserting it for a few centimeters it interferes with a vital. This situation can be avoided by the prior application of collision detection techniques, which are able to ensure the absence of collisions between robot arms and body parts such as bones and vital organs.

Collision detection, also referred to as contact determination or interference detection, is used to determine or predict the occurrence of a contact between geometric models such as polygonal meshes. It is not only used in the medical and general computer graphics field, but also in research areas like animation, computer simulated environments, robotics, automation, manufacturing, computer-aided design, and computer-aided machining (*CAM*). Many approaches such as spatial partitioning, analytical methods, algebraic formulations, optimization methods, geometric reasoning, and hierarchical representation are available. They all fit different assignments depending on the simulation environment, the query types, and the model representation [43].

The planning tool provides a simple interface for testing collisions between robot arms and vital organs or bones. Before testing collisions using the `PortPlacementView`, only important materials such as vitals or bones can be shown. For instance, the patient's skin and the left lung, which is closest to the heart, can be hidden, because the skin may be wounded and one lung is usually deflated during a cardiac operation. Performing collision detection computations on these materials and the telemanipulator arms, all colliding triangles are highlighted and their intersecting objects are described in a pop-up window.



(a) A *virtual endoscope* view can be activated, which simulates the view of the endoscope arm's camera. Therefore, a specific distance of the camera to the region of interest can be set and modified in real time. To be able to distinguish from the normal view, a black frame is shown.



(b) *Collision detection* techniques check whether any teleoperator arms collide with critical organs or bones. In the case of an intersection all colliding triangles are highlighted (blue). Additionally, a pop-up window shows a textual description of all colliding structures.

Figure 4.18: The validation process, which is based on visual verification and collision detection techniques.

4.6 Export

As illustrated before, the planning tool is just a subcomponent of a complete system for pre-operative planning and intra-operative placement of the teleoperator arms. The intra-operative system developed by *Jörg Traub* [79] requires the reconstructed model of the patient as well as the planned data of the telemanipulator arms. Therefore, the planning tool features export functions to save the three-dimensional triangle mesh and every planned robot arm using the *Open Inventor* file format. These files can be re-used by the intra-operative placement tool, i.e. the virtual model is registered with the real patient and the tracked robot arms are aligned with the planned arms.
Additionally, all planned data can be saved at any stage of the planning process.

5 Medical Image Segmentation Techniques

There is nothing worse than a sharp image of a fuzzy concept.

Ansel Adams

As introduced in section 1.3.1.2, image segmentation is a very important field in medical imaging. Recent image acquisition modalities such as computed tomography and magnetic resonance imaging allow for non-invasive 'peering' into the patient's interior. However, having stacks of two-dimensional gray scaled images is rarely sufficient. Anatomical structures as well as other regions of interest need to be detected and extracted from these image stacks, preferably requiring as little manual interaction as possible in order to reduce time and human errors. Therefore, image segmentation algorithms were developed over the last two decades to delineate the contour of anatomical structures not only pre-operatively, but also intra-operatively [86]. They are used to quantify tissue volumes, diagnose diseases, localize pathology, study anatomical structures, plan treatment, and to partially correct the volume of functional imaging data [58].

Image segmentation techniques need to deal with this vast variety of applications along with their peculiarities such as the enormous size of data sets, the used image acquisition modalities, and arising imaging artifacts like noise, low contrast, partial volume effects, and motion. Generally, there is no ideal segmentation approach that leads to an optimal solution for all kinds of applications. Usually, single approaches tailored to a specific modality or particular application area provide better results than overall approaches. However, very often more than one segmentation algorithm is applied to a data set, resulting in a higher accuracy of defining boundaries for anatomical structures.

After clarifying the terminology and basics used for image segmentation (cf. section 5.1), this chapter gives an overview on various segmentation techniques that can be utilized to automatically or interactively receive the boundaries of a region of interest or anatomical structures. As proposed by *Lakare* [39], the presented techniques can be divided into three classes. Stochastic techniques (cf. section 5.2), being the traditional segmentation techniques, work on discrete voxels. For each voxel the decision is made whether it belongs to a certain region or not. Whereas stochastic techniques leave out structural information about the region to be segmented, structural techniques (cf. section 5.3) use exactly this information, i.e. the interdependent elements and definite organizational pattern of the region, to segment image data. Hybrid techniques (cf. section 5.4) make use of both stochastic and structural techniques.

5.1 Image Segmentation Foundation

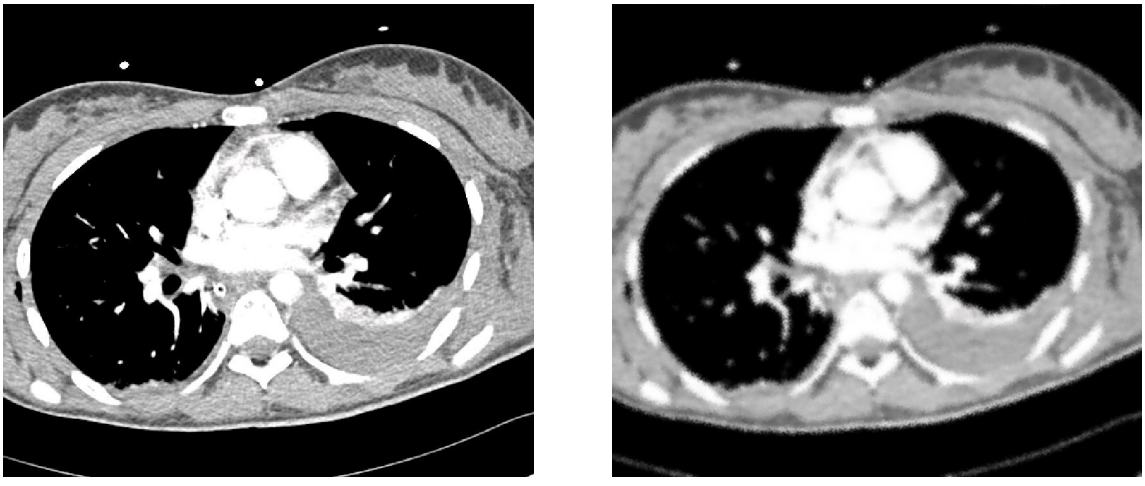
When talking about image segmentation, certain terms, definitions, and related issues need to be clarified, as presented in the following.

Basic Terminology Medical images are usually created by two- or three-dimensional measurements such as radiation absorption (X-ray, computed tomography), acoustic pressure (ultrasound), or *radio frequency* (*RF*) signal amplitude (magnetic resonance imaging). These measurements represent pixel or voxel intensities visualized by gray values and serve as a base for classifying the image. *Classical segmentation* partitions an image into non-overlapping connected regions by means of intensities. Mathematically, a segmentation satisfies

$$I = \bigcup_{k=1}^{K} S_k,$$

where $S_k \cap S_j = \emptyset$ for $k \neq j$. The entire image I is segmented into K *sets* S_k. In contrast, *pixel classification* does not require that segmented regions called *classes* are connected.
The process of *labeling* assigns an identifier, which may be an anatomical structure or the region of interest, to each segmented region S_k. Usually the labels are defined by physicians or their assistants.

Field Inhomogeneities The imaging artifact noise, which can be caused by spatial resolution, oversampling, patients' position, and other factors, can lead to *intensity inhomogeneities*. These inhomogeneities appear when regions that are part of the same tissue possess varying levels of intensity.



(a) Ideal image

(b) Blurred image as acquired by the image sampling modality

Figure 5.1: Partial volume effects as a result of the image sampling process.

In the case that two or more tissues contribute to the intensity of the same pixel or voxel *partial volume effects* occur, leading to blurring of intensities along the boundaries of the regions. Figure 5.1 shows a comparison of an ideal image and a blurred image as it might be acquired by the image sampling modality. In the blurred image the boundaries of image structures can hardly be detected. To overcome this effect, *hard segmentation* techniques, where each pixel or voxel is assigned to exactly one region, are replaced by so-called *soft segmentation* techniques, where each image element might be partitioned into multiple regions. Therefore, *membership functions* $m_k(j)$ are defined:

$$0 \leq m_k(j) \leq 1, \forall k, j$$

$$\sum_{k=1}^{K} m_k(j) = 1, \forall j,$$

where $j \in I$. The value of a membership function $m_k(j)$ describes, how much a region k contributes to a location j. Hence, if two or more sections are contributing to the same location j with a membership value greater than zero, they will be overlapping. In contrast, if $m_k(j)$ equals one for a each single region k, k is the only contributing region and, thus, the segmentation could be reduced to hard segmentation again. Another way of reducing the problem to hard segmentation is to assign a pixel or voxel to its region with the highest membership value.
Membership functions are usually accomplished by fuzzy clustering and classifier algorithms [34, 57], but also by statistical algorithms [42, 87] and estimates of partial volume fractions [16].

Manual interaction When approaching a computer-assisted segmentation task, very often the prior knowledge of human operators is incorporated. If the human operator joins in the segmentation, generally the accuracy can be improved. Consequently, the degree of automation is reduced, resulting in a time-consuming and laborious process. The degree of automation and hence interaction varies from entirely manual delineation of anatomical structures or regions of interest to the generation of partitions during thresholding (cf. section 5.2.1) or the planting of a seed point for region growing (cf. section 5.4.1). Nevertheless, whenever the human operator participates in the segmentation process, reliability might be compromised.

Validation Once a specific segmentation technique is applied, its result should be evaluated against other techniques and especially validation models. Therefore, so-called *truth models* such as manual segmentation outcomes or phantoms are created, which the results are compared to. On the one hand, manual segmentation by a skillful human operator leads to a truth model, which might be error-prone due to human errors, wherefore it cannot be considered to be a perfect. On the other hand, phantoms lack a realistic or sophisticated representation of anatomical structures. Phantoms can be divided into *physical phantoms*, which depict the image acquisition process accurately, but lack a realistic reflection of physiological properties of anatomical structures, and *computational phantoms*, which are more realistic and precise, but depict the image acquisition process by non-adequate, simplified models.

5.2 Stochastic Techniques

Stochastic segmentation techniques use statistical analysis only to segment an image. They work on pixel and voxel classification methods, but not on structural information.

5.2.1 Thresholding

Being one of the simplest segmentation techniques, thresholding divides an image by setting a binary partitioning of image intensities. The partition is achieved by a so-called *threshold*, which is a single intensity value that separates the image into regions. Depending on this threshold intensity, pixels or voxels with greater intensities are member of one region, whereas pixels or voxels with smaller intensities are member of another region. To visualize the occurrence of image intensities, two-dimensional *histograms* are used having the intensity on the x-axis and the number of pixels or voxels with this intensity on the y-axis, as shown on figure 4.9. Potential thresholds are set by looking at the valleys of a histogram, which is usually done manually, but can be automated as well [64]. If more than one threshold is set, it is referred to as *multi-thresholding*.

The advantages of using thresholding techniques are their simplicity, leading to a minimal implementation and computation complexity, and their effectiveness, when the regions to be segmented have contrasting intensities, but the intensity is homogeneous within the same region. Having many inhomogeneous intensities or noise causes their main disadvantage of not considering spatial image characteristics, leading to a difficult process of determining proper thresholds, especially from magnetic resonance imaging and ultrasound images. Assessing these aspects, thresholding techniques are very often used as an initial or final step in a series of image processing and segmentation procedures.

Sahoo et al. provide a survey on thresholding techniques [64], whereas *Sankur* and *Sezgin* claim to give a most comprehensive and recent survey [66]. The latter divide thresholding techniques into histogram shape-based, clustering-based, entropy-based, object attribute-based, spatial, and local methods.

5.2.2 Classification

Classification is the most commonly used segmentation technique in practical applications. It is a pattern recognition technique that separates a *feature space* of the image based on previously manually segmented training data [13, 68]. Mathematically, a feature space refers to an n-dimensional range space of image functions. For instance, features might be pixel or voxel intensities, gradients, and distances of pixels or voxels to the image boundary.

The training data, which the classification process is based on, is manually obtained and mapped to its extracted features. This mapping is actually used as a reference for the automatic classification of new data, wherefore classification techniques belong to the *supervised* image segmentation methods. It is essential for applying classifiers to select an appropriate feature space and classifier method. When taking the pixel or voxel intensity as a feature space, the classifier might be reduced to a generalized thresholding problem again.

Traditional classification methods can be grouped into non-parametric classifiers, which assume that the evaluation data does not own a certain statistical structure, and parametric classifiers, which assume an underlying statistical distribution.
Examples of non-parametric classifiers are the *nearest-neighbor* classifier, where each pixel or voxel is placed in the same region as a training datum having the same or the closest intensity, the *k-nearest-neighbor* (*kNN*) classifier, where a pixel or voxel is placed with respect to the majority of the k closest training data, and the *Parzen* classifier. The latter is applied to a local area defined by the Parzen window, where again the majority principle is used to determine the closest intensities within this window.
An example for the most commonly used parametric classifier is the *maximum likelihood* (*ML*) or *Bayes* classifier. The Bayesian model gives the *a posteriori* probability $P(c|X)$, whether a certain class c is present when a sample pixel or voxel X was observed. Every observed image pixel or voxel is assigned to the region with the highest a posteriori probability. It may be determined by following equation

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)},$$

where X is a feature vector of N gray values and c the region/class, a pixel or voxel belongs to. The probability $P(X|c)$ that a sample X is observed when a class c was present, might be computed from the multi-dimensional *Gaussian probability density function*

$$P(X|c) = (2\pi)^{-\frac{N}{2}} |\psi|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\left((X-\mu)^T \psi^{-1}(X-\mu)\right)\right], \tag{5.1}$$

where ψ is the covariance matrix of class c and μ is the mean vector of class c. The *a priori* probability $P(c)$ of a class c can be computed by using the distribution of different types of anatomical structures or tissues. [74]

Although classification methods feature computational efficiency by their non-iterative nature, a couple of disadvantages arise. For instance, classification methods require feature spaces to distinguish between classes and manual interaction to obtain the training data. If the same training data is applied to a large population, not all anatomical variances can be considered, leading to biased results. Furthermore, classification techniques are, similarly to thresholding, sensitive to noise and intensity inhomogeneities.

5.2.3 Clustering

In contrast to supervised classification methods, clustering techniques are *unsupervised*, since they do not use training data, meaning that the evaluation data needs to train itself. However, the way an image is segmented by clustering is similar to classifying it. Analog to classifiers, feature spaces may be pixel or voxel intensities, gradients, or neighborhood information. For clustering, first initial segmentation parameters are selected, which characterize the properties of each region/class. Next, the actual clustering algorithm is alternating between segmenting the classes depending on the characterized properties of the classes and re-characterizing the segmentation parameters.

Clustering algorithms can be divided into the two major groups hierarchical and non-hierarchical methods. Whereas representatives of hierarchical methods are the *K-means* or

ISODATA [19] and *fuzzy c-means* [13] algorithms, an example for a non-hierarchical method is the *expectation maximization* (*EM*) algorithm [87]. When using hierarchical methods, the input data is not partitioned into classes at once, but rather in successive steps, which agglomerate classes, i.e. fuse all image pixels or voxels into several final segmented classes, or divide one starting class into several segmented sub-classes. For non-hierarchical methods, a dedicated number of classes is known or assumed beforehand and each image pixel or voxel is assigned to precisely one class.

The *K-means* algorithm alternates between segmenting image pixels or voxels into classes with the closest mean feature values and calculating new mean feature values of classes. The iterative *K-means* algorithm stops, when for each class a minimal total distance of its mean to each pixel or voxel within the class is reached. The *fuzzy c-means* algorithm as a generalization to *K-means* applies fuzzy set theory to allow for soft segmentation. *Expectation maximization* is based on the Gaussian mixture model (cf. equation 5.1) and iterates between the computation of the *a posteriori* probabilities and the computation of maximum likelihood estimates of the Gaussian mixture model's means, covariances, and mixing coefficients.

As mentioned before, compared to classification techniques, the main advantage of clustering is that it does not require training data. Analog to classifiers, the clustering segmentation process is sensitive to noise and intensity inhomogeneities. Moreover, it is sensitive to initial parameters, especially when using the expectation maximization algorithm.

5.2.4 Markov Random Field Models

Markov random field (MRF) modeling is not a segmentation method itself, but incorporates spatial correlations into the segmentation process as *a priori* models, for instance into a *K-means* algorithm under a Bayesian *a priori* model [33]. Being a conditional probability model, Markov random fields are able to model the probability of a pixel's or voxel's membership to a certain class based on its neighborhood. In medical context, neighboring or nearby image pixels or voxels are mostly members of the same class. Physically, the essence of Markov random fields can be seen as the very low probability of the occurrence of an anatomical structure, which only a single pixel or voxel is member of.

The disadvantages of using Markov random fields are the requirement of time-consuming, computationally intensive algorithms and the determination of the right parameters, which are responsible for controlling the spatial correlations. If the spatial correlations are weighted too strongly, a smooth segmentation having too little structural details can occur. Nevertheless, many applications make use of Markov random fields to model segmentation classes, intensity inhomogeneities [33], and texture properties [61].

5.3 Structural Techniques

Structural segmentation techniques aim to detect structural properties of the image such as intersecting surfaces. By means of this structural information boundaries are formed to segment images.

5.3.1 Edge Detection

Edge detection techniques try to localize the pixels or voxels of an image, which correspond to the edges or surfaces of anatomical structures. Edges mainly lie between two regions with different intensities. A typical edge detection algorithm first detects local edges by applying numerical differentiation and second organizes these local edges in such a way that boundaries are formed, which define regions. In detail, intensity changes are localized by derivatives of different types and scales. However, before differentiation is actually applied, it needs to be regularized by a regularizing filter operation. [78]

Classical two-dimensional edge detection techniques include *Robert's Cross*, *Sobel*, *Prewitt*, and *Canny*. These techniques have been extended to the three-dimensional space, for instance *Robert's Cross* by *Liu* [45] and *Sobel* by *Zucker* and *Hummel* [90].

Edge detection techniques work well on images with good contrast between single regions. However, if many edges are detected, it will be difficult to identify the edges needed to separate certain regions. Additionally, edge detection techniques are sensitive to noise. Usually they are used in a series of segmentation techniques.

5.3.2 Morphology

A massive drive of mathematical morphology was performed by *Serra* [70]. Morphological image analysis is based on point sets X and morphological operations on them. An operation is a transformation Ψ of a set into another, followed by a measure μ. The outcome of $\mu(\Psi(X))$ is a number, which might be a weight, volume, or surface area. Having an image, a so-called *structuring element* B is applied to its pixels or voxels, where a structuring element B_x has a certain shape centered in point x. These structuring elements actually define the transformations.

Based on this theory, the two most fundamental transformations of *erosion* and *dilation* can be illustrated:

$$Y_{eroded} = X \ominus \check{B} = \{x : B_x \subset X\}$$

defines an erosion of a point set X by B, where the eroded set Y_{eroded} is the locus of center points x such that B_x is included in X. In other words, an erosion shrinks a region.

$$Y_{dilated} = X \oplus \check{B} = \{x : B_x \cap X \neq \emptyset\}$$

defines a dilation of a point set X by B, such that the dilated set $Y_{dilated}$ is the locus of center points x of the B_x, which hit the set X. In contrast to erosions, a dilation expands a region.

Figure 5.2 shows these two basic transformations, assuming that a point set X consists of only two phases, namely black and white. It also clarifies the terms of *closing* and *opening*. A series of dilations followed by erosions is called closing and fills holes in image regions. Contrary, a series of erosions followed by dilations is termed opening and may introduce holes in regions.

Morphology can change the structure of regions enormously, wherefore a risk of loosing important data is introduced. When accuracy is desired during segmentation, morphological algorithms should be avoided. As many other segmentation techniques, they are usually an intermediate step in a series of image processing and segmentation procedures.

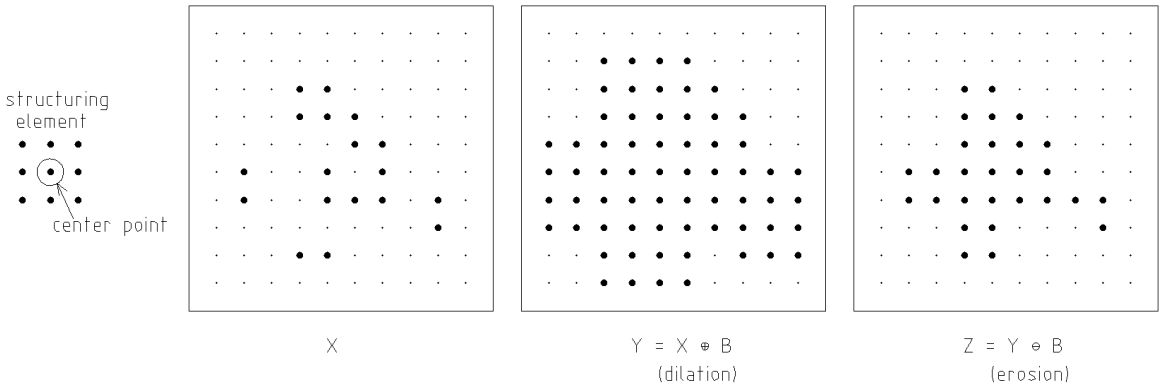


Figure 5.2: Using morphological techniques to perform *closing* on an image region. First, a *dilation* transformation using the *structuring element* B on the left is applied on a black and white point set X to expand the black region. On its result Y a second transformation of the *erosion* type is applied to shrink the black region again. The result of this closing operation is that all holes in the black region are filled.

5.3.3 Deformable Models

Deformable models, mainly stemming from the appliance of elasticity theory in the physical context, can be seen as elastic bodies, which respond to applied forces and constraints by nature [51]. Within the imaging domain, they can be defined as curves or surfaces that move under the influence of *internal forces*, which try to preserve their smooth shape, and *external forces*, which try to inflict the image data on them, so they are driven towards a desired feature of interest such as the boundary of a region. To delineate a boundary, first a closed curve or surface must be placed in its neighborhood manually. Next, the contour passes through an iterative relaxation process, wherein the internal and external forces are calculated.

Different views of deformable models lead to the three sub-categories of parametric, dynamic, and probabilistic deformable models. However, the most popular two-dimensional form of parametric deformable models is that of *active contour models*, also referred to as *snakes* [36], which are briefly outlined in the following.

The curve of a snake is parameterized by $\nu(s) = (x(s), y(s))$, where $(x, y) \in \mathbb{R}^2$ is the image plane, x and y are coordinate functions, and $s \in [0, 1]$ is the parametric domain. The snake's contour is dictated by a *minimum* of the energy functional

$$E^*_{snake} = \int_0^1 E_{snake}(\nu(s))\mathrm{d}s = \int_0^1 E_{int}(\nu(s)) + E_{image}(\nu(s)) + E_{con}(\nu(s))\mathrm{d}s,$$

where E_{int} refers to the snake's internal spline energy and the image forces E_{image} added by the constraint forces E_{con} form its external energy. The internal spline energy can be described as

$$E_{int} = \left(\alpha(s)\left|\frac{\mathrm{d}\nu(s)}{\mathrm{d}s}\right|^2 + \beta(s)\left|\frac{\mathrm{d}^2\nu(s)}{\mathrm{d}s^2}\right|^2\right),$$

where the first-order term controlled by $\alpha(s)$ specifies the snake's elasticity or tension and the second-order term controlled by $\beta(s)$ defines the snake's stiffness or rigidity. For instance, if

$\beta(s)$ is set to zero at a point, the snake develops a corner at this point. Whereas the constraint forces E_{con} are caused by the user or other higher level processes, the image forces E_{image} can be formalized as

$$E_{image} = \omega_{line}E_{line} + \omega_{edge}E_{edge} + \omega_{term}E_{term}.$$

Therewith, a snake gets attracted to the image features lines (E_{line}), edges (E_{edge}), and terminations (E_{term}). By adjusting their corresponding weights ω_{line}, ω_{edge}, and ω_{term} the snake behavior can be altered in various ways.

Advantages of deformable models are their robustness against noise and spurious edges, resulting from internal forces, and their capability to generate a closed curve or surface directly from the image by calculating external forces. Nevertheless, deformable models lack a full automation, since an initial model needs to be placed manually and appropriate parameters need to be chosen. Additionally, their performance depends on the quality of the initial model placement.

5.4 Hybrid Techniques

Hybrid techniques are working on both pixel or voxel classification methods and structural information. Therefore, they are termed hybrid and cannot be classified as purely structural or stochastic.

5.4.1 Region Growing

Region growing techniques are able to segment an image relying on a previously defined connectivity criterion such as image intensities, textures, spatial features, or edges. A region growing algorithm is started by manually planting a *seed point*. The algorithm extracts all pixels or voxels, which are connected to this point and fulfill the criterion. The criterion is usually an intensity range, in which the connected neighboring pixels must lie.

(a) Initially, a seed point is planted.

(b) All pixels, which are connected to the seed point and have a similar intensity, are selected.

(c) The selected region is growing until a certain criterion is not fulfilled any more.

Figure 5.3: Example for segmenting a kidney using two-dimensional region growing.

Region growing is effective in segmenting simple and small structures. A disadvantage of basic region growing algorithms is, besides their requirement for manual interaction, that

they are sensitive to noise. Thus, holes within a segmented region might be introduced or two separate regions might be connected due to partial volume effects. As for thresholding or morphology, region growing is generally used in a chain of image processing and segmentation steps. Recently, approaches were made to segment whole volume sets automatically, called *unseeded region growing* [44]. Another automatic technique is titled *split and merge* [49]. Being a top-down approach, it first considers the whole image as one region and later divides it into sub-regions using quadtrees or octrees (cf. section 6.3.1.1), respectively.

5.4.2 Artificial Neural Networks

Biologically, neural networks are seen as an adequate model of the human brain and its functionality. In the mathematical context, this functionality is represented by functions, which are created by combining elementary computational units, and learnt by examining a set of examples. In an *artificial neural network*, these units are forming the nodes of the network, which are connected to each other via input and output links. Each unit consists of an input function, which combines and weights the input links, and a generally non-linear activation function, which computes the output. The adaption of correct weights for the input links is considered to be the learning process of artificial neural networks [63].

Advantages of neural networks are their ability of inductive learning from examples, a massive parallelism, fault tolerance, and graceful degradation. Segmentation techniques try to incorporate these advantages to generalize knowledge, reject image noise, be fault tolerant, and get an optimum seeking behavior. Artificial neural networks are used as supervised classifiers [27, 32], where training data is used to specify the weights of input links, but also in unsupervised clustering techniques [13, 60] and for deformable models [85].

5.4.3 Atlas-Guidance

Atlas-guided segmentation approaches are based on a standard atlas or template, which is mapped to the acquired images. A manually generated atlas contains all information about the anatomical structures, which need to be segmented, for instance the human brain or cardiac region. Similarly to classifiers, atlas-guided techniques are based on manually segmented data, but they are situated in the spatial image domain and not in a feature space.

The mapping of a segmented atlas image to an acquired image is usually performed by registration, wherefore a transformation needs to be found. This mapping process is also called *atlas warping* and achieved by linear transformations. Chiefly applied for segmenting magnetic resonance images, atlas-guidance has been used mainly for segmenting brain anatomy [17], but lately also for cardiac structures [46].

Atlas-guidance's advantage is that not only an appropriate segmentation can be transferred to the acquired images, but also the segmentation's labels. Moreover, it supports morphometric methods that involve the measurement of structures' shapes. However, anatomical variability complicates an accurate segmentation process based on linear transformations. Hence, additional non-linear transformations were added to linear transformations [17] to become better segmentation results. Even so, the segmentation of complex anatomical structures remains difficult and can only be improved by so-called *deformable probabilistic*

atlases [77], which depend on a time-consuming interactive process. Atlas-guided segmentation techniques work best for structures, which are stable over the population of study [58].

6 Implementation Details

> I want to know God's thoughts; the rest are details.
>
> Albert Einstein

This chapter covers a selection of algorithms, computations, modules, and classes, which were used to implement the port placement planning system. It does not cover the tools, which were already available by amira, for instance the image segmentation utilities.

6.1 Convenience Utilities

Despite the enormous functionality of *Open Inventor* and amira, certain additional methods and classes are required by the planning module to achieve specific computations or conversions just using one function call. Therefore, the amira package `common` was developed, including the classes `Utility` and `TCLInterpreter`. The `Utility` class provides a collection of commonly used static methods for data type conversion and three-dimensional computations, whereas `TCLInterpreter` is a wrapper class for accessing the global instance of `HxInterpreter`.

6.1.1 Geometric Computations

A few geometrical three-dimensional computations used in the planning module are implemented as static methods in the `Utility` class. They all make use of *Open Inventor*'s base classes `SbVec3f`, `SbLine`, and `SbPlane`, representing a vector, line, and plane in three-dimensional space.

`getCentroid` The `getCentroid` method computes the geometric centroid $\vec{b}$ of a tetrahedron defined by four vectors $\vec{c}$, $\vec{d}$, $\vec{e}$, and $\vec{f}$. This is simply done by following equation:

$$\vec{b} = \left(\frac{\vec{c} + \vec{d} + \vec{e} + \vec{f}}{4} \right) = \begin{pmatrix} (c_x + d_x + e_x + f_x)/4 \\ (c_y + d_y + e_y + f_y)/4 \\ (c_z + d_z + e_z + f_z)/4 \end{pmatrix}$$

`getVectorAngle` The `getVectorAngle` methods returns the $angle$ between two vectors $\vec{u}$ and $\vec{v}$ in radian measure:

$$angle = \arccos \left(\frac{\vec{u} \cdot \vec{v}}{|\vec{u}||\vec{v}|} \right)$$

`rotate` The `rotate` function computes the rotation of a point $\vec{p}$ about an arbitrary axis, which is defined by a point $\vec{d}$ and a direction vector $\vec{e}$, by an arbitrary angle δ, and returns the coordinates of the rotated point. Therefore, a transformation matrix having homogeneous coordinates can be composed.
Following steps are illustrating how the rotation can be performed in an understandable, but time-consuming way.[1]
Firstly, the axis point $\vec{d}$ is translated to the origin using the translation matrix T:

$$T = \begin{pmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.1}$$

Secondly, the rotation's direction vector $\vec{e}$ is rotated about the x-axis, so it lies in the x-z plane:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6.2}$$

where $\cos\alpha = e_z/u$, $\sin\alpha = e_y/u$, and $u = \sqrt{e_y^2 + e_z^2}$.
Thirdly, $\vec{e}$ is rotated about the y-axis, so it is aligned with the z-axis:

$$R_y = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6.3}$$

where $\cos\beta = u/v$, $\sin\beta = e_x/v$, and $v = \sqrt{u^2 + e_x^2}$.
Now the actual rotation of the point $\vec{p}$ can be performed about the z-axis by δ:

$$R_z = \begin{pmatrix} \cos\delta & \sin\delta & 0 & 0 \\ -\sin\delta & \cos\delta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.4}$$

The final three matrices, which are necessary for the composition, are the inverses of the matrices provided in equations 6.3, 6.2, and 6.1. They are all easy to determine, since the inverse of a rotation matrix is equivalent to its transpose and the inverse of a translation matrix is the translation matrix with the opposite signs on each of the translation components.
If all matrices of equations 6.1 to 6.4 and the inverses of 6.3, 6.2, and 6.1 are put together, following equation will give the arbitrary rotation of a point $\vec{p}$:

$$\vec{p}' = T^{-1} R_x^{-1} R_y^{-1} R_z R_y R_x T \vec{p} \tag{6.5}$$

[1] The computations are all based on the assumption that $\vec{e}$ is of unit length, i.e. $\sqrt{e_x^2 + e_y^2 + e_z^2} = 1$. If not, it can be normalized easily: $\vec{e}_{unit} = \frac{\vec{e}}{|\vec{e}|}$

However, all rotation matrices can still be simplified by composing them into one single rotation matrix R:

$$R = \begin{pmatrix} e_x^2 + \cos\delta(1-e_x^2) & e_x e_y(1-\cos\delta) - e_z\sin\delta & e_z e_x(1-\cos\delta) + e_y\sin\delta & 0 \\ e_x e_y(1-\cos\delta) + e_z\sin\delta & e_y^2 + \cos\delta(1-e_y^2) & e_y e_z(1-\cos\delta) - e_x\sin\delta & 0 \\ e_z e_x(1-\cos\delta) - e_y\sin\delta & e_y e_z(1-\cos\delta) + e_x\sin\delta & e_z^2 + \cos\delta(1-e_z^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.6}$$

So the rotation of a point $\vec{p}$ about an arbitrary axis by an arbitrary angle can be described as

$$\vec{p}' = T^{-1}RT\vec{p}. \tag{6.7}$$

Actually, a more elegant way for composing an arbitrary rotation matrix R is to use *Rodrigues' rotation formula* [12]:

$$\vec{x}' = \vec{x} + (\sin\delta)\vec{e}\times\vec{x} + (1-\cos\delta)\vec{e}\times(\vec{e}\times\vec{x}) \tag{6.8}$$

The cross product of $\vec{e}$ and $\vec{x}$ can be described defining a 'cross product matrix' E, so that $E\vec{x} = \vec{e}\times\vec{x}$:

$$E = \begin{pmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix}$$

This cross product matrix E is substituted into Rodrigues' rotation formula 6.8:

$$\vec{x}' = \vec{x} + (\sin\delta)E\vec{x} + (1-\cos\delta)E^2\vec{x}$$

If $\vec{x}$ is factored out, the rotation matrix is obtained [50]:

$$R_{Rodrigues} = I + (\sin\delta)E + (1-\cos\delta)E^2, \tag{6.9}$$

where I is the identity matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Expanding the rotation matrix $R_{Rodrigues}$ and expressing it in homogeneous coordinates gives the rotation matrix R already defined in equation 6.6. Again, the complete transformation can be performed using equation 6.7.

6.1.2 Tcl Access

Not all classes of amira are provided by the collection of C++ header files obtained through amiraDev. Hence, certain objects in amira's object pool cannot be manipulated directly, but only via the Tcl command interface, which may be either the console window or `theInterpreter`, a global instance of `HxInterpreter`. Since the console window does not excel in user-friendliness, commands are usually received through ports that have a graphical user interface. These ports can be embedded into the planning module, which is also capable of directly accessing the `eval` method of `HxInterpreter`. Therefore, the argument that is passed to `eval` is a long character string, which needs to be constructed over

and over. Thus, the wrapper class `TCLInterpreter`, which can be accessed from any other class, was introduced to enable simple access to `theInterpreter`.

Because only one instance of the class is needed throughout execution of the planning module, wherefore no additional memory space is wasted, `TCLInterpreter` can be only instantiated once. All classes referring to it only get a reference to this single instance, also called *singleton*. Using the singleton design pattern, `TCLInterpreter` features an avoidance of direct creation of objects of the class by setting the constructor private, the creation of an instance of the class as a class variable (`obj`), and public access to this single instance via two class methods, one each for single threaded (`Instance`) and multi threaded applications (`MultiThreadedInstance`). The single instance of the class is automatically generated by calling the private constructor, when it is requested for the first time. To determine the first or a subsequent request, i.e. whether the class is already instantiated, a boolean variable (`instanceFlag`) is set.

A class diagram showing the attributes and operations implementing the singleton design pattern as well as operations commonly used by the planning module can be seen on figure 6.1.

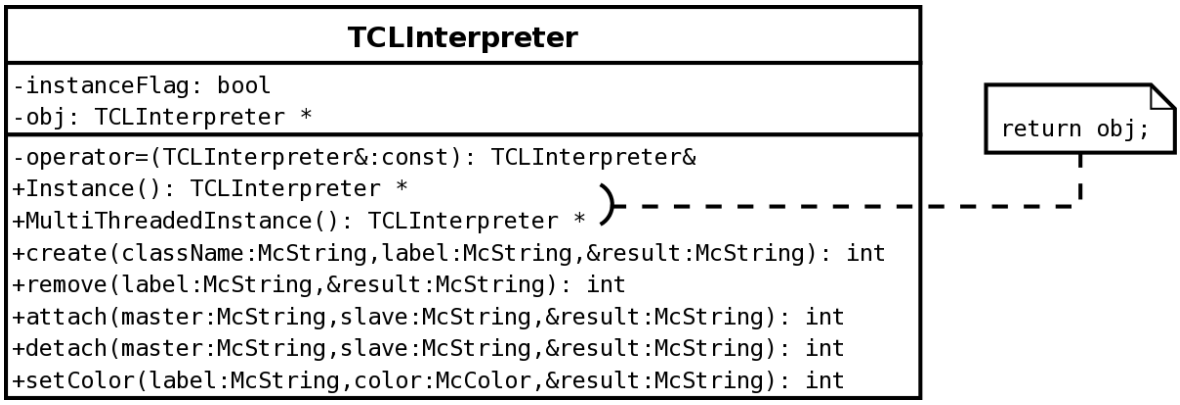


Figure 6.1: The attributes and operations of the `TCLInterpreter` wrapper class, which are important for the singleton design pattern and the planning module.

6.2 Planning Module

As highlighted before, the main planning functionality is achieved by an amira module called `PortPlacementView`, which is derived from the original `HxDisplaySurface` module and, thus, inherits all of its features. Moreover, many additional amira ports have been implemented to enable the users to perform the planning process in an intuitive way. They only need to deal with a single amira module that encapsulates three `HxLandmarkSet` data objects and three `HxDisplayLandmarks` display modules, one per definition of the region of interest and the ports for endoscope or instrument arms, respectively. Hence, also all features of landmarks are inherited.
The instances of all landmark objects are automatically created, when the module is attached to a `HxSurface`.

Whenever any values of the planning module's ports change, the `compute` method of `PortPlacementView` is called. This method, which is the core function of every amira module, traverses all necessary changes, for instance, the context sensitive menus are updated, the scene graph is redrawn, the camera pose is transformed, file dialogues are opened, collisions are computed, color and transparency of objects are changed, or objects are shown or hidden.

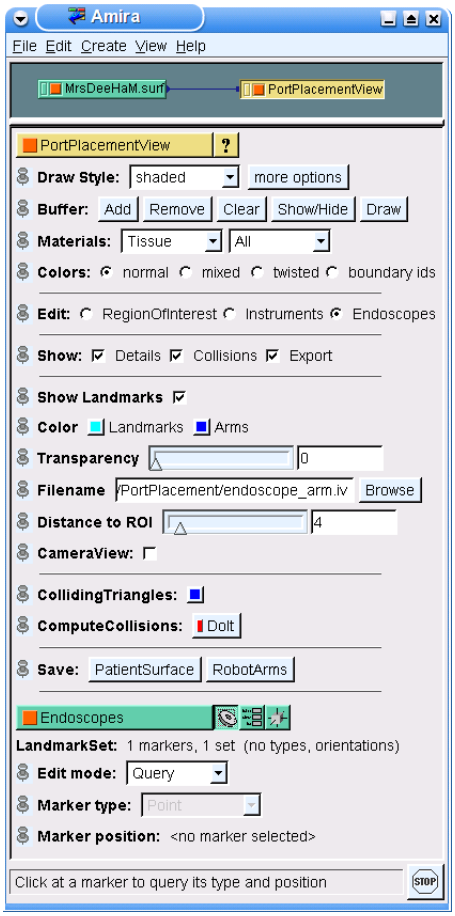


Figure 6.2: The user interface of the planning module.

In the following, the functionality of `PortPlacementView`'s graphical user interface is described. The users can select via a toggle list, what kind of landmark set they want to edit. Depending on the selection, the correlated `HxDisplayLandmarks` module appears in order to query, add, remove, move, or transform landmarks. For each added landmark representing an instrument or endoscope arm a corresponding instance of `RobotArm` is created and added to the scene graph. Additionally, if a more detailed list of selections is preferred, a context sensitive menu can be extended to display details on landmark sets or instrument arms. Landmarks can be shown or hidden and their color can be changed. Furthermore, if available in the current context, the color, transparency, and *Open Inventor* file of the telemanipulator arms can be changed or their distance to the region of interest's center adjusted.

The planning data in its current state can always be stored. Therefore, either all objects in amira's object pool can be saved as a so-called network file or only specific objects such as the triangle mesh. Activating a special toggle in `PortPlacementView` it is possible to export the patient's model and the telemanipulator arms in their current pose as *Open Inventor* files. When exporting, the dimensions of all models are already aligned properly, so no further scaling needs to be executed.

amira 's viewer window is able to switch between a parallel and a perspective presentation of the scene to the users. Actually, the view is implemented as a camera's view on the scene. By navigating inside the viewer window the camera's pose can be transformed in space.
In the context of planning the incision port for an endoscope arm the detailed menu is extended by a toggle button. When the users activate this button, they are able to switch to the virtual endoscopic view, meaning that the camera is positioned exactly on top of the endoscope arm and aligned with its direction. Therewith, the presented virtual endoscopic view is identical to the one surgeons have during a telemanipulator based operation, which they are more used to, so they are capable of judging possible risks in a more intuitive way.

Additionally, the planning module offers a toggle button to expand the collision detection menu, which can be used for activating triangle collision tests, as described in section 6.3.1. When activating the test by pushing a button, all collisions between the telemanipulator arms and the patient's model are calculated. If all collisions are computed, a window showing all detected collisions pops up and the colliding triangles are visualized in a certain color that, beforehand, can be chosen by the users. All colliding triangles are forming a new `HxSurface`, which is included, along with a `HxDisplaySurface` module, into `amira`'s object pool, so they can be shown, hidden, and saved separately.

6.3 Collisions

Dealing with collisions is addressed twice in the planning module: Once, when the instrument arms are placed and visualized without collisions, and another time, when the three-dimensional models of the telemanipulator arms and the patient are checked for intersections. Once more, it is important to become aware of the difference, that in one case collisions are avoided and in the other case they are detected.

6.3.1 Detecting Intersections of Geometric Models

Finding the intersections between geometric models can be a very time-consuming process, wherefore efficient algorithms need to be applied. The collision detection technique used in the planning module is a so-called *Open Inventor* action. If a collision detection request is detected in the `compute` method of the planning module, this action will be applied to the branch of the scene graph that includes the robot arms' and the patient's model. The action only traverses the nodes of this branch of the scene graph, so unnecessary checks can be avoided. The traversal works as follows: Firstly, the bounding boxes of all related shapes within the nodes are compared to each other, i.e. the `SoTriSurface`'s bounding box is compared to the bounding boxes of all telemanipulator arms and the telemanipulator arms are compared to each other. If an intersection between two shapes exist, the collision detection mechanism gets refined. In a second step both shapes are compared to each other regarding their containing triangles. Whenever the comparison of two triangles is finished, a callback method is responsible for continuing the tests until every triangle pair is checked. However, only collisions between objects selected to be shown are detected. For instance, most commonly the left lung and the skin are hidden for collision tests.

6.3.1.1 Finding Neighboring Triangles

The time-consuming part of those collision detection tests is the second step, since typical patient's models contain hundreds of thousands of triangles that all need to be compared to the telemanipulator arms' triangles. Therefore, an optimization technique based on the concept of *octrees* is used that reduces the number of comparisons by just comparing neighboring triangles. An octree data structure is sorting all triangles into a hierarchal tree, whose nodes all have none or eight children. It is generated in following way. Firstly, a cube is chosen, which covers the whole volume of triangles that need to be tested. Next, this cube is

divided into sub-cubes having half its side length. This procedure is repeated for every cube that still contains triangles or parts of them. However, a minimum cube's side length is set to get a maximum tree depth, so memory space can be saved. For each cube, it is determined whether it entirely lies inside or outside the triangle or only partially, i.e. it intersects with at least one triangle's edge.
Traversing the emerging octree, only triangles, which are entirely inside the same cube, need to be compared.

6.3.1.2 Triangle-Triangle Intersection Tests

Still, all neighboring triangles must be checked for intersections. Therefore, methods to test the intersection of two triangles in three dimensions are depicted in *Devillers'* and *Guigue*'s research report [21] as well as *Tomas Möller*'s paper [54].
Their general approach is based on the definition of half-spaces by the two planes π_1 and

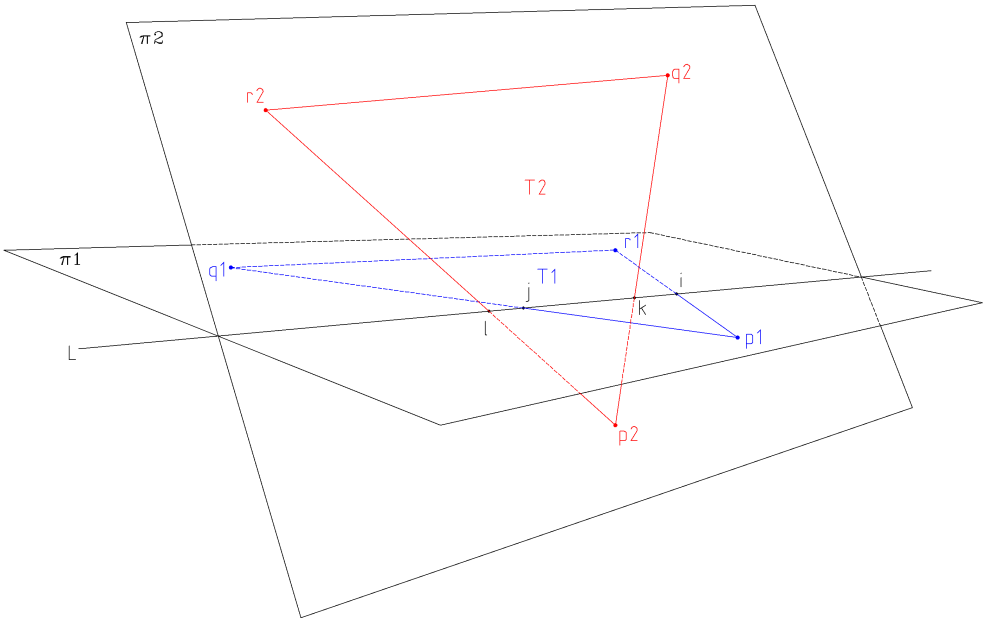


Figure 6.3: Triangles T_1 and T_2 and their corresponding planes π_1 and π_2. L is the intersection line of the planes, whereas i, j, k, and l are the intersection points of L with the edges p_1r_1, p_1q_1, p_2q_2, and p_2r_2, respectively.

π_2, which the triangles T_1 and T_2 lie in, as illustrated in figure 6.3. Firstly, it is determined, whether one triangle T_1 lies entirely inside one of the two half-spaces defined by the other triangle's plane π_2. If it does not, it intersects π_2. In that case, the second step of the approach checks, whether T_1 also intersects with the other triangle T_2. Therefore, after determining the intersection line L of the two planes, it is computed, whether the first triangle's intersection with L overlaps the second triangle's intersection with L. In the case of having two coplanar planes, it is determined whether both triangles are planar, so a two-dimensional intersection test can be performed.

The Basics The crucial idea is to define a determinant for the two-dimensional and the three-dimensional intersection test, respectively. Having three two-dimensional points $a = (a_x, a_y)$, $b = (b_x, b_y)$, and $c = (c_x, c_y)$, the determinant is defined as

$$D_{2D} := [a, b, c] = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix}.$$

Alternatively, having four three-dimensional points $a = (a_x, a_y, a_z)$, $b = (b_x, b_y, b_z)$, $c = (c_x, c_y, c_z)$, and $d = (d_x, d_y, d_z)$, it is defined as

$$D_{3D} := [a, b, c, d] = \begin{vmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{vmatrix} = \begin{vmatrix} a_x - d_x & a_y - d_y & a_z - d_z \\ b_x - d_x & b_y - d_y & b_z - d_z \\ c_x - d_x & c_y - d_y & c_z - d_z \end{vmatrix}.$$

For the two-dimensional case, depending on whether D_{2D} is positive, negative, or equals zero, the point c is on the left of, on the right of, or on the line from a towards b. For the three-dimensional case, if D_{3D} is positive, negative, or zero, the point d will be above, beneath, or on the plane specified by the right-hand rule, taking a as center, b as first direction vector, and c as second direction vector. Equivalently, if D_{3D} is positive or negative, a screw aligned with the ray $\overrightarrow{ab}$ will turn in the same or opposite direction of $\overrightarrow{cd}$. However, computing the determinant actually gives the area (volume) of a triangle (tetrahedron), so an area (volume) of zero gives collinear (coplanar) points.

The Three-Dimensional Approach In detail, the collision of two triangles T_1, which is defined by the points p_1, q_1, and r_1, and T_2, defined by p_2, q_2, and r_2, will be tested for. Therefore, being the first step of the approach, it is determined, whether T_1 intersects with π_2, i.e. the algebraic signs of the three determinants $[p_2, q_2, r_2, p_1]$, $[p_2, q_2, r_2, q_1]$, and $[p_2, q_2, r_2, r_1]$ are computed. Three cases can happen:

1. All three determinants are unequal zero and their algebraic signs are all equal. Then T_1 entirely lies inside one of the two half-spaces described by π_2, meaning that no intersections occur.
2. All three determinants equal zero, so the two triangles T_1 and T_2 are coplanar. Therefore, a two-dimensional intersection test is executed (see below) after projecting the triangles into a convenient plane. The convenient plane is either the x-y, x-z, or y-z plane, depending on whether the z, y, or x value of the triangles' normal is highest, so that the projection area is maximized.
3. The determinants' algebraic signs are unequal, meaning that T_1 definitely intersects with π_2. Thus, the next step of the approach is executed.

Now, in an intermediate step the points of both triangles are reordered by circular permutations and swap operations in such a way that p_1 is the only point of triangle T_1 that lies in the positive half-space described by π_2. The analog applies for p_2. One possible result of all reordering operations can be seen on figure 6.3.

Finally, the three-dimensional triangle-triangle intersection test takes place. Therefore, it looks at the intersection points i, j, k, and l of L with the edges p_1r_1, p_1q_1, p_2q_2, and p_2r_2. If the intervals $[i, j]$ and $[k, l]$ overlap, their corresponding triangles T_1 and T_2 intersect. However, the intermediate reordering step arranged all points in such a way, that $[i, j]$, $[k, l]$, and L are all having the same direction. Hence, if $i \leq l$ and $k \leq j$, the intervals will overlap. Applying these inequalities to the considerations on three-dimensional determinants, an overlap and, hence, an intersection will occur, if

$$[p_1, r_1, r_2, p_2] \leq 0 \wedge [p_1, q_1, p_2, q_2] \leq 0.$$

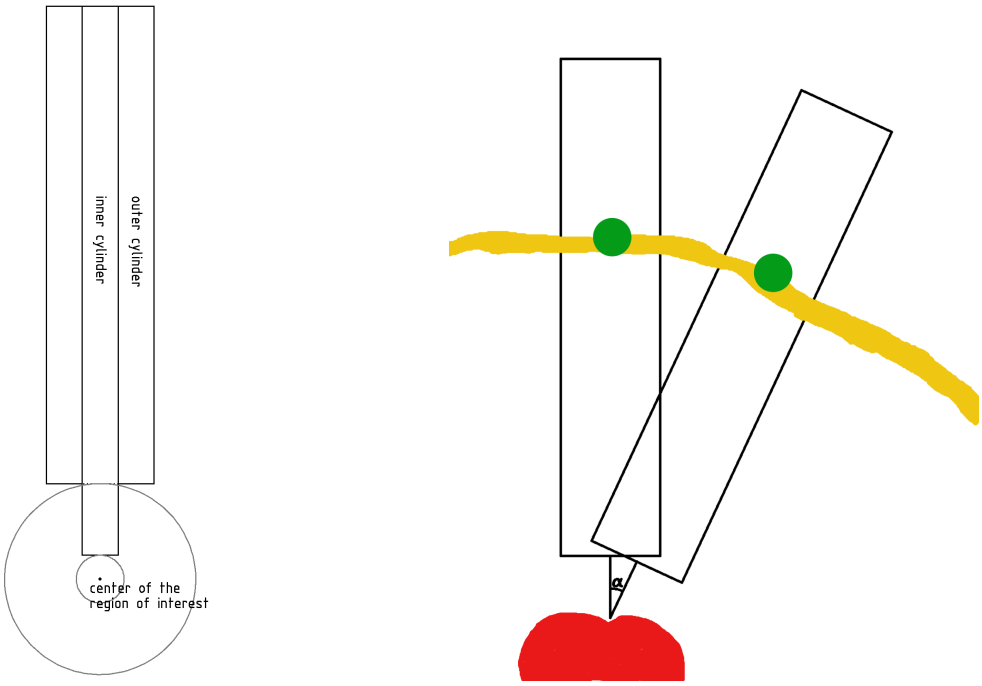
The Two-Dimensional Approach Testing for intersections of two coplanar, i.e. two-dimensional triangles is again based on determinants. The whole two-dimensional approach is not described in detail here, but can be found in *Devillers*' and *Guigue*'s report [21]. Nevertheless, its steps are briefly outlined.
Since the orientation of both triangles is important for determinant operations, in a first step two points of each triangle might be swapped to obtain a counterclockwise orientation of all points, so triangle T_1 is described by $p_1q_1r_1$ and T_2 by $p_2q_2r_2$. Next, it is checked whether p_1 lies on T_2's edges, interior, or exterior by applying $[p_2, q_2, p_1]$, $[q_2, r_2, p_1]$, and $[r_2, p_2, p_1]$, so T_2 divides the plane into seven regions, where the triangle itself forms a region that is surrounded by six additional regions. In the case that p_1 is inside or on T_2, a collision has been detected and it can be stopped now. In any other case, the points of T_2 might be rotated, so p_1 is positioned in one of two regions, representing two general configurations. The third and last step executes for each configuration additional orientation tests based on the determinants of three points. Anyhow, at most five of these orientation tests are needed to determine, whether triangle T_1 intersects with T_2.

6.3.2 Collision Freedom of Instrument Arms

A general problem after placing landmarks arises during the visualization of more than one robot arms, since they must not collide with each other. If the arms are simply aligned with the line between a landmark and the region of interest's center and placed too close to each other, a collision will occur. As figure 6.4(b) clarifies, certain transformations of the arms need to be executed, so the arms can be kept in the required distance to the region of interest's center on the surface of the heart.

Only collisions between instrument arms are implemented, since the endoscope arm usually has enough distance to the region of interest and, thus, to any other arms. Therefore, the very common case of avoiding collisions between two instrument arms is considered in section 6.3.2.1. Furthermore, for preventing collisions of these arms with an additional assistance arm, computations that help to avoid collisions between three instrument arms are depicted in section 6.3.2.2.

Actually, during following computations an instrument arm is considered to consist of two cylinders only. The outer cylinder representing the arm body is wider, but also further away from the region of interest's center than the inner cylinder, which is containing the tool tips. Figure 6.4(a) shows a cut through the arm's axis it describes.
If comparing instrument arms and their corresponding cylinders, they always need to have



(a) A simplification of a robot arm consisting of two cylinders, one inner and one outer cylinder. A certain distance to the region of interest's center is kept.

(b) If two or more arms need to be placed they cannot be aligned with the lines between their ports and the region of interest's center. A rotation and translation of the arms needs to be executed.

Figure 6.4: Several considerations need to be made for robot arms, which are represented by cylinders, in order to avoid collisions.

the same distance from the region of interest's center, i.e. their inner and outer cylinders each need to exactly touch a sphere around the region of interest's center. Therefore, always two computations are executed, one for each cylinder.

6.3.2.1 Two Arms

The idea of a collision-free visualization of two instrument arms is to place the arms on a direct line between the ports and the region of interest's center, to check for collisions, and to resolve these collisions by determining two transformations, each consisting of a rotation around an axis defined by the port of an arm and a translation along the axis the arm describes after the rotation. Therefore, the following process is performed.
In a first step, a convenient plane is defined by the two landmarks and the center point of the region of interest, so all computations can be done in two dimensions. In a second step, the arms are aligned with the line between their corresponding landmark and the region of interest's center, so the convenient plane is exactly bisecting both arms that are represented by their inner and outer cylinders, which are quadrangles in two dimensions.
In the following, all computations for the inner and outer cylinder are analog, so, henceforth,

they are only illustrated for one cylinder. The next step is to check whether the arms in their current position intersect each other. If they do intersect, a rotation will be performed around the axis, which is defined as the convenient plane's normal through the cylinder's landmark. Therefore, the rotation angle needs to be computed. Additionally, the rotated cylinder is translated along the axis it is describing, wherefore the translation distance is determined as well.

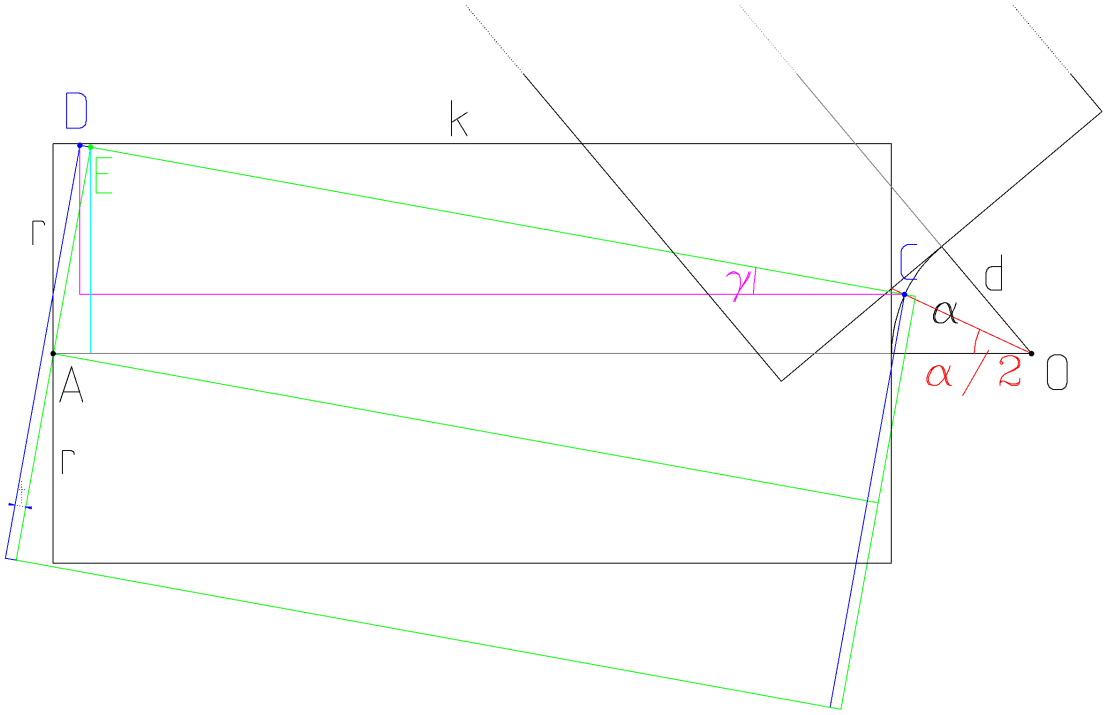


Figure 6.5: A geometrical clarification of how collisions between two cylinders are detected. The rotation angle γ and the translation t for a cylinder having radius r and length k can be found by defining auxiliary points C, D, and E.

Figure 6.5 clarifies geometrically, how cylinder collisions are detected and the rotation angle and translation distance of one cylinder are found. If the figure is flipped, all computations can be applied to the other arm's cylinder. The cylinders' radius is r, their distance from the region of interest's center $O = (0, 0)$ is d, and the angle between them is α, previously acquired by `getVectorAngle`. The length of the lower cylinder is k, its point referring to the robot arm's landmark is $A = (a_x, a_y)$.
First, a check is performed whether the lower cylinder intersects with the upper cylinder. Therefore, following equation is helpful:

$$a = r - (\tan\frac{\alpha}{2})d$$

If $a \geq 0$, an intersection occurs and it can be proceeded by calculating the rotation angle γ and the translation t of the arm to align its upper right corner with $C = (c_x, c_y) = (-(\cos\frac{\alpha}{2})d, (\sin\frac{\alpha}{2})d)$ in such a way that C is still having the same distance d from O.
The points D and E are defined as follows:

$$D = (d_x, d_y) = (c_x - (\cos\gamma)k, c_y + (\sin\gamma)k)$$

$$E = (e_x, e_y) = (a_x + (\sin\gamma)r, (\cos\gamma)r)$$

γ can now be found by resolving following equation:

$$\tan\gamma = \frac{d_y - e_y}{e_x - d_x} = \frac{c_y + (\sin\gamma)k - (\cos\gamma)r}{a_x + (\sin\gamma)r - c_x + (\cos\gamma)k}$$

The solution set for γ, as it was derived by *Maple*[2], is given by

$$\gamma = \{\gamma_1, \gamma_2\} =$$
$$\left\{\arctan\left(-\frac{-c_y\sqrt{-(a_x-c_x)^2(-a_x^2+2c_xa_x-c_y^2-c_x^2+r^2)}+ra_x^2-2rc_xa_x+rc_x^2}{(c_y^2+a_x^2-2c_xa_x+c_x^2)(a_x-c_x)},\right.\right.$$
$$\left.\frac{rc_y+\sqrt{-(a_x-c_x)^2(-a_x^2+2c_xa_x-c_y^2-c_x^2+r^2)}}{c_y^2+a_x^2-2c_xa_x+c_x^2}\right),$$

$$\arctan\left(-\frac{c_y\sqrt{-(a_x-c_x)^2(-a_x^2+2c_xa_x-c_y^2-c_x^2+r^2)}+ra_x^2-2rc_xa_x+rc_x^2}{(c_y^2+a_x^2-2c_xa_x+c_x^2)(a_x-c_x)},\right.$$
$$\left.\left.\frac{rc_y-\sqrt{-(a_x-c_x)^2(-a_x^2+2c_xa_x-c_y^2-c_x^2+r^2)}}{c_y^2+a_x^2-2c_xa_x+c_x^2}\right)\right\} \quad (6.10)$$

Whereas γ_1 is the solution for a positive result of the square roots in equation 6.10, γ_2 is its counterpart for negative square roots. However, only γ_1 needs to be considered as the solution, because C++'s `sqrt` function always returns the non-negative square root.
Having the rotation angle γ the translation t can now be computed by

$$t = \frac{a_x - (c_x - (\cos\gamma)k - (\sin\gamma)r)}{\cos\gamma}.$$

6.3.2.2 Three Arms

For the non-trivial case to visualize three arms without collisions a different approach was chosen. Although it is not giving the optimal solution, it aims to be a good approximation within a reasonable tolerance. As figure 6.6(a) is showing, the approach tries to attach all three arms to a circle in such a way that their inner or outer cylinder, respectively, are just touching the circle.

In detail, following steps are executed to visualize the triplet of instrument arms. Therein, analog to the visualization of two instrument arms, an arm consists of two cylinders. In the following only the computations for one of these cylinders is depicted, wherefore 'arm' and 'cylinder' can be considered to be synonymous.

In the beginning, it is checked for a pairwise intersection of cylinders, when they are aligned with the line defined by the landmarks of the arms (A, B, and C) and the region

[2] *Maple* is a computer algebra system provided by *Maplesoft* – `www.maplesoft.com`

of interest's center O, which they have a desired distance d to. This is achieved in the same way as for two arms.

If no collisions can be detected, the arms are visualized. If a collision occurs, all arms are rearranged around a circle, whose normal is aligned with the line through O and the geometric centroid Y of the tetrahedron $ABCO$ and whose center Z is on this line. The arrangement is performed in such a way that the arms are touching the circle at the points P', Q', and R'. To entirely ensure collision freedom between all arms, the radius ρ of the circle matches the cylinder's radius r. Therewith, the worst case of having all arms coplanar to the circle and two of them parallel to each other, does not lead to intersections. However, for a common setup that is still capable of avoiding most of the collision cases the radius can be reduced. The circle's center Z equals O, if the required distance d is smaller than r. In any other case it is on the line through O and Y, so all circle points including P', Q', and R' have their required distance d to O.
To determine all corresponding circle points for A, B, and C, two heuristics are applied. First, all angles between A, B, and C are computed. If at least one of them is greater than 90 degrees, then the three points approximate a line, wherein a 'middle' point can be found, which is also set on the circle first. If all angles are smaller than 90 degrees, in a next step the points A, B, and C are projected onto the plane, the circle lies in. The projection point with the greatest distance to the circle's center Z is set as a starting point on the circle.

Now that the first point is set on the circle the other circle points can be set 120 degrees away from the starting point. Therefore, a circle lying in its plane $\vec{N}(\vec{X} - \vec{Z}) = 0$ is parameterized as

$$\vec{X} = \vec{Z} + \rho((\cos\delta)\vec{U} + (\sin\delta)\vec{V}), \tag{6.11}$$

where $\vec{N}$ is a unit length normal to the plane, obtained by normalizing $\vec{Y} - \vec{Z}$, so $\vec{U}$, $\vec{V}$, and $\vec{N}$ form a right-handed orthonormal coordinate system. Assuming that P' was chosen to be the starting point on the circle, $\vec{U}$ is obtained by normalizing $\vec{P'} - \vec{Z}$, so $\vec{V}$, which is of unit length as well, can be computed by $\vec{N} \times \vec{U}$. All computations for Q' and R' are analog.
The coordinates of P', Q', and R' are

$$\vec{P'} = \vec{Z} + \rho\vec{U},$$

$$\vec{Q'} = \vec{Z} + \rho(-\frac{1}{2}\vec{U} + \frac{\sqrt{3}}{2}\vec{V}),$$

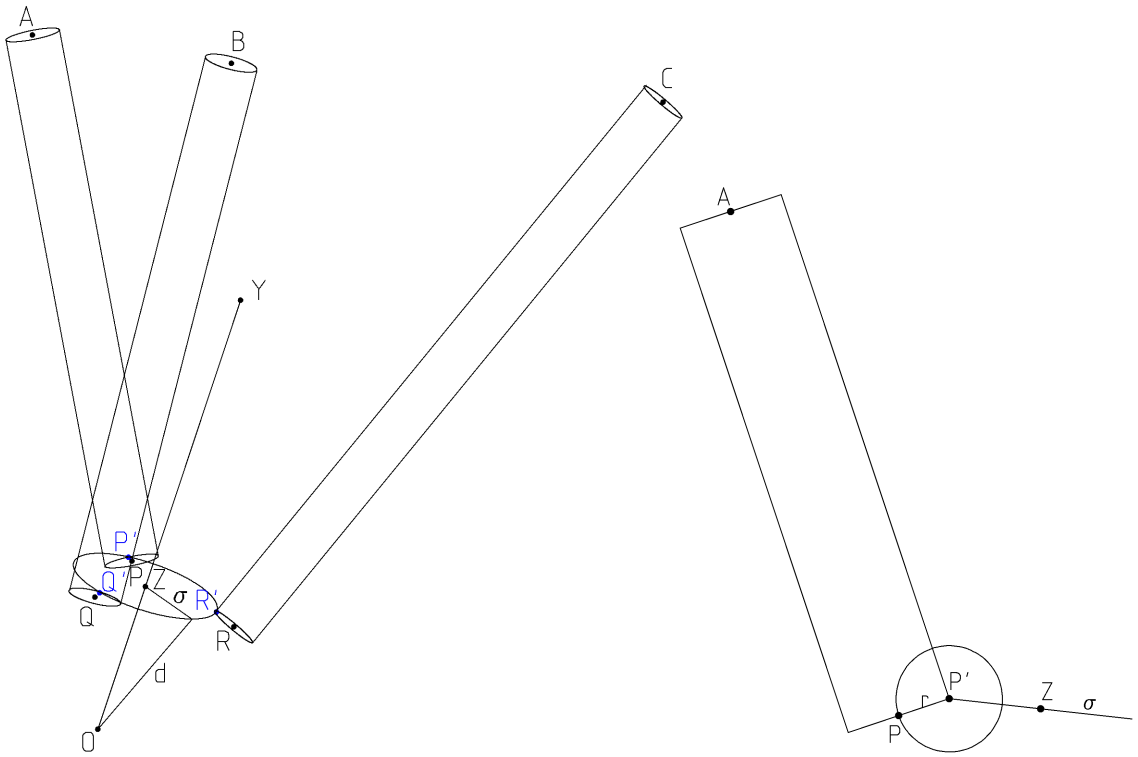
and

$$\vec{R'} = \vec{Z} + \rho(-\frac{1}{2}\vec{U} - \frac{\sqrt{3}}{2}\vec{V}).$$

The plane through P', Z, and Y defines two half-spaces, one for each of the two other circle points. Depending on which half-space Q' or R', respectively, lies in, aforesaid coordinates might be swapped.

Finally, P is computed as shown in figure 6.6(b). Again, the computations for Q and R are analog. A new circle is defined, which is lying in the plane through P', Z, and A. It has the radius r of the robot arm's cylinder and a center in P'. Similarly to equation 6.11, the coordinates of P can be determined by

$$\vec{P} = \vec{P'} + r((\cos\delta)\vec{U} + (\sin\delta)\vec{V}),$$



(a) All cylinders of the instrument arms defined by the landmarks A, B, and C are aligned in such a way that they are touching a circle with a radius ρ around a center Z. The circle is orthogonal to the line through the region of interest's center O and the geometric centroid Y of tetrahedron $OABC$.

(b) For computing the cylinder point P a new circle is defined having a radius equally to the cylinder radius r and a center P'. It is lying in the plane through P', Z, and A.

Figure 6.6: Considerations for computing a collision-free fit for a triplet of instrument arms.

where $\delta = \arccos \frac{r}{|\vec{A}-\vec{P'}|}$, and $\vec{U}$, $\vec{V}$, and $\vec{N}$ is the basis of a right-handed orthonormal coordinate system. $\vec{U}$ can be found by normalizing $\vec{A} - \vec{P'}$, $\vec{N}$ by computing the normal of plane P', Z, and A in unit length, and $\vec{V}$ by taking the cross product $\vec{N} \times \vec{U}$.
The instrument arm's cylinder can now be aligned with $\vec{A} - \vec{P}$, having its tool tips in $\vec{P}$.

7 Conclusion

> A conclusion is the place where you got tired thinking.
>
> Martin H. Fischer

This chapter summarizes the results for the design and implementation of the port placement planning tool. The problems that arose in the course of this work are discussed in detail in section 7.2. Thereafter, possible improvements are identified, which are addressed in future work, as depicted in section 7.3.

7.1 Result

Within the scope of this work a fully functional prototype for the port placement in robotically assisted minimally invasive cardiovascular surgery was developed as an add-on to amira. As it is based on miscellaneous platform-independent industry standards, the planning system itself remains almost entirely independent of the used operating system. It runs on *Linux*, *Solaris*, *HP-UX*, *IRIX*, and *Windows*, while only the amira standard edition is required for the execution of the planning system.

Through various discussions with physicians at the *Deutsches Herzzentrum München* its desired minimum as well as 'nice-to-have' functionality was defined and implemented. It is very important for cardiac surgeons, who will use the planning tool, to test and verify different *ports* with varying *regions of interest* for an optimally planned intervention. Therefore, the straightforward concept of adjustable landmarks provides them with a maximum degree of freedom. Additionally, collisions of robot arms with each other and with vitals or bones can be detected on request at a satisfactory speed. Also, while the *virtual endoscope view* is closing the gap between reality and virtuality by simulating an operative view for the surgeons, the gap between two and three dimensions is closed by visualizing both computed tomography slices and a reconstructed segmented model aligned in the same coordinate system.

Since the virtual planning tool developed in this work and the augmented navigation tool developed by *Jörg Traub* [79] are interdependent and complementary to one another, solely planning a perfect port placement without its intra-operative visualization would not be of any use to a surgeon. However, the availability of an overall planning and navigation prototype is – compared to the current situation – a big step towards an enhanced and more sophisticated port placement process. The prototype's final functionality was presented within a series of lectures at the *Deutsches Herzzentrum München*. Having got encouraging and very positive feedback from the participating surgeons and radiologists, it would be a distinct possibility to carry on the project and develop a mature product.

7.2 Problems and Discussion

During the design and implementation of the planning system a couple of problems arose, some of which will be discussed below.

First of all, the planning system relies heavily on its core amira, whose functionality as well as bugs can be transferred to the planning tool. amiraDev is missing several C++ header files, so existing modules cannot be extended and are only accessible via the cumbersome *Tcl* interface. Additionally, amira crashed arbitrarily, which complicated an otherwise systematic and reliable testing phase of the planning module. When it comes to image segmentation, it only features very basic, mainly manual techniques, resulting in a very time-consuming interactive process to obtain a classified volume of a patient. For instance, it usually takes a small group of assistants up to a week to achieve an adequate segmentation. A computed tomography scan, however, is performed just the day before an intervention, so faster and fully-automated segmentation techniques are required by the planning system. Therefore, methods stated in chapter 5 may be considered for implementation. Nevertheless, amira was a good starting point to design the planning system. Worth mentioning is its capability – contrary to *Open Inventor* – to display large data sets, which often emerge in medical applications. Future releases with more functions and reliability have promise to serve as a solid base for standard applications not requiring exceptional functionality. If time, however, is not a limiting factor during development of similar medical applications, it might be advisable to implement the required functionality independently, e.g. by extending free software tools such as *Open Inventor*.

As with every project, the unexpected happens in the course of development. In this case the collision-free visualization of more than two telemanipulator arms in real time turned out to be a significant problem. Many more approaches than the one presented in section 6.3.2.2 were considered, but none of them led to a satisfactory result. However, various tests confirmed that the one presented based on a worst-case scenario is a robust approximation. It appears to be sufficient for the port planning process, where an accuracy of less than one centimeter is desirable. Additionally, collision detection techniques applied to large data sets require a longer runtime than expected. Therefore, triangle collisions are only checked on request and not for every change in the position of landmarks.

Finally, the system is after all a prototype and lacks testing and evaluation in the operating theater. Further drawbacks as well as caveats need to be identified by surgeons. For instance, the human thorax and the heart, which are continuously deformed, are both considered to be rigid during the planning process. It needs to be evaluated, how strongly the disregard for the fourth dimension time influences an optimal port placement. Moreover, only the telemanipulator arms are modelled for the planning process, but not the entire robot, whose configuration limits the arm positions. Therefore, it needs to be tested, how often the virtual telemanipulator arms happen to reach impossible positions. If the frequency is too high, the telemanipulator itself needs to be considered as well.

7.3 Future Work

As denoted before, the planning tool has not undergone a series of analyzes, tests, and evaluations. Therefore, additional evaluations need to be executed to determine the effect of certain disregarded requirements, which were not prioritized during the implementation phase. For instance, an automatic computation of the best fit for all telemanipulator arms need to be developed and tested. Moreover, an interface to the intra-operative navigation tool may be made available in order to visualize the telemanipulator arms in the accustomed planning environment.

Furthermore, a couple of suggestions for improvements on the planning tool emerged during its implementation. First of all, more efficient collision detection algorithms need to be implemented, so any adjustments to the telemanipulator arms can be checked for intersections in real time and automatically, and not only upon request, leading to a more effective and less time-consuming planning process. Another limitation of the current planning system is that the telemanipulator arms' positions are all defined via landmarks. A more intuitive way of placing the arms may be by simply selecting them and moving them around. However, the positions of the arms' tool tips are still constrained by a region of interest. Now, an arm's body can be selected to adjust its position arbitrarily by dragging it with a pointing device. Coupled with the concept of real-time collision detections, an optimally adjusted, collision-free arm position might alter its intuitive color, for instance from red to green.

Based on an optimal planning system with the aforementioned improvements, an enhanced planning situation could be realized. Therein, surgeons are able to perform the planning on a fully functional, four-dimensional model of the patient, which is capable of realistically deforming its thorax and heart. Its functionality can be achieved by real-time deformation techniques developed at the *Computer Graphics and Visualization Group*[1] of the *Technische Universität München*. Additionally, the surgeons are provided with a three-dimensional view on the patient's model through a similar display device as the one used for a three-dimensional sketching device[2] developed by the *Lehrstuhl für Produktentwicklung* at the *Technische Universität München*, where surgeons need to wear shutter glasses to get a realistic three-dimensional impression. In contrast, 'glasses-free' approaches might be considered, which are based on three-dimensional monitors such as the ones provided by *X3D Technologies*[3] or *StereoGraphics*[4]. As a whole, a very realistic and intuitive planning process can be achieved.

[1] `wwwcg.in.tum.de`
[2] `www.pe.mw.tu-muenchen.de/index.php?inhalt=projekt&nummer=18`
[3] `www.4d-vision.de`
[4] `www.stereographics.com`

Appendix

A Use Cases

Use Case: **Acquire3DModels**

Participating Actor: Initiated by `Surgeon`

Communicates with `TelemanipulatorArms, CTData`

Flow of Events:

1. The `Surgeon` initializes the pre-operative planning tool.
2. The `Surgeon` loads the `CTData` into the planning tool.
3. The `Surgeon` uses automatic and manual segmentation techniques to classify the `CTData`.
4. The planning tool reconstructs a three-dimensional model of the patient out of the segmented `CTData`.
5. The `Surgeon` creates a three-dimensional model of the `TelemanipulatorArms`, if not previously done so.
6. The `Surgeon` saves all three-dimensional models.

Use Case: **PlanPortPlacement**

Participating Actor: Initiated by `Surgeon`

Communicates with `CTData`

Flow of Events:

1. If not done yet, the `Surgeon` initializes the pre-operative planning tool.
2. The `Surgeon` loads the three-dimensional models of the patient and the telemanipulator arms.
3. The `Surgeon` loads the `CTData`.
4. The `Surgeon` sets the region of interest for the operation.
5. The planning tool suggests the best poses (position and orientation) for the telemanipulator arms and their corresponding ports.
6. The `Surgeon` improves/redefines the suggested poses interactively by means of the models and the `CTData`.
7. The poses of the ports are saved.

Use Case: **ValidatePortPlanning**

Participating Actor: Initiated by `Surgeon`
Communicates with `CTData`

Flow of Events:
1. If not done yet, the `Surgeon` loads the three-dimensional models and the planned poses for the ports into the planning tool.
2. The `Surgeon` uses the camera view mode of the planning tool to see exactly what the endoscope views.
3. The `Surgeon` activates a collision detection mode to avoid collisions between the models of the telemanipulator arms and vitals or bones.
4. The planning tool displays the `CTData` to the `Surgeon`, so they can be aligned with the planned data.
5. The `Surgeon` improves/redefines the suggested pose interactively and might continue with step two.
6. The adjusted poses of the ports are saved.

Use Case: **ExportPlannedData**

Participating Actor: Initiated by `Surgeon`
Communicates with `SceneGraphData`

Flow of Events:
1. If not done yet, the `Surgeon` loads the three-dimensional models and the planned poses for the ports into the planning tool.
2. The `Surgeon` exports the three-dimensional model of the patient as `SceneGraphData`.
3. The `Surgeon` exports the planned three-dimensional model of the telemanipulator arms as `SceneGraphData`.
4. The planning tool is shut down.

B Telemanipulator Arms as Open Inventor Scene Graphs

Endoscope Arm

```
#Inventor V2.1 ascii
DEF base Separator
{
    DEF base_trans Transform
    {
        translation 0 -18.75 0
    }
    DEF base_separator Separator
    {
        DEF arm Separator
        {
            DEF arm_body Cylinder
            {
                radius 0.6
                height 37.5
            }
            Separator
            {
                Transform
                {
                    translation 0 -26.1 0
                }
                DEF arm_foot Cylinder
                {
                    radius 1.95
                    height 14.7
                }
            }
        }
    }
}
```

Instrument Arm

```
#Inventor V2.1 ascii
DEF base Separator
{
    DEF base_trans Transform
    {
```

```
        translation 0 -21.0 0
    }
    DEF base_separator Separator
    {
        DEF arm Separator
        {
            DEF arm_body Cylinder
            {
                radius 0.5
                height 38.2
            }
            Separator
            {
                Transform
                {
                    translation 0 -23.85 0
                }
                DEF arm_foot Cube
                {
                    width 3.0
                    height 9.5
                    depth 7.0
                }
            }
            Separator
            {
                Transform
                {
                    translation 0 19.6 0
                }
                DEF arm_head Cylinder
                {
                    radius 0.3
                    height 1.0
                }
            }
            Separator
            {
                Transform
                {
                    translation 0 20.55 0
                }
                DEF tool_tips Cube
                {
                    width 0.3
                    height 0.9
                    depth 0.3
                }
            }
        }
    }
}
```

List of Figures

Glossary

3D. Three dimensions; *see* THREE-DIMENSIONAL

AR. *see* AUGMENTED REALITY

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

C-Arm. A mobile X-RAY system to provide fluoroscopic and radiographic imaging. It is used for a variety of clinical procedures, especially intra-operatively.

CAT. COMPUTED AXIAL TOMOGRAPHY, *see* COMPUTED TOMOGRAPHY

Collision Detection. A technique used to determine or predict the occurrence of a contact between geometric models.

Computed Tomography. A technique for producing two-dimensional cross-sectional axial image slices of an object such as the human body by X RAYS.

CT. *see* COMPUTED TOMOGRAPHY

Degrees of Freedom. The number of independent variables or parameters required to specify the POSE of an object. Very often, they are used to classify robot arms.

DICOM. *see* DIGITAL IMAGING AND COMMUNICATIONS IN MEDICINE

Digital Imaging and Communications in Medicine. A standard created by the *National Electrical Manufacturers Association* (*NEMA*) to aid the distribution and viewing of medical images, such as COMPUTED TOMOGRAPHY scans and MAGNETIC RESONANCE or ULTRASOUND images.

DOF. *see* DEGREES OF FREEDOM

DWARF. Distributed Wearable Augmented Reality Framework. A framework system that provides *Corba* based communication between different augmented reality components developed at the *Technische Universität München*.

Field of View. The maximum, mostly angular area that is visible through a lens or an optical instrument. It can be distinguished between a horizontal and a vertical field of view.

FOV. *see* FIELD OF VIEW

Global Positioning System. A satellite-based radio navigation system such as the American *NAVSTAR*, the Russian Federation *GLONASS*, or the European *GALILEO*.

GPS. *see* GLOBAL POSITIONING SYSTEM

HMD. *see* HEAD-MOUNTED DISPLAY

Head-Mounted Display. Also referred to as head up display is worn on the user's head directly in front of his or her eyes.

LED. *see* LIGHT EMITTING DIODE

Light Emitting Diode. A semiconductor device that emits visible light when charged with electricity.

Magnetic Resonance Imaging. An imaging technique used primarily in medical settings to produce high quality images of the interior of the human body. It is based on the principles of NUCLEAR MAGNETIC RESONANCE.

Matching. To bring two modalities/images into spatial alignment in order to compare the modalities/images directly.

Minimally Invasive Surgery. An operative procedure, i.e. a therapeutic or diagnostic procedure that wounds the surface of the body. It is performed with smaller transections ($< 10mm$) of the patients' skin and less exposures to the patients themselves than a traditional invasive procedure.

MIS. *see* MINIMALLY INVASIVE SURGERY

Mixed Reality. A type of VIRTUAL REALITY that combines real and virtual objects.

MR. *see* MIXED REALITY

MRI. *see* MAGNETIC RESONANCE IMAGING

NMR. *see* NUCLEAR MAGNETIC RESONANCE

Nuclear Magnetic Resonance. A spectroscopic technique to obtain microscopic chemical and physical information about molecules.

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.

Port. An incision point in MINIMALLY INVASIVE SURGERY.

Pose. Combines the position and orientation of a point or object in a six DEGREES OF FREEDOM vector.

Registration. Process of estimating the parameters for the matching of two modalities/images.

RGBA. A four-vector data set composed of three values for the colors red, green, and blue, as well as a fourth alpha component called opacity or transparency.

RMIS. *see* ROBOTICALLY ASSISTED MINIMALLY INVASIVE SURGERY

Robotically Assisted Minimally Invasive Surgery. MINIMALLY INVASIVE SURGERY supported by robots such as TELEOPERATORS.

Segmentation. An imaging technique used to cluster or isolate similar regions and label them.

Telemanipulator. *see* TELEOPERATOR

Teleoperator. A device that is able to extend the operator's sensory and manipulatory abilities to a remote location. It needs to have sensors, arms and/or hands, and multimodal communication channels to and from the operator.

Telepresence. A human-machine interface, in which the human operator of a technical system receives sufficient information about the TELEOPERATOR and the task environment. All information is displayed in a natural way, so the human operator feels physically present at a distant operation site.

Three-Dimensional. Dealing with or restricted to a space in which location can be completely described with exactly three orthogonal axes. In computers, tridimensionality refers to an image that provides the perception of depth.

UI. *see* USER INTERFACE

Ultrasound. Sound waves in the 210 MHz frequency range, that are used in medical diagnostics to produce sectional images or to measure and image blood flow.

User Interface. Everything designed into an information device, which a human being interacts with, for instance a display screen and its appearance/ways of interaction, keyboard, mouse, and many more. In case of a display it is also referred to as graphical user interface (GUI).

Virtual Reality. A computer based technology that allows its user to act in purely virtual environments.

VR. *see* VIRTUAL REALITY

X ray. Also referred to as *Röntgen* ray, is an electromagnetic radiation of short wavelength produced when highly energetic electrons strike a solid target. By exposing photographic film to X rays radiograms can be made.

Bibliography

[1] *The Definition of Telepresence*. `http://3dgraphics.about.com/library/glossary/bldef-Telepresence.htm`.

[2] *Digital Imaging and Communications in Medicine (DICOM) Set*. `http://medical.nema.org/dicom/2003.html`.

[3] L. ADHAMI, *An Architecture for Computer Integrated Mini-Invasive Robotic Surgery: Focus on Optimal Planning*, PhD thesis, École des Mines de Paris, 2002.

[4] L. ADHAMI, È. COSTE-MANIÈRE, and J.-D. BOISSONNAT, *Planning and Simulation of Robotically Assisted Minimal Invasive Surgery*, in Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI'00), vol. 1935 of Lect. Notes in Comp. Sc. 1954, Springer, Oct. 2000.

[5] L. M. AUER, D. AUER, and J. F. KNOPLIOCH, *Virtual endoscopy for planning and simulation of minimally invasive neurosurgery*, Lecture Notes in Computer Science, 1205, 1997, pp. 315–325.

[6] S. AUKSTAKALNIS and D. BLATNER, *Silicon Mirage: The Art and Science of Virtual Reality*, Peachpit Press, October 1992.

[7] R. AZUMA, *A survey of Augmented Reality*, Precense, 6(4), 1997, pp. 355–385.

[8] R. AZUMA, Y. BAILLOT, R. BEHRINGER, S. FEINER, S. JULIER, and B. MACINTYRE, *Recent Advantages in Augmented Reality*, IEEE Computer Graphics and Applications, 21(6), 2001, pp. 34–47.

[9] D. BARTZ, W. STRASSER, M. SKALEJ, and D. WELTE, *Interactive Exploration of Extra- and Interacranial Blood Vessels*, in IEEE Visualization '99, D. Ebert, M. Gross, and B. Hamann, eds., San Francisco, 1999, IEEE, pp. 389–392.

[10] M. BAUER, B. BRUEGGE, G. KLINKER, A. MACWILLIAMS, T. REICHER, S. RISS, C. SANDOR, and M. WAGNER, *Design of a Component-Based Augmented Reality Framework*, in Proceedings of ISAR 2001, 2001.

[11] R. BAUERNSCHMITT, H. MEHMANESH, and R. LANGE, *Telemanipulators in cardiac surgery: towards minimally invasive operations on the heart*, in Advances in Interactive Multimodal Telepresence Systems, Technische Universität München, Germany, March 2001, pp. 125–133.

[12] H. BEHNKE, F. BACHMANN, K. FLADT, and W. SÜSS, *Foundations of Mathematics: The Real Number System and Algebra*, The MIT Press, Cambridge, Massachusetts, and London, 1986.

[13] J. BEZDEK, L. HALL, and L. CLARKE, *Review of MR Image Segmentation Techniques Using Pattern Recognition*, Medical Physics, 20(4), July 1993, pp. 1033–1048.

[14] U. BOCKHOLT, A. BISLER, M. BECKER, and W. M.-W. G. VOSS, *Augmented Reality for Enhancement of Endoscopic Interventions*, in IEEE Virtual Reality, March 2003, pp. 97–101.

[15] S. CARLSSON, *Geometric Computing in Image Analysis and Visualization*. Lecture Notes, 2002.

[16] H. CHOI, D. HAYNOR, and Y. KIM, *Partial Volume Tissue Classification of Multichannel Magnetic Resonance Images - A Mixel Model*, IEEE Transactions on Medical Imaging, 10(3), September 1991, pp. 395–407.

[17] G. CHRISTENSEN, S. JOSHI, and M. MILLER, *Volumetric transformation of brain anatomy*, IEEE Trans. Med. Imag., 16(6), 1997, pp. 864–877.

[18] K. CLEARY and C. NGUYEN, *State of the Art on SUgical Robotics: Clinical Applications and Technology Challenges*, Computer Aided Surgery, 6(6), 2003, pp. 312–328.

[19] G. B. COLEMAN and H. C. ANDREWS, *Image Segmentation by Clustering*, Proc. IEEE, 67(5), May 1979, pp. 773–785.

[20] È. COSTE-MANIÈRE, L. ADHAMI, F. MOURGUES, O. BANTICHE, D. LE, D. HUNT, N. SWARUP, K. SALISBURY, and G. GUTHART, *Optimal Planning of Robotically Assisted Heart Surgery: Transfer Precision in the Operating Room*, in Lecture Notes in Control and Information Sciences, Experimental Robotics VIII, to appear, 2002.

[21] O. DEVILLERS and P. GUIGUE, *Faster Triangle-Triangle Intersection Tests*, Rapport de recherche 4488, Projet PRISME, Unité de recherche INRIA Sophia Antipolis, 2002.

[22] O. FAUGERAS, *Three-Dimensional Computer vision – A Geometric Viewpoint*, The MIT Press, 1 ed., 1993.

[23] M. FIGL, W. BIRKFELLNER, C. EDE, J. HUMMEL, R. HANEL, F. WATZINGER, F. WANSCHITZ, R. EWERS, and H. BERGMANN, *The Control Unit for a Head Mounted Operating Microscope Used for Augmented Reality Visualization in Computer Aided Sugery*, in International Symposium on Mixed and Augmented Reality (ISMAR), September 2002, pp. 69–75.

[24] M. FIGL, W. BIRKFELLNER, J. HUMMEL, R. HANEL, P. HOMOLKA, F. WATZINGER, F. WANSCHITZ, R. EWERS, and H. BERGMANN, *Current Status of the Varioscope AR, a Head-Mounted Operating Microscope for Computer-Aided Surgery*, in International Symposium on Augmented Reality (ISAR), October 2001, pp. 20–29.

[25] J. D. FOLEY, A. V. DAM, S. K. FEINER, and J. F. HUGHES, *Computer Graphics: Principles and Practice*, Systems Programming Series, Addison-Wesley, Reading, Massachusetts, second ed., 1990.

[26] H. FUCHS, A. LIVINGSTON, R. RASKAR, D. CULUCCI, K. K. ADN A. STATE, J. CRAWFORD, P. RADEMACHER, S. DRAKE, and A. MEYER, *Augmented Reality Visualization for Laparoscopic Surgery*, in MICCAI - Medical Image Computing and Computer-Assisted Intervention, Springer, 1998, pp. 934–943.

[27] E. GELENBE, T. FENG, and K. KRISHNAN, *Neural Network Methods for Volumetric Magnetic Resonance Imaging of the Human Brain*, Proceedings of the IEEE, 84(10), October 1996, pp. 1488–1496.

[28] M. GOPI, S. KRISHNAN, and C. T. SILVA, *Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation*, EUROGRAPHICS, 19(3), 2000.

[29] W. E. L. GRIMSON, R. KIKINIS, F. A. JOLESZ, and P. M. BLACK, *Image-Guided Surgery*, Scientific American, 280(6), 1999, pp. 62–69.

[30] M. GROHER, *Development of a Planning and Navigation Tool for Endoscopic Treatment of Aortic Aneurysms - Computer Supported Implantation of a Stent Graft*, Master's thesis, Technische Universität München, Department of Computer Science, December 2003.

[31] J. V. HAJNAL, D. L. HILL, and D. J. HAWKES, eds., *Medical Image Registration*, Biomedical Engineering Series, CRC Press, 2001.

[32] L. HALL, A. BENSAID, L. CLARKE, R. VELTHUIZEN, M. SILBIGER, and J. BEZDEK, *A Comparison of Neural Network and Fuzzy Clustering Techniques in Segmenting Magnetic Resonance Images of the Brain*, IEEE Transactions on Neural Networks, 3(5), September 1992, pp. 672–682.

[33] K. HELD, E. R. KOPS, B. J. KRAUS, W. M. WELLS III, R. KIKINIS, and H. W. MÜLLER-GÄRTNER, *Markov Random Field Segmentation of Brain MR Images*, IEEE Trans. on Medical Imaging, 16(6), 1997.

[34] R. HERNDON, J. LANCASTER, A. TOGA, and P. FOX, *Quantification of White Matter and Gray Matter Volumes Using T1 Parametric Images Using Fuzzy Classifiers*, Journal of Magnetic Resonance Imaging, 6, 1996, pp. 425–435.

[35] J. ISDALE, *What is Virtual Reality? A Web-Based Introduction*. `http://www.isdale.com/jerry/VR/WhatIsVR.html`, September 1998.

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

[37] 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, Department of Computer Science, December 2003.

[38] 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.

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

[40] F. LANGLOTZ, M. STUCKI, R. BCHLER, C. SCHEER, R. GANZ, U. BERLEMANN, and L.-P. NOLTE, *The first twelve cases of computer assisted periacetabular osteotomy*, Computer Aided Surgery, 2(6), 1997, pp. 317–326. to be added.

[41] M. LEVOY, *Display of Surfaces from Volume Data*, IEEE Computer Graphics and Applications, 8(3), 1988, pp. 29–37.

[42] Z. LIANG, *Tissue Classification and Segmentation of MR Images*, IEEE Engineering in Medicine and Biology Magazine, 12(1), March 1993, pp. 81–85.

[43] M. C. LIN and S. GOTTSCHALK, *Collision detection between geometric models: a survey*, pp. 37–56.

[44] Z. LIN, J. JIN, and H. TALBOT, *Unseeded Region Growing for 3D Image Segmentation*, in Selected papers from Pan-Sydney Workshop on Visual Information Processing, P. Eades and J. Jin, eds., vol. 2, Sydney, Australia, 2000, ACM International Conference Proceeding Series (ACS), pp. 31 – 37.

[45] H. K. LIU, *Two- and Three-Dimensional Boundary Detection*, Computer Graphics and Image Processing, 6(2), April 1977, pp. 123–134.

[46] M. LORENZO-VALDÉS, G. I. SANCHEZ-ORTIZ, R. MOHIADDIN, and D. RUECKERT, *Atlas-Based Segmentation and Tracking of 3D Cardiac MR Images Using Non-rigid Registration*, in MICCAI - Medical Image Computing and Computer-Assisted Intervention, vol. 1, Springer, 2002, pp. 642–650.

[47] K.-L. LOW, A. ILIE, G. WELCH, and A. LASTRA, *Combining Head-Mounted and Projector-Based Displays for Surgical Training*, in IEEE Virtual Reality, March 2003, pp. 69–75.

[48] A. MACOVSKI, *Medical Imaging Systems*, Prentice-Hall information and system sciences series, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1983.

[49] I. MANOUSAKAS, P. UNDRILL, T. REDPATH, and G. CAMERON, *Split and Merge Segmentation of Magnetic Resonance Medical Images: Performance Evaluation and Extension to Three Dimensions*, Computers and Biomedical Research, 31, 1998, pp. 393–412.

[50] M. MASON, *Mechanics of Manipulation - Representing Rotation*. Lecture Notes, Carnegie Mellon University, 2003.

[51] T. MCINERNEY and D. TERZOPOULOS, *Deformable Models in Medical Images Analysis: A Survey*, Medical Image Analysis, 1(2), 1996, pp. 91–108.

[52] J. MELZER and K. MOFFITT, eds., *Head-Mounted Displays: Designing For The User*, McGraw-Hill, 1997.

[53] P. MILGRAM and F. KISHINO, *A taxonomy of mixed reality visual displays*, IEICE Trans. on Information and Systems (Special Issue on Networked Reality), E77-D(12), 1994, pp. 1321–1329.

[54] T. MÖLLER, *A Fast Triangle-Triangle Intersection Test.*, Journal of Graphics Tools: JGT, 2(2), 1997, pp. 25–30.

[55] Y. OHTA and H. TAMURA, eds., *Mixed Reality : Merging Real and Virtual Words*, Springer Verlag, 1999.

[56] T. S. PALEOLOGOS, J. P. WADLEY, N. D. KITCHEN, and D. G. THOMAS, *Clinical Utility and Cost-effectiveness of Interactive Image-guided Craniotomy: Clinical Comparison between Conventional and Image-guided Meningioma Surgery*, Neurosurgery, 47(1), 2000, p. 40. to be added.

[57] D. L. PHAM and J. L. PRINCE, *An Adaptive Fuzzy C-means Algorithm for Image Segmentation in the Presence of Intensity Inhomogeneities*, in SPIE Medical Imaging 1998: Image Processing, vol. 3338, 1998, pp. 555–563.

[58] D. L. PHAM, C. XU, and J. L. PRINCE, *A Survey of Current Methods in Medical Image Segmentation*, Technical Report JHU/ECE 99-01, Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore MD 21218, 1999.

[59] W.-C. C. RAMESH RASKAR, GREG WELCH, *Table-Top Spatially-Augmented Reality: Bringing Physical Models to Life with Projected Imagery*, in MICCAI - Medical Image Computing and Computer-Assisted Intervention, 1999, pp. 64–74.

[60] W. REDDICK, J. GLASS, E. COOK, T. ELKIN, and R. DEATON, *Automated Segmentation and Classification of Multispectral Magnetic Resonance Imaging of Brain Using Artificial Neural Networks*, IEEE Transactions on Medical Imaging, 16(6), December 1997, pp. 911–918.

[61] T. R. REED and J. M. H. D. BUF, *A Review of Recent Texture Segmentation and Feature Extraction Techniques*, Computer Vision, Graphics, and Image Processing, 57(3), May 1993, pp. 359–372.

[62] M. ROSENTHAL, J. L. A. STATE, G. HIRATO, J. ACKERMAN, K. KELLER, E. PISANO, M. JIROUTEK, K. MULLER, and H. FUCHS, *Augmented Reality Guidance for Needle Biopsies: A Randomized, Controlled Trial in Phantoms*, in MICCAI - Medical Image Computing and Computer-Assisted Intervention, Springer, October 2001, pp. 241–248.

[63] S. RUSSELL and P. NORVIG, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[64] P. SAHOO, S. SOLTANI, and A. WONG, *A Survey of Thresholding Techniques*, Comput. Vision Graphics Image Process., 41, 1988, pp. 233–260.

[65] B. SANKUR, *A Survey over Image Thresholding Techniques and Quantitative Performance Evaluation*. 2003.

[66] B. SANKUR and M. SEZGIN, *A Survey Over Image Thresholding Techniques And Quantitative Performance Evaluation*, Journal of Electronic Imaging (JEI), to appear, 2003.

[67] M. SATI, H.-U. STÄUBLI, Y. BOURQUIN, M. KUNZ, and L.-P. NOLTE, *Real-time computerized in situ guidance system for ACL graft placement*, Computer Aided Surgery, 7(1), 2002, pp. 25–40. to be added.

[68] R. J. SCHALKOFF, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, New York, 1991.

[69] H. SEIBERT, M. SCHNAIDER, B. SCHWALD, and T. WELLER, *Medarpa - A Medical Augmented Reality System for Minimal-Invasive Interventions*, tech. rep., Zentrum füer Graphische Datenverarbeitung e.V. (ZGDV), Juli 2002.

[70] J. SERRA, *Image Analysis and Mathematical Morphology*, vol. 1, Academic Press, London, England, 1982.

[71] A. R. SMITH, *Volume Graphics and Volume Visualization: A Tutorial*, Tech Memo 176, Pixar, May 1987.

[72] G. SOETE, J. V. DE STEENE, D. VERELLEN, V. VINH-HUNG, D. V. DEN BERGE, D. MICHIELSEN, F. KEUPPENS, P. D. ROOVER, and G. STORME, *Clinical Use of Stereoscopic X-ray Positioning of Patients Treated with Conformal Radiotherapy for Prostate Cancer*, International Journal of Radiation Oncology-Biology-Physics, 54(3), 2002, pp. 948–952.

[73] G. SOETE, J. V. DE STEENE, D. VERELLEN, V. VINH-HUNG, D. V. DEN BERGE, D. MICHIELSEN, F. KEUPPENS, P. D. ROOVER, and G. STORME, *Initial clinical experience with infrared-reflecting skin markers in the positioning of patients treated by conformal radiotherapy for prostate cancer*, International Journal of Radiation Oncology-Biology-Physics, 52(3), 2002, pp. 694–698.

[74] J. SURI, S. SINGH, and L. REDEN, *Computer Vision and Pattern Recognition Techniques for 2-D and 3-D Cerebral Cortical Segmentation (Part-I): A State of the Art Review*, 2002.

[75] H. TAMURA, H. YAMAMOTO, and A. KATAYAMA, *Mixed Reality: Future Dreams Seen at the Border between Real and Virtual Worlds*, IEEE Computer Graphics and Applications, 21(6), 2001, pp. 64–70.

[76] D. TERZOPOULOS and K. FLEISCHER, *Deformable Models*, The Visual Computer, 4(6), 1988, pp. 306–331.

[77] P. M. THOMPSON and A. W. TOGA, *Detection, Visualization and Animation of Abnormal Anatomic Structure with a Deformable Probabilistic Brain Atlas Based on Random Vector Field Transformations*, Medical Image Analysis, 1(4), July 1996, pp. 271–294.

[78] V. TORRE and T. A. POGGIO, *On Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8, 1986, pp. 147–163.

[79] J. TRAUB, *Design of an Intra-Operational Augmented Reality Enhanced Navigation Tool for Port Placement and Tracking of Instruments in Robotically Assisted Minimally Invasive Cardiovascular Surgery*, Master's thesis, Technische Universität München, Department of Computer Science, November 2003.

[80] 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.

[81] J. T. VALVODA, *Virtual Reality Meets Medicine - A state of the art review of methods and applications in medical Virtual Reality*, October 2001.

[82] P. P. VARGA, Z. HOFFER, and I. BORS, *Computer-assisted percutaneous transiliac approach to tumorous malformation of the sacrum*, Computer Aided Surgery, 6(4), 2001, pp. 212–216. to be added.

[83] VARIOUS, *Special Issue: Medical Robotics*, IEEE Transactions on Robotics and Automation, 19(5), October 2003.

[84] A. M. VAZE, *Robotic Laparoscopic Surgery : A Comparison of the Da Vinci and Zeus Systems*, Bombay Hospital Journal, 44(2), April 2002.

[85] D. VILARIÑO, V. BREA, D. CABELLO, and J. PARDO, *Discrete-time CNN for Image Segmentation by Active Contours*, Pattern Recognition Letters, 19(8), 1998, pp. 721–734.

[86] S. K. WARFIELD, A. NABAVI, T. BUTZ, K. TUNCALI, S. G. SILVERMAN, P. BLACK, F. A. JOLESZ, and R. KIKINIS, *Intraoperative Segmentation and Nonrigid Registration for Image Guided Therapy*, in MICCAI - Medical Image Computing and Computer-Assisted Intervention, 2000, pp. 176–185.

[87] W. M. WELLS III, W. E. L. GRIMSON, R. KIKINIS, and F. A. JOLESZ, *Adaptive Segmentation of MRI Data*, in Computer Vision, Virtual Reality and Robotics in Medicine, N. Ayache, ed., Springer-Verlag, 1995.

[88] J. WERNECKE, *The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*, Addison Wesley, 1994.

[89] S. ZUCKER, *Region Growing: Childhood and Adolescence*, Computer Graphics and Image Processing (CGIP), 5, 1976, pp. 382–399.

[90] S. ZUCKER and R. HUMMEL, *A 3-Dimensional Edge Operator*, Proc. IEEE Conf. on Pattern Recognition and Image Proocessing, July 1977.