Abstract

Intra-operative Magnetic Resonance Imaging is a new modality for image-guided therapy, and Augmented Reality (AR) is an important emerging technology in this field. AR enables the development of tools which can be applied both pre-operatively and intra-operatively, thus helping users to see into the body, through organs and visualize the relevant parts useful for a specific procedure.

The work presented in this paper aims at solving several problems in order to develop an Augmented Reality system for real-life surgery in an MR environment. Specifically, ways of correctly registering 3D-imagery with the real world is the major problem of both Augmented Reality and this thesis. Emphasis is put on the static registration problem. Subproblems of this include: calibrating a video-see-through Head Mounted Display (HMD) entirely in Augmented Reality, registering a virtual object on a patient by placing a set of points on both the virtual object and patient, and calculating the transformation needed in order for two overlapping tracking systems to deliver tracking signals in the same coordinate system. Additionally, problems and solutions related to the visualization of volume data and internal organs are presented: Specifically, how to view virtual organs as if they were residing inside the body of a patient through a cut, thought no surgical opening of the body has been performed, and the visualization and manipulation of a volume transfer function in a real-time Augmented Reality setting.

Implementations use the Studierstube and OpenTracker software frameworks for visualization and abstraction of tracking devices respectively. OpenCV, a computer vision library, is used for image processing and calibraton together with an implementation of Tsai's calibration method by Reg Willson. The Augmented Reality based calibration implementation uses two different calibration methods, referred to in litterature as Zhang and Tsai camera calibration, for calibrating the intrinsic and extrinsic camera parameters respectively. Registering virtual-real objects and overlapping tracking systems is performed using a simplified version of the Iterative Closest Point (ICP) procedure solving a problem commonly referred to as the absolute orientation problem. The virtual-cut implementation works by projecting a rendered texture of a virtual organ and mapping this to a mesh representation of a cut which is placed on the patient in Augmented Reality. The volume transfer functions are implemented as Catmull-Rom curves, and have control points which are movable in Augmented Reality. Histograms represent transfer functions as well as distribution of volume in-

tensities.

Results show that the Augmented Reality based camera calibration procedure suffers from inaccuracies in the sampling of points for extrinsic camera calibration due to the dynamics present when wearing an HMD and holding a tracked pen. This type of calibration should occur by sampling statically and averaging over several samples to reduce noise. The virtual-real and overlapping tracking systems are also sensitive to sampling, and care has to be taken in order to do this accurately. The virtual-cut technique has been shown to increase the feeling of a virtual object residing within the body of a patient, and the volume transfer function became easier to use after implementing the histogram visualization, reducing the time needed to set up a transfer function.

There are many issues which need to be solved in order to set up a useful medical Augmented Reality implementation. This thesis attempts to illustrate some of these problems, and introduces solutions to a few. Further developments are needed in order to bring the results from this paper into a clinical setting, but the possibilities are many if such an integration is achieved.

Acknowledgements

The work presented in this thesis was conducted at the Interventional Center, Rikshospitalet, Oslo, Norway during spring semester 2005.

First and foremost, a lot of thanks go to Eigil Samset who allowed me to do my Master's thesis at the Interventional Center. He has been a very helpful guide throughout the semester, and also incouraged me to come to Graz on a meeting of the ARIS*ER project. Thanks go to Ole Jakob Elle whom, with his nice temper, made the stay a pleasant one. Petter Risholm deserves credit for always helping to find solutions to problems and being an all the way intelligent and outgoing guy. Many thanks go to the other students doing their work at the Interventional Center. I had a good time relaxing from my studies while playing table tennis with and without a table, distorting pictures of eachother and just talking. Of course, not to forget, my main mentor, Richard Blake, whom deserves many thanks for being my mentor both this and the previous semester.

I would also like to thank my family and friends for always being there for me and encouraging me in my efforts.

Contents

Preface

There are many ways in which people can experience the world. Some sense their way through their daily lives using just the touch of their fingers or the sounds which surround them. Others may look at the world in black and white, while a few sense sounds or touch by color patterns displayed by their brain. Usually people sense the world using a multitude of natural sensors available to the human body. Still, even more relevant information and knowledge can be gathered from the world around us and it can be augmented to degrees which makes the augmentation a part of the world as we perceive it. This augmented experience could feel as real to the person as the other sensory input from the real world. Such an experience is what Augmented Reality is all about, but it is a difficult task to achieve. For this to work, the augmentation has to conincide with the real world in various ways. The problems involved with this registration is the main problem of this thesis.

This Master's thesis was written at the end of a computer science study at the Norwegian University of Science and Technology, and conducted at the Interventional Center, Rikshospitalet, Oslo, Norway.

Chapter 1

Introduction

1.1 Background

Computers have in recent years been introduced to the operating rooms, and tendency is to have them play an increasingly important role during surgery. The systems, which help the surgeon during a procedure, are often called planning systems due to their nature. Image guided surgery, image guided therapy, computer integrated surgery, image guided operating robot and augmented reality in surgery are all titles that represent the same emerging technology which is on the way to revolutionize the field of medicine.[1]

Minimally Invasive Surgery (MIS) is becoming a widespread trend in modern medicine. This developing technology is a tool which inflicts less damage on tissues, allowing for faster recovery time, and is generally more cost-effective for hospitals. But even though these are great possibilities, there are also problems involved with this field of medicine. The limiting factors of MIS are[1]:

- Immature and unreliable tools for real-time 3D navigation.
- MIS is developed by separate groups, causing fragmented research. which prevents rapid development of the technology.
- Lack of cross-disiplinary researchers having insight into many important fields of research.

These problems have become the main focus of an EU-project named **ARIS*ER**, coordinated by the University of Oslo.[1]

ARIS*ER is a Marie Curie Research Network, and the name is an abbreviation which can be expanded to: "Augmented Reality in Surgery". The project spans over a time-period of 48 months, starting late 2004. It is conducted by a multi-national consortium consisting of several cross-disciplinary groups with different interests and focus areas.[1]. The goals of the project are many, but overall, the main goals are to educate researchers in the field of Augmented Reality so that the technology can be

more easily adopted into real-life surgery. Thus leading the way for better and safer healthcare for european citizens. Another major goal is to develop fast, reliable and user-friendly software which can be used within Minimally Invasive Therapy (MIT). MIT is a compound of Minimally Invasive Surgery, Image Guided Surgery and Interventional Radiology[2].

1.2 Problem formulation

Intra-operative MRI is a new modality for image-guided therapy. The Interventional Center has an open MR scanner, where it is possible to perform surgery on a patient, in the same time as imaging is conducted. A stereo-display system is installed with video-see-through goggles. The project aims at developing the software to drive these goggles, so that images from externally mounted video cameras can be co-registered with MR images and tracking data creating an augmented reality for the surgeon.

This Master's thesis, will address several issues in order to develop an Augmented Reality system for use in real-life surgery in an MR-environment. Specifically, ways of correctly registering 3D-imagery with the real world is the major problem of both Augmented Reality and this thesis. Emphasis will be put on the static registration problem. Some weight will also be left on dynamic registration if needed.

The thesis will be conducted at the Interventional center, Rikshospitalet, Oslo, Norway.

1.3 Clinical background and motivation

At the Interventional Center, there is an open MR-scanner which is used for diagnosing diseases, non-conventional positioning of patients in a static or dynamic study, and performing surgery. The most beneficial property of such a system is that imaging can be conducted while the surgeon still has access to the patient during the procedure[3]. However, the full potential of the machine is not utilized without the surgeon being able to see a good and intuitive reconstruction of the data. Additionally, the data should located in an optimal way relative to the surgeon and the patient. The natural way for human beings to experience and interact with data, is through our senses. The more the computer is able to interface with the human senses, the better we will be able to interact with the information presented. To cite the wise words of my mentor at the Interventional Center, Eigil Samset[3]:

"... Surgical- and interventional procedures are much more demanding, both technologically and practically. The key feature for successful MRI-guided interventions is a well-designed interactive image guidance system. For each procedure special care has to be taken in the design of instruments, the choice of image acquisition technique, patient positioning, positioning devices, image guidance systems and human-computer interface.".

This citation highlights the need for a good and interactive guidance system for MR-imaging. By making such a system, the surgeon should be able to operate more accurately, and the overall invasiveness of procedures can be reduced, thus reducing the recovery time and expenses related to the procedure. But, of course, only being presented with data from the MR-scanner does not necessarily help the surgeon perform better. The data must be presented in an easily understandable fashion and highlight the problem at hand. Specifically, the system should ease the hand-eye coordination problem often experienced during computer-aided procedures due to the data being visualized indepenently of the patient location.

The conventional image guidance systems for intra-operative imaging are not really designed with good human-computer interfaces in mind. Instead, the images are often conformant with the old radiological way of viewing the body through predefined planes. Such a non-intuitive mapping between what the eye sees and the body does is a skill which takes much time to learn, and might not function satisfactory in stressfull situations.[3].

1.4 The Interventional Center

The Interventional Center is a research and development department located at Rikshospitalet University Hospital. The center is a cross-disciplinary group of people working to further the development and usage of minimally invasive interventions and patient treatment. The work presented in this thesis was conducted at the Interventional Center during spring semester 2005.

1.5 Chapter organization

This Master's thesis continues the work presented by the author in [5]; a project thesis in the subject *TDT4715 Algorithm construction and visualization, depth study* conducted autumn semester 2004. The work presented in the current document deals with the calibration and registration problems found in Augmented Reality in addition to visualization and interaction issues. All topics are covered with real-life medical applications in mind.

- **Chapter 1** gives an introduction to the thesis and presents relevant background information.

- **Chapter 2** introduces various topics and background information including Augmented Reality, volume graphics, MR-imaging, Minimally Invasive Surgery, Human Computer Interaction, and more.

- **Chapter 3** presents a short summary of the results achieved during the semester.

- **Chapter 4** describes the software and hardware development environments and resources available at the Interventional Center.
- **Chapter 5** discusses and elaborates on the registration problem of Augmented Reality and illustrates the steps towards possible solutions to different problem areas.
- **Chapter 6** aims at introducing and solving problems related to the visual appearance of Augmented Reality applications in a medical setting.
- **Chapter 7** discusses the work presented in an overall manner.
- **Chapter 8** concludes the thesis.
- **Chapter 9** presents possible future work.

Chapter 2

Background

2.1 Augmented Reality

2.1.1 What is Augmented Reality

[1] Augmented Reality (AR) is a special form of Virtual Reality, and makes use of Head Mounted Displays (HMDs), or modified optical instruments to superimpose meaningful virtual images onto the user's view of the real world[6]. Virtual Reality (or Virtual Environment) is a completely immersive environment in which the user cannot see the real world around him. In contrast, Augmented Reality superimposes a virtual world on top of the real one, augmenting the world we see with virtual objects[6][9]. Augmented Reality is defined by Azuma[10] as:

- Combines real and virtual
- Interactive in real time
- Registered in 3D

AR is actually a subgroup of something called Mixed Reality[11]; A term that spans the continuum between Virtual Reality and actual reality, and includes Augmented Reality, Augmented Virtuality, and other mixed configurations. Augmented Virtuality being a system which provides the user with additional sensory input through e.g. smell, wind blowing in the face, etc. There are two main groups of Augmented Reality systems.

- Systems using optical-see-through HMDs
- Systems using video-see-through HMDs

There are, however, AR systems using neither of these technologies. Sometimes only a video projector or a portable screen is used instead of an HMD. An interesting project

[1] Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

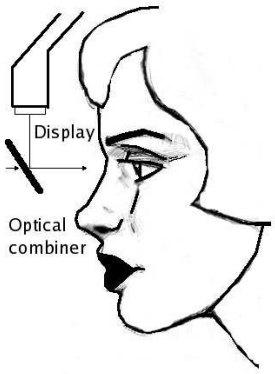

Figure 2.1: *Optical-see-through HMD*

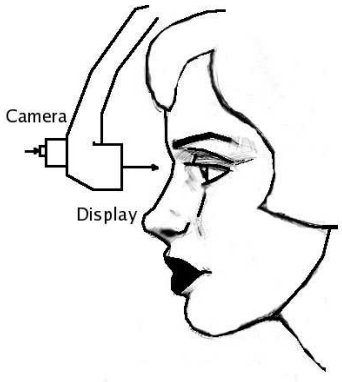

Figure 2.2: *Video-see-through HMD*

at the Bauhaus University, Weimar[12], uses consumer cell phones with digital cameras embedded as an Augmented Reality platform.[11]

Normal Virtual Reality HMDs do not have the capacity of viewing the real world in addition to the virtual. This is because of the lack of sensors/cameras to relay pictures of the real world to the displays on the HMD.

Optical-see-through HMDs (figure 2.1) have semi-transparent optical combiners that let some light enter from the real world, but also projects an image of a virtual world on the optical combiner, which reflects some of the light towards the user's eyes, thereby making it look like virtual objects are placed in the real world.

Video-see-through HMDs (figures 2.2 and 2.3) use cameras mounted on them to collect images of the real world. These images are sent to a computer where they are combined with generated images of the virtual world. The modified images are then sent back to the HMD and shown on the displays.

If one compares optical-see-through and video-see-through approaches to Augmented Reality, there are advantages and disadvantages to both. Optical-see-through

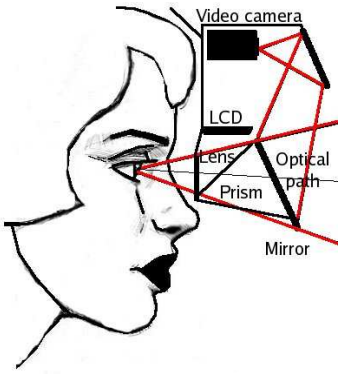

Figure 2.3: *An HMD which attempts to minimize the parallax error by making sure the images captured by the video cameras are captured as if the cameras were at the true eye locations[30]*

HMDs have no delay when it comes to the imaging of the real world. The only images to worry about, are those generated by the computer. Video-see-through approaches make it possible to do further processing of images before they are sent back to the user. Azuma[13] notes that medical applications mostly use the video-see-through approach probably because of the flexibility in blending real and virtual, and for the additional registration procedures available.

Even though Augmented Reality may seem new and interesting, it is by far a new thought in the history of the computers. Already back in the 1960s, Sutherland and Sproul experiemented with AR. Their idea was that the more natural the interaction between human and computer becomes, the more useful the computer will be. By 1968, the first see-through HMD was a reality, and combined with a 3D tracking system, a user was able to interact with the system by moving within the sensor area of the tracker. The problem of the time was that computers were not fast enough to process the data in a useful manner[14]. Since then, hardware has come a long way, but there are still many problems not yet solved to a satisfactory level.

2.1.2 Uses of Augmented Reality

[2] Possible uses of Augmented Reality is practially endless, but some of them are worth mentioning. Medical visualization, maintenance and repair of complex equipment, annotation, path planning, entertainment and military aircraft navigation and targeting are just some of the possibilities[13][15]. The military is interested in AR because of its ability to display information, the most important weapon in modern warfare, and thus spends money on research and equipment. A collaborative augmented environment could also be used for problem solving, where people can see and interact with

[2]Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

the same augmented environment.

Within **medicine**, the interest in the field of Augmented Reality has grown the last few years. This is probably because of the possibilities of training doctors and doing more accurate surgery without opening patients wide open. The problem minimally invasive surgery deals with, is how to augment the view of the surgeons, giving them a view of the internals of a patient, without actually creating a big opening in the patient's body[13]. Usually, the surgeon will get video signals from camera(s) inside the patient or from an MR or CT-scanner, and the information is displayed on a video screen in the operating room, but research has been conducted on how to augment the view of the surgeon and display information on the body of the patient, giving the surgeon "X-ray vision". A similar approach has been implemented in a system for examining the inside of the human body using ultrasound. It is often very difficult to get an accurate understanding of what is seen on the screen when doing ultrasound of e.g. the heart[16]. Augmenting the view of the examinor could help him get a better understanding of what is actually going on under the skin, thus being able to make a more accurate diagnosis. Example uses of Augmented Reality within medicine include viewing a 3D fetus inside the womb of a woman as if a virtual cut was made, displaying information for guiding a biopsy needle through a breast tumor, and doing laparoscopic surgery while images from small cameras are relayed to an HMD[17]. A concrete example includes the Mederpa[18] system, which is a medical AR system for minimally invasive interventions that is being set up to support areas such as heart surgery, radiation oncology and respiratory medicine.

This thesis builds on the approach of previously mentioned projects, and deals with the visualization of volume data from an open magnetic resonance (MR) scanner on top of a patient, giving the surgeon more accurate information of what is going on under the skin.

For an excellent and more comprehensive introduction to Augmented Reality, the interested reader is referred to papers [13] and [15]. For more on different Augmented Reality implementations, the reader is referred to [19], a paper which compares and evaluates some of these.

2.1.3 The registration problem

Augmented Reality enhances the world we see with virtual objects. The placement of the virtual objects relative to the real ones, referred to as the **registration**, is of major importance in order to make the combined world believeable. Registering the systems and data in an Augmented Reality implementation for use in surgery is not an easy task. Several factors have to be considered. One such factor is to ensure the registration is as correct as possible in order to perform accurate and safe medical procedures. Another, and often colliding, factor is the speed in which the registration needs to be performed. Latency could reduce or destroy the feeling of the virtual and real worlds existing together. This brings us to the definition of the registration problem: "Making the real world and the virtual world appear to exist together, not as

distinct universes, but as one unit". Unfortunately, registering virtual objects with the real world is a hard problem to solve. One of the reasons for this, is the fact that human beings are very good at detecting small misregistrations. Even just a few pixels wrong is easily perceived.[20][22]

Registration errors fall into two main categories: **static and dynamic**. Dynamic registration error is what occurs due to delays within the system when the user moves. It is often the source of the largest registration error. Static registration error is the mismatch experienced when the user stands completely still, and is the basis for the dynamic registration. Both of these issues are extremely important within the domain of Augmented Reality. In fact, registration is also a problem in Virtual Reality, but is not as obvious because there is no mixing of the real world.[10][9]

The static camera calibration process can be divided into two main parts:

- Calibrating the intrinsic camera parameters
- Calibrating the extrinsic camera parameters

The intrinsic camera parameters are those which only are dependent on the internal camera specific attributes. These include camera optics, distortions, dimensions, etc. The extrinsic camera parameters are those which depend on the placement and orientation of the camera in the world coordinate system.

The terms **registration** and **calibration** are often used synonymously. This is not completely accurate. In this thesis, the word **registration** will be used as being the alignment of virtual objects and their associated real counterparts[21]. While **calibration** will mean the process of obtaning the registration.

2.2 Volume Visualization and Magnetic Resonance Imaging

2.2.1 Volume visualization

[3] Volume data is a set of 3D entities that may hold information inside them, and volume visualization is the process of extracting information from volume data and displaying this information using a computer. The volume data is often gathered by sampling using e.g. Magnetic Resonance Imaging (MRI), or Computer Tomography (CT), but could also be generated by other sampling, simulation or modeling techniques.[23]

Volume visualization is concerned with synthesis, modeling, manipulation, and rendering of volumetric objects, stored in a volume buffer of voxels, and most often uses sampled or computed datasets, while volume graphics (as a distinction from volume visualization) usually is concerned with modeled geometric scenes.[23] The process of creating 3D voxel representations that best describe geometric objects is

[3] Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

called **voxelization** or 3D scan conversion. Basically a discrete 3D database is created that represents continuous objects.[23]

The volume data is a set, S, of values, v, at a location (x, y, z). The value can be just a binary value, or a more complex value, like a vector. The data could be sampled at random locations in space, but most usually, the samples are taken with regular spacing in a limited domain. When samplings are taken along the three axes with regular spacing, the sampling is called **isotropic**. If the sampling along each axis is regular, but have different sampling distances along the three axes, the sampling is called **anisotropic**. In these cases, the samples are located on a **regular grid** which is called a **volume buffer** (3D array). There are also other data structures used to represent the 3D volume data[23]:

- **Rectilinear grid** - the cells are axis-aligned, but the spacing between the grid elements is irregular. Often called computational space.
- **Curvilinear grid** - If a rectilinear grid is non-linearly transformed while keeping its topology, it becomes curvilinear. Often called physical space.
- **Unstructured grid** - For all other grids. These are grids whose connectivity have to be specified explicitly because of the cells having arbitrary shapes.

The set S of points with values can have an associated function $f(x, y, z)$ that also is defined for regions not having any associated value. If (x, y, z) is a point in space with a value, and there exist neighboring points also having values, f defines the regions between the points. If f is a *zero-order function*, the nearest defined point value will be returned by the function. When the grid is regular, the spacing between each point is the same, and the function defines square regions with equal volumes. These square regions are called **voxels**.[23]

Many techniques have been developed to visualize volume data, and several of these are based on already known primitives such as triangles. Surfaces are then extracted from the volume data as approximations, and much of the existing volume information is lost. Surface techniques are much easier and faster to render in many circumstances, but on the downside, they do not provide the massive amount of information available from the volume representation.[23]

Direct volume rendering (DVR), is the rendering of the entire volume data at once without the extraction of iso-surfaces or other surface approximations. This way of displaying volume data has been increasingly popular especially within medicine, where CT and MR data are collected and used as a basis for e.g. surgery. After the introduction of hardware accelerated 3D textures and GPU programming, this way of visualizing volume data gets a great deal of focus. There are currently two different hardware accelerated volume rendering techniques being used; object aligned, and view-aligned. The object-aligned approach has the down side that three copies of the volume has to be sent to the graphics hardware - one for each major axis. View-aligned

volume rendering does not have this problem as the graphics hardware does the interpolation of the volume data set so it always is sliced with faces towards the viewer. [24] One of the important features of volume representation is the ability to represent the inner structure of objects. Volume visualization has advantages over surface graphics by being viewpoint independent, insensitive to scene and object complexity, and lending itself to the realization of block operations, CSG modeling and hierarchical representation. [23]

Transfer function specification is also known as intensity classification, and plays an important role within volume visualization[24]. What transfer function specification does, is creating a mapping between the intensity values of the volume, and color and opacity values. Each intensity in the volume data gets a specific color and a specific opacity defined by the transfer function. Thus, the volume data will be rendered with different transparency and color depending on the transfer function.

Volume scenes often require a substantial amount of memory. A scene using a 512^3 grid with two bytes per voxel will need 256MB of memory for storing all the data. This has not been possible until recently, but as the cost of memory falls, and the amount available in graphics cards increases, volume visualization/graphics will be more and more widespread in the years to come.

2.2.2 Magnetic resonance imaging

[4] Magnetic Resonance Imaging (MRI) was introduced in the early 1980s. Since then, the number of medical procedures taking advantage of the technology has been ever increasing. MRI has among other things been very important in diagnosing problems within the human body.[3]

Magnetic Resonance Imaging is a process where an electromagnetic field is generated to create an image of the internals of objects. MR uses the principles of **nuclear magnetic resonance** to view structures inside the human body in a non-invasive way. Protons within the nucleus of atoms contain electrical charge, and while spinning create a magnetic field. When these are spinning in a strong magnetic field, they align themselves with the field in a parallel or antiparallel way. When radiofrequency radiation is sent towards the atoms in this configurations, some protons will absorb radiation and momentarily be shifted out of alignment or resonate. The radio frequency the different atoms absorb uniquely identifies the type of atom and its chemical environment (chemical shift). As the external radiofrequency radiation is turned off, the protons will return to their normal alignment in the magnetic field and release a small amount of radiation at exactly the same frequency as caused them to shift out of alignment. This radiation is picked up by sensors mounted on the scanner. One of the limitations of MR scanners is that they only can be used on magnetic nuclei. This is the reason why MR imaging usually is targeted at hydrogen nuclei since they make up about one third of the atoms in the human body, and the resulting images generated give a map

[4]Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

over the distribution of water and lipids within the tissues.[25]

MR scanners can be found with field strengths varying from 0.02T to 3.0T (tesla). Different types of magnets can be used for the scanners. The three possible types are **permanent**, **resistive** and **superconducting**. The permanent magnets use magnetic materials which always produce magnetic fields. They are inexpensive, but are very sensitive to temperature, and are huge and heavy. Resistive magnets are made by coils wrapped around a bore, and electricity is run through the wires, thus generating an electric field. These magnets are relatively cheap to make, but require huge amounts of electricity to generate strong magnetic fields, making them quite expensive to use. The most used magnets in MR scanners are superconducting magnets. These magnets are very much equal to resistive magnets, but are supercooled using liquid helium to temperatures close to absolute zero. The cooling decreases the power consumption and makes the scanner more economic to use.[25]

In practice, a human is placed on a bore inside the MR scanner and a radiofrequency coil or antenna is placed over the part of the body to be scanned. The magnetic field generated by the magnets create distinct planes which all have different resonance conditions. This makes it possible to collect data directly from a specific slice without having to move the patient. The data collected by the MR scanner is then sent to a computer for calculations before cross-sectional images of the body are created.[25]

The design of MR scanners to be used while doing surgery is a balance between different requirements. On one side, the **accessibility** of the patient, and on the other side, the **strength** of the magnetic field, which has strong impact on image quality.[25]

The use of real-time MR imaging to facilitate minimally invasive therapeutic procedures is referred to as **interventional MR imaging** whereas guidance to facilitate open surgical procedures is usually referred to as **intraoperative MR imaging**. Collectively these two procedures fall under the acronym **IMRI**. IMRI falls into three main categories depending on how the patient is accessed during the procedures[25]:

- Moving the patient to the magnet
- Moving the magnet to the patient
- Operating within the magnet

There does not exist a lot of IMRI scanners around the world. [25] reports that just 40 IMRI systems were installed around the world at the beginning of 2003.

2.3 Minimally Invasive Surgery

2.3.1 What is Minimally Invasive Surgery

Minimally Invasive Surgery (MIS) is a procedure carried out by entering the body through small incisions or natural body cavities. The procedure attempts to minimize the trauma inflicted on the patient[7]. Other beneficial factors of MIS include[4]:

- Reduced risk of infection
- Reduced cost
- Reduced recovery time

All of these factors make MIS a promising and widely researched topic in hospitals and research laboratories around the world. However, there are problems related to this type of surgery. In short, the surgeon suffers under these procedures[26]:

- Using smaller incisions, the doctor has less space to work with
- The dexterity of the surgeon is greatly reduced
- The surgeon loses the feeling of touch within the body; this because most instruments do not support tactile feedback

Hence, the surgeon will not be able to fully utilize the human senses[26]. The loss of tactile feedback is perhaps the biggest disadvantage related to MIS. The sense of touch is important for locating hidden anatomical structures and evaluating tissue properties. For example, locating a tumor is often done by feeling the denser structures it has relative to an organ[26]. Advances in technology have reduced some of these problems, and will help resolve many more in the future.[27]

Minimally invasive procedures include: endoscopy, laparoscopic surgery, cryosurgery, angioplasty, stereotactic surgery, and many more[7].

2.3.2 A historical perspective

The history of Minimally Invasive Surgery is relatively long. Already as early as 1804, the "Lichtleiter" was introduced by Philipp Bozzini. The initial developments did, however, only rely on natural access through body openings. As time progressed, the technology remained nearly unchanged. The early developments in medical visualization during the 1990s was restricted by the hardware available; keeping computer aided surgery on a research level only[27]. Only in very recent years has the technology matured to a level taking MIS out of research labs and into the operating rooms.[28] The introduction of miniature cameras, fiberoptic cables and new instruments made this transition possible.[7]

For more on the current status and problems related to Minimally Invasive Surgery, the interested reader is referred to [27].

2.4 Human Computer Interaction

2.4.1 What is Human Computer Interaction

Human Computer Interaction (HCI) is defined by Rogers as being[31]:

- The study of relationships between people and computers/computer-mediated information.
- The design, development and evaluation of models, systems, techniques and applications from a human-centered perspective.

Basically, the field of HCI is concerned with how to improve the interaction between human beings and computers by making the computers easier to use.[8]

Research has been conducted on the use of human-computer interfaces, and much is known about the dynamics of human-computer interaction for different kinds of people. However, the multitude of interaction methodologies that exist make it impossible to deliver a "perfect" interactive system.[32]

When programmers develop software for a computer, they have to consider the nature of the computer. But the nature and needs of the computer are often completely different from the nature and needs of the human who will eventually use it. To make things worse, programming languages do not really build around the concept of human-computer interaction, thus the programmer has to fight with the languages to implement this, and often ends up creating an interface which is full of functionality and works great for other engineers, but not so good for the end user. A well-designed human computer interaction system needs to have high usability for the end users. More on this in the following section.[31]

2.4.2 Usability and Usability Engineering

Usability is a major part of Human Computer Interaction. A good interface needs to have high usability, but what is it really? The ISO9241 standard defines usability as[31]: *"is a measure of the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in a particular environment"*. Usability is a combination of factors which work together in creating a total user experience with the system. Some of the usability factors are[33]:

- How easy it is to learn
- How effective it is to use
- How memorable the system is
- How often errors occur and how severe they are
- How satisfied the user is

The goal of software designed around a user, is to optimize the usability of the software. This could, to some degrees, be done by **usability engineering**[31]. Usability engineering is a methodical approach aimed at developing a user-centered product that works for the users, and includes a set of methods which, applied at the right time, can aid the development of user-centered software. Some of the methods include[31]:

- Gathering requirements
- Designing
- Developing prototypes
- Testing prototypes
- Evaluating design alternatives
- Analyzing usability problems
- Proposing solutions
- Testing the interface with users

Traditional software development has usually relied on the well-known waterfall model, where software is developed in stages and cycles occur if needed. User-centered design is different from this traditional model in several respects. User-centered development is[31]:

- **User-centered** (not data-centered). Users are a part of the whole process as much as possible
- **Interdisciplinary**. User-centered software is inspired from several different fields, including: art, psychology, computer science, etc.
- The software has to be **tested** rigorously and **revised** as needed, especially before the final implementation.

The user-centered software life-cycle can be seen in figure (2.4). For more theoretical and concise background on HCI and usability, the interested reader is referred to [31] and [33].

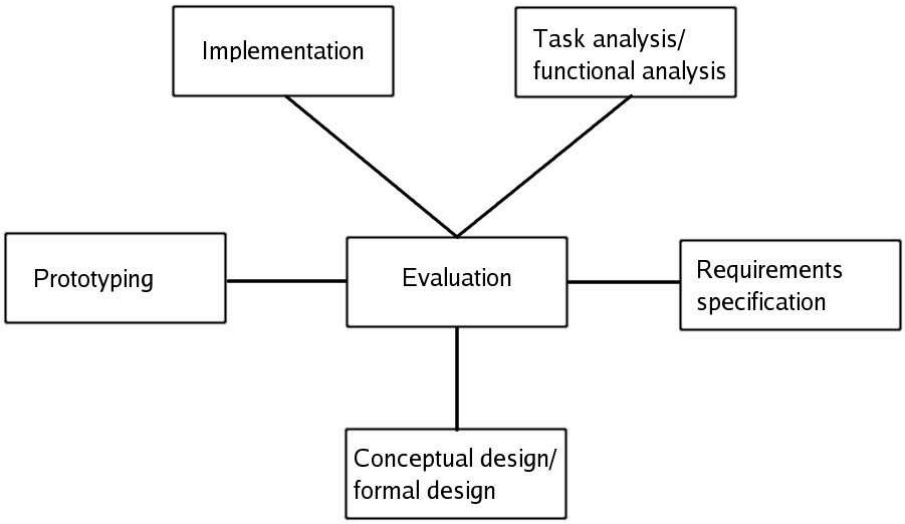

Figure 2.4: *Star life cycle - Conceptual development model for user-centered applications. Order is not as important as going through all stages of development*

Chapter 3

Summary of Results

Work conducted during the thesis has not focused on a single problem alone. Several subproblems within registration and calibration have been investigated and solutions to some of these proposed. Additionally, topics within visualization of data have been investigated and possible solutions presented. The following list gives a short overview of the results achieved during spring semester 2005:

- A calibration application for video-see-through Head Mounted Displays (HMDs) in the Studierstube Augmented Reality framework. This was the main problem for the thesis.
- A method for registering a virtual object on a real patient by point-point correspondence.
- A method to automatically register two tracking systems with eachother after sampling a moving tracked device in both tracker coordinate systems.
- A virtual cut technique which makes it possible to view a virtual object as if it were placed inside a cut on the patient, though the patient is not surgically opened.
- A way of performing volume transfer function manipulation in Augmented Reality. The previous approach has been improved upon, and additional features include histogram visualization.

All results play their part in a medical Augmented Reality system: Having calibrated the cameras is a prerequisite for convincingly blending real and virtual realities. Being able to register virtual objects relative to real ones is needed for matching virtual and real scenes and objects with eachother. Having placed a virtual cut on a patient, surgery could be performed through minimally invasive procedures using computer visualized images of volume data acquired from an MR-scanner as a reference. The visibility and color properties of the volume could be manipulated in real-time using a transfer function in order to bring out areas of interest. All of this could be tracked by

several heterogeneous tracking systems where signals are transformed into a common reference coordinate system.

Chapter 4

Development Environment

4.1 Hardware

4.1.1 Signa SP MR-scanner

The MR-scanner is a $0.5T$ Signa SP (Special Procedures) scanner by General Electric Medical Systems. The scanner is an **open** MR-scanner, meaning it consists of two donut-shaped rings with a $60cm$ gap between. In this gap, imaging can be conducted in an open fashion, making surgery possible during scanning. The scanner has RF-shielded monitors, and an integrated optical FlashPoint 5000 tracking system is located between the two bores, above the patient being scanned. The system consists of three infrared cameras tracking the movement of instruments when they enter the tracking area of the cameras. The MR-scanner can be seen in figure (4.1).[3]

4.1.2 CardioView Head Mounted Display

The Head Mounted Display (HMD) used for testing was a CardioView HMD, running at a resolution of 640×480. This is the minimum resolution acceptable for such HMDs, and a better HMD with a wider field of view is desired, but such an HMD should not be too big. For the wearer to feel comfortable and accept the virtual world displayed, the HMD has to be light and easy to wear. The CardioView HMD has separate channels for right and left eye, so synchronization of graphics cards was not necessary when connecting two separate computers. A slightly modified version of the CardioView HMD is found in figure (4.4).

4.1.3 HMD in MR-scanner environment

The HMD specially developed for the Interventional Center and colocated with the MR-scanner, is currently of unknown type. A normal HMD was modified slightly to tolerate the harsh MR-environment without breaking. This HMD will be used for implementing the system described in this thesis in a clinical setting.

Figure 4.1: *The 0.5T Signa SP Open MR-scanner located at the Interventional Center[3]*

4.1.4 Unibrain Fire-i

The digital cameras mounted on the front of the CardioView HMD are of the type Unibrain Fire-i - firewire cameras. They have much higher framerate than corresponding USB cameras, producing about 30 frames per second at 640×480. Even though the cameras are able to operate at this speed, there still exists latency in the grabbing system, causing images to appear a little later than they actually were fetched from the cameras. This could cause undesired effects in an Augmented Reality system.

4.1.5 FlashPoint 3000 and 5000

The tracking system used for testing was a FlashPoint 3000 system, and the system located in the MR-lab is a FlashPoint 5000 system. Both of which were manufactured by Image Guided Technologies, Inc. These are optical tracking systems using infrared diodes and cameras. There are three cameras which together find the spatial location of the tracked units in the tracker coordinate system. One of these systems is illustrated in figure (4.2).

Some problems are encountered when using optical tracking systems. The major problem is the line-of-sight needed from the cameras to the light-emitting diodes on the tracker. If the diodes are not visible from the cameras, tracking will stop, and

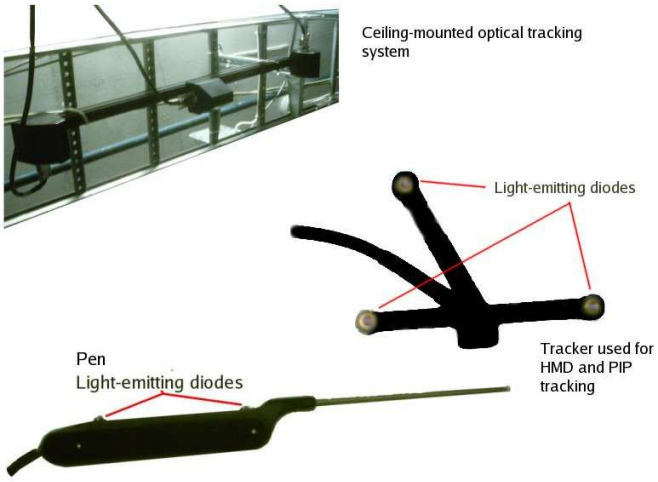

Figure 4.2: *The tracking system used for testing at the Interventional Center. This is a ceiling-mounted optical tracking system. A tracked pen and additional tracked tools can be attached and mounted on e.g. the PIP or HMD. A problem which became apparent with the tracked pen, having all light- emitting diodes on a line, is the fact that rotation around one of the axes will not generate correctly oriented tracking signals.*

registration will suffer. This can easily be the case if e.g. a hand or person come between the cameras and the tracker units. The current system also has a pretty small working area. A new optical tracking system will be installed at the Interventional Center in the future, which has a much greater working area and performs better than the current systems.

4.1.6 Homemade device for holding cameras and optical tracker

To be able to use the CardioView HMD with the firewire cameras, the cameras had to be attached in some way. For this purpose, a holding mechanism was constructed and fixed to the HMD. In addition to the cameras, a mechanism for attaching a head tracker was added to the device. A schematic view of the holding mechanism is seen in figure (4.3), and an image of the CardioView HMD with the holding mechanism in place is seen in figure (4.4).

4.1.7 System overview

The different hardware parts described in the previous sections, have been composited into one unified working system. Several computers are connected to eachother in a network configuration. These computers have different roles in the network. One or two are used to render to a stereo display system worn by the user. An other computer

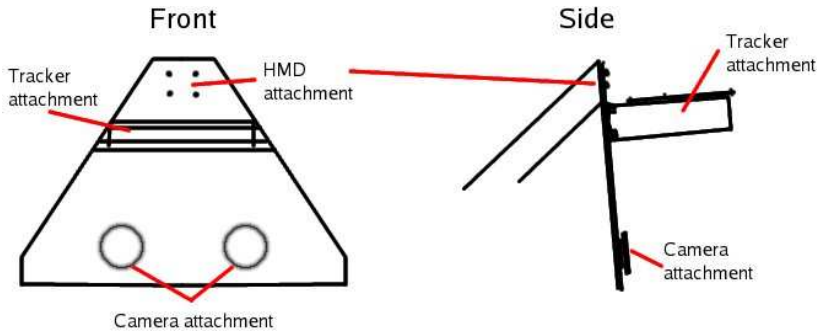

Figure 4.3: *Schematic view of device developed to hold cameras and tracker unit on the CardioView HMD. This device is mounted in front of the displays of the HMD, attempting to reduce the parallax error which is bound to appear in this setup.*

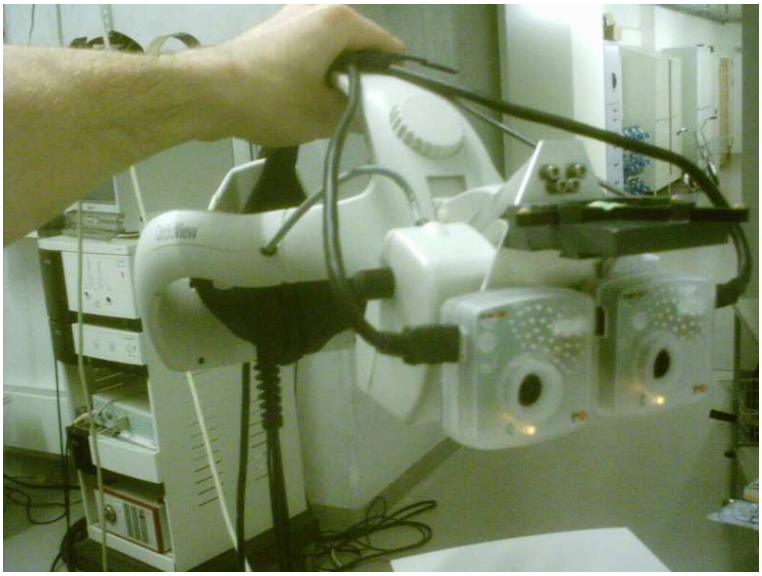

Figure 4.4: *An Augmented Reality enhanced CardioView HMD. The cameras are mounted upside down in front of the displays of the HMD to reduce the parallax error. The tracker is located above the cameras to be easily seen by the tracking system located in the ceiling.*

is connected to the tracking system, transferring data to listening computer(s). When two rendering slaves are used, an other computer is added and acts as the master for the slaves. The job of the master computer is to distribute the scenegraph, which is a list of 3D objects in a graph, to the rendering slaves. The HMD has two cameras attached in front of the displays. They gather images from the real world and deliver them to the rendering slave(s), where the images can be processed and renderings of the virtual world can be added appropriately. How these systems look schematically is seen in figure (4.5).

4.2 Software

4.2.1 OpenGL

OpenGL is a platform independent open standard for high-performance hardware-accelerated 2D and 3D graphics. The OpenGL interface consists of several hundred functions which enable the visualization of complex scenes using the graphics card processor. The specification is a leading standard, competing with Direct3D from Microsoft, which is not a platform independent library. The OpenGL library only knows about rendering functionality, and does not consern itself with windowing system or input/output devices. Higher-level functionality has to be handled by the application programmer, or other libraries which build upon OpenGL. One such abstracting library is Coin3D, an OpenGL derived library following the Open Inventor standard, which hides many of the difficulties of OpenGL from the application programmer, allowing her/him to consentrate on the functionality of his/her software.[34]

4.2.2 Coin3D

Coin3D, based on the Open Inventor standard, is an object-oriented software library of classes and functions which, among other things, constitute nodes in a scenegraph. The library is a set of tools and buildingblocks, which makes the programmer able to create 3D-graphics applications with minimum programming effort. The nodes in the library are easily extended and can be modified to suit application needs. The Inventor file format, which is extensible through adding new nodes, is a portable way of writing Inventor files with programmable nodes[39]. The Coin core system is window-system independent. The binding to the graphical user interface is done by GUI-libraries. Some of these bindings include Qt, Gtk, Xt, and more. Coin3D also handles other issues than rendering, e.g. picking, searching and querying the database and calculating the bounding box.

Coin3D is fully compatible with Open Inventor 2.1, and also adds support for more nodes and more advanced features, including VRML97 support and multiprocessor rendering.[35]

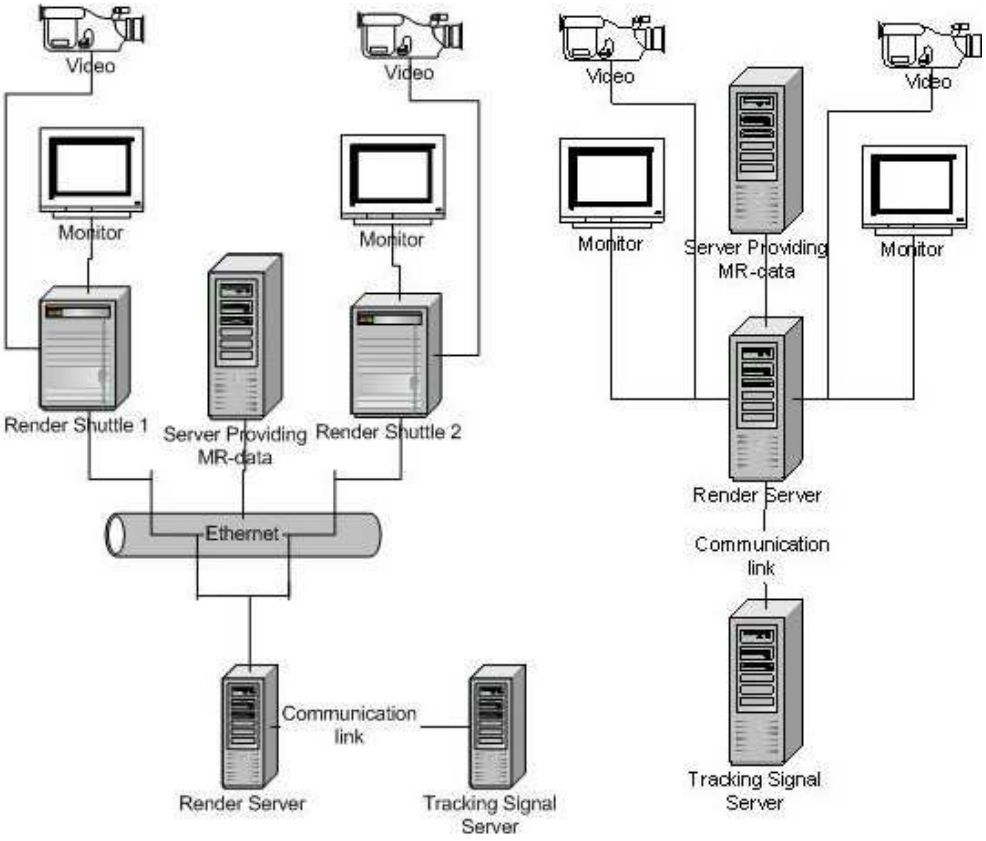

Figure 4.5: *System overview. Illustrates the two configurations typically adopted when working in Augmented Reality. One or two computers are used as rendering slaves, and in the case of two computers, an other computer acts as a master, controlling the two slaves. When just using one machine for rendering, the same computer has to be both master and slave, thus it will receive tracking signals, camera signals and perform the rendering by it self. MR-data is gathered from an other computer over the network.*

4.2.3 Studierstube

Studierstube is a GPL licensed development environment for creating Augmented Reality applications. The name is german, translates to "study room", and is meant to indicate a place where special computer equipment is available to create insight. Studierstube has support for specific hardware, and also provides an application programming interface called StbAPI. The programming is done using C++, and follows concepts found in the Open Inventor graphics toolkit. Actually, most classes found in Studierstube are made as Open Inventor nodes - making the design very uniform. Applications are a set of nodes in a scenegraph, and the scenegraph is read from Open Inventor files. To add further capabilities to Studierstube, addon nodes can be created and dynamically loaded.[36]

Studierstube was made to support most any type of hardware setup by making it a straight forward process to add support for new hardware devices. For tracking support, OpenTracker is used. OpenTracker is a C++ library which provides an abstraction between hardware and tracking coordinates. The location of tracked devices is sent into Studierstube as 3D events in the scenegraph.[36]

Working with Studierstube is done by executing an application loader program, thereby displaying a 3D graphical user interface (GUI). Applications are loaded by clicking with a pen on menu items in the 3D GUI. Even multitasking is possible with the system. Controlling application execution is done using a Personal Interaction Panel (PIP) which is a GUI component making it possible to start, stop and manage application programs.[36]

A special feature of Studierstube, is the ability for multiple users to interact together in the same augmented environment where each user get their own pen and PIP.[36]

4.2.4 OpenTracker

OpenTracker is a tracking framwork which acts as middleware for Studierstube and other applications in the need for hardware abstraction of tracking devices. Tracking signals provided by OpenTracker may come from a multitude of different hardware configurations, but this is invisible to the application programmer. Tracking signals might even come from a separate source which broadcasts signals over the network. OpenTracker reads this input and routs the data to listening applications.[40] OpenTracker is based on an object oriented design and XML, and attempts to achieve nothing less than **write once, input anywhere**.[40][41]

4.2.5 OpenCV

OpenCV is an open source computer vision library released by Intel, and provides C-functions and C++ classes which interface with powerful image processing algorithms. This enables the programmer to easily perform advanced operations on images. One

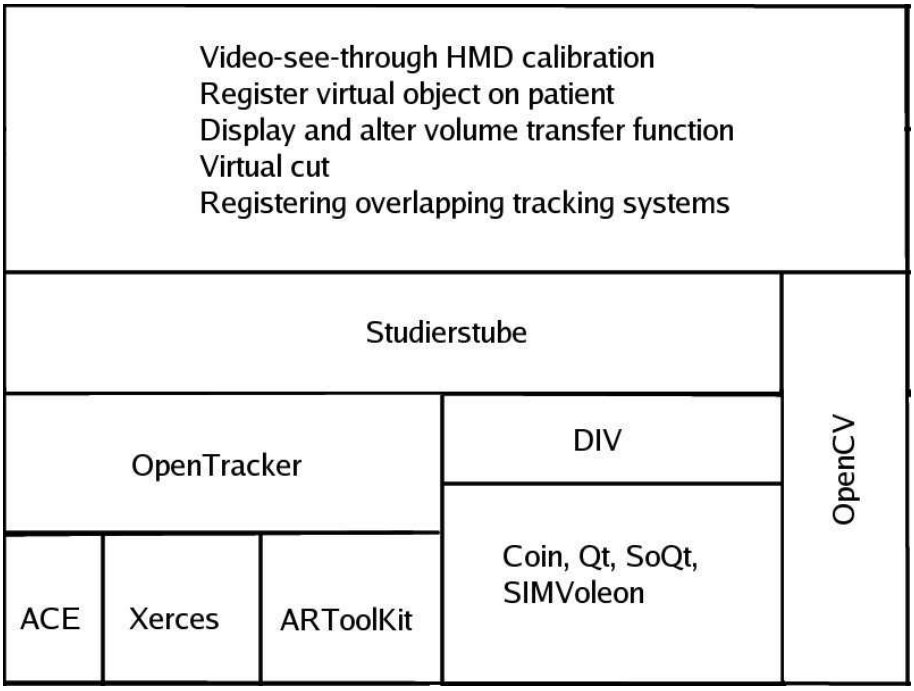

Figure 4.6: *The software hierarchy for which most software developed through the thesis is based on.*

of the important parts built into OpenCV, is the calibration algorithm developed by Zhang.[37][38]

The library is mainly intended for usage in real-time computer vision applications, where interesting areas span from object identification, face and gesture recognition, motion tracking to mobile robotics. The functions in OpenCV are high-performance, low-overhead operations performed mainly on images. Though having alot of low-level functionality, OpenCV is meant to be a high-level abstraction of the underlying algorithms. Thus making it possible to easily perform operations like: feature detection, tracking, 3D reconstruction, and camera calibration, among others. The software has specifically been optimized for Intel processors, performing best if this is the case. OpenCV is designed to function together with the Intel Image Processing Library (IPL), and uses the Intel Performance Primitives (IPP) to optimize performance if the binary is found during startup.[38]

4.2.6 System overview

The software development environment can be illustrated as a set of abstraction layers as illustrated in figure (4.6). The main software components of interest for this project are: Studierstube, OpenTracker, Coin3D and OpenCV. They are the foundation for all work conducted during the thesis.

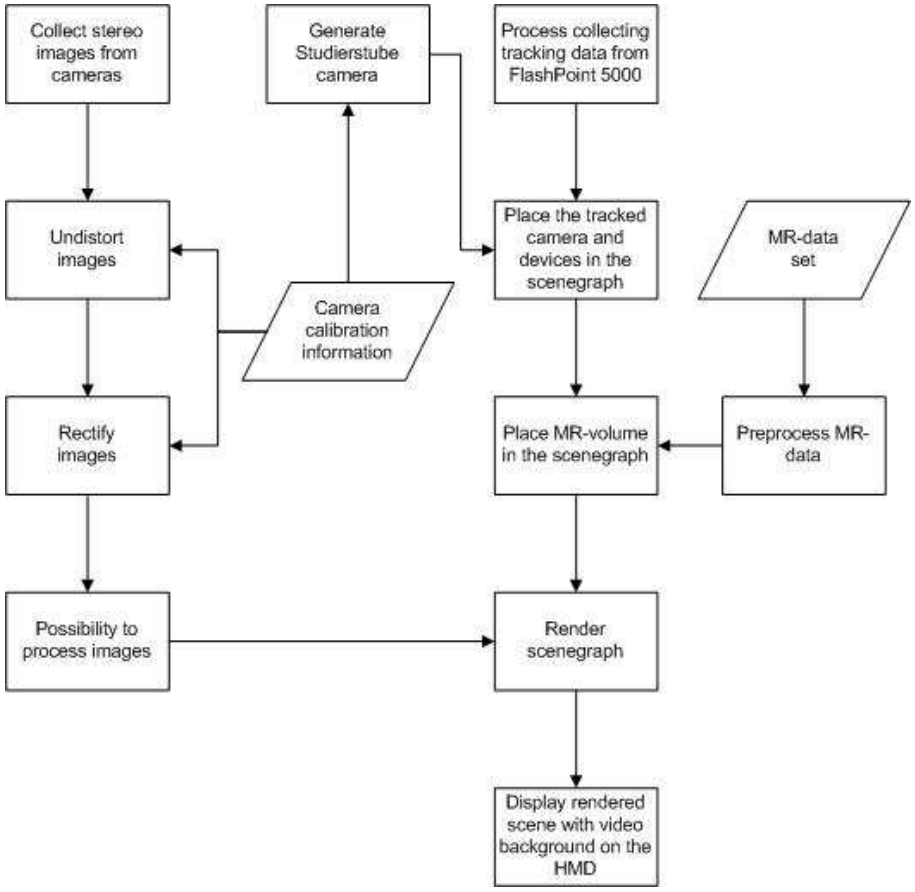

Figure 4.7: *An overview of an Augmented Reality system meant for clinical applications. It is important that the virtual camera(s) to be correctly calibrated to match the real camera(s). A volume is rendered based on the known location of the camera(s)*

Without further introduction, the Augmented Reality system developed for clinical use during surgery is illustrated in figure (4.7). Several processes are performed in order to accurately register a virtual object in the real world as seen through video cameras.

Chapter 5

Calibration and Registration

5.1 Motivation

Augmented Reality is essentially a mix between two worlds - the virtual world and the real world. In order to convincingly mix these two realities, the mapping from one world to the other is essential. We're more or less stuck with the real world, so what we can change is the virtual. It has to be scaled, translated and rotated to fit on top of the images from the real world. This mapping is mostly done by the tracking system, but there is another mapping problem that has to be addressed. The virtual and real cameras have to match with regards to position, perspective, etc. These problems are not present in immersive virtual environments where the only world visualized, is the virtual one. The problem of solving these mapping problems is called calibration, and is *the* essential problem in Augmented Reality. It has to be done at least once for every hardware setup.[10].

Especially medical applications have very high requirements when it comes to registration. A surgeon wearing a see-through HMD displaying virtual objects on top of a patient is not likely to trust in that system unless the registration errors are kept extremely small - below millimeters. Without good registration, systems using Augmented Reality may never be accepted in serious applications.[9] For more information on the registration problem, the reader is referred to section (2.1.3).

5.2 Static calibration and registration

5.2.1 Overview

[1] Static registration errors are those misalignment errors the user encounters when standing completely still while watching an augmented scene. There are four main sources of static registration errors[13]:

[1] Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

1. Optical distortion
2. Errors in the tracking system
3. Mechanical misalignments
4. Incorrect viewing parameters

Optical distortion

The video cameras used for capturing images almost always have some kind of optical distortion that has to be compensated for, in addition to the errors caused from optics in front of the displays of the HMD. The distortion is a function of the radial distance away from the optical axis, and the farther out from the center, the errors increase, which is why optical distortion is a bigger issue with wide field-of-view displays. This kind of distortion can, in most cases, be rectified by simulating the noise and applying an inverse filter, or by adding additional optics to the camera and display systems.[13]

Errors in the tracking system

Augmented Reality has an extremely high sensitivity to tracking errors compared to immersive Virtual Reality. The slightest error can cause a perceptual mismatch, and the illusion that the virtual and real world coexisting will be compromised. Hence, this is often the most serious type of static registration errors. They are unfortunately not easy to measure or eliminate, and can be non-systematic and difficult to fully characterize. The problem is not made any easier from the fact that almost all commercially available tracking systems are not accurate enough for serious AR applications[13].

Mechanical misalignments

Mechanical misalignments occur when the specifications given by the manufacturer do not match that of the actual product. Optics, displays or cameras may not be located exactly where they are meant to be, creating a registration problem within AR applications. Some mechanical misalignments might even happen during use, e.g. when a user rapidly rotates his head, minor movement of parts could occur, causing registration errors. A subset of these errors could be compensated for, but the best thing is to get it right initially, removing the need for extra calibration and compensation.[13]

Incorrect viewing parameters

Incorrect viewing parameters is a special case of alignment errors in which calibration procedures can be used to correct the view. For an HMD based system, the viewing parameters of interest are[13]:

- Center of projection and viewport dimensions

- Offset in both translation and orientation between the location of the head tracker and the user's eyes/cameras.

- Field of view

To correct these parameters, one could manually try to adjust them while looking at a registration error from a particular viewpoint, but this will generally not work for all viewpoints. Another approach would be to use tools like rulers for measuring distance between the eyes, and distance between the eyes and the tracker. This direct measurement approach for finding viewing parameters has not had much success, and other approaches are generally used. Making a user perform certain operations while wearing the HMD is another approach to calibration. The user gives input which is used to calculate the viewing parameters. These view based tasks assume the user gives correct information and that the tracker is accurate. Video based approaches have also adopted a lot of previous work done in robotics and photogrammetry on how to calibrate different viewing parameters of cameras.[13]

Other problems worth mentioning

Most all HMDs have **fixed eye accomodation**, meaning they have a predefined focus distance which cannot be altered. Some prototype HMDs are able to change the focal distance, but these are still very experimental[15]. There is also a problem with virtual and real images being in and out of focus. Virtual images are modeled by a pinhole camera which has infinite depth perception. Video cameras, on the other side, are only able maintain focus within a certain distance from the camera. This problem could be resolved by performing additional computations to simulate limited depth of field for the virtual images. In addition, the video camera could have autofocus.[13]

Most video-see-through HMDs have **parallax error** because the cameras are mounted away from the true eye location. If the cameras are mounted on top of the HMD, the view will be significantly different than just seeing with the eyes, and the person wearing the HMD has to get used to the parallax error[15]. Also, having the cameras mounted with different interpupilary distance than the user's eyes is a source for parallax error. It is actually possible to reduce this error by either mounting the cameras very close to the true eye locations, or by using mirrors to get an optical path more equal to that of the eyes[13]. Both approaches are illustrated in figures (2.2 and 2.3).

The captured video signals rendered to the HMD have far **less resolution than the resolving power of the fovea**[13], giving the wearer a less accurate impression of the real world than would be given by using optical-see-through systems.

5.2.2 Previous work

Studierstube

At Vienna University of Technology, several projects on registration and calibration have been conducted. The goal of some of these, is to automate the calibration process as much as possible. The calibration processes developed focus mainly on optical-see-through HMDs. A working framework for calibrating these HMDs has been built into Studierstube[10].

In [21], a simple and fast calibration scheme not requiring additional instrumentation or complicated procedures is presented. The process of calibration is done, even by inexperienced users, by following an interactive guide/wizard that asks the user for certain input. The method has shown to produce stable results and is applicable for both optical-see-through and video-see-through HMDs, but is not meant to be the optimal way of calibrating video-see-through HMDs.

Video-based HMDs are essentially immersive HMDs with attached cameras[21]. The cameras provide the video that is rendered as the background of the Virtual Environment. The calibration procedure for video-see-through HMDs is a more general procedure than the calibration of optical-see-through HMDs. This is because the calibration does not have to be done for each user of the system. The reason being the calibration is done for the video-cameras, and not our eyes, as it would be in the optical approach.

5.2.3 Problem statement

Several approaches have been made in previous reports that deal with the static calibration problem of Augmented Reality, but not so many deal with the calibration of systems by untrained users[21]. If doctors, and other personnel are to use the AR-environment, they have to be able to easily perform such calibration without much or any a priori knowledge of the intricacies behind the procedure. Also, many calibration procedures have been presented in photogrammetry, robotics, Augmented Reality, etc. Some of the most interesting being almost automatic. The author does not believe he could do much better than what already has been done on this field. However, when it comes to the process of automating the calibration process within Augmented Reality environments, there is still much work to be done.

In the previous work section, it was mentioned that Studierstube has a calibration procedure made for optical-see-through HMDs (that also works for video-see-through HMDs), but the properties of optical-see-through HMDs make their calibration significantly different from the calibration of video-based HMDs[21]. This opts for the development of a calibration framework (for Studierstube) that is easy to use, but especially meant for calibrating video-see-through HMDs.

The problem at hand in this part of the thesis is the calibration of general video-see-through head mounted displays in the Studierstube environment.

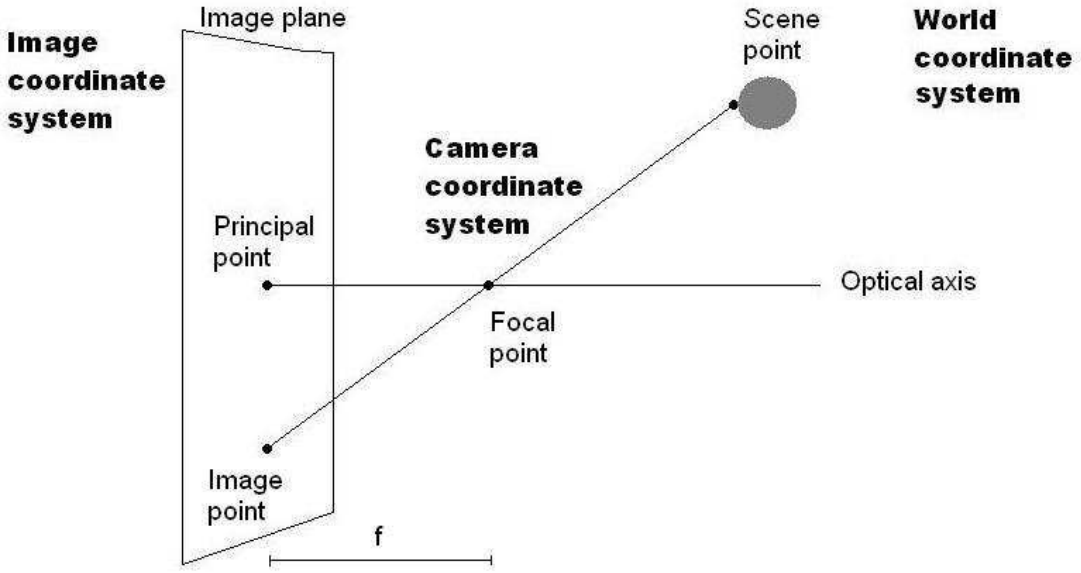

Figure 5.1: *Normal pinhole camera used in computer graphics*

5.2.4 Procedure

Camera model

A camera basically only performs a projection from 3D coordinates into 2D coordinates. The camera model of choice for most graphics applications, is the pinhole camera model. It is defined as seen in figure (5.1). Another popular representation of the pinhole camera is to have the image plane in front of the focal point, thus not mirroring the scene as done with the other representation.

Camera distortions

Cameras use lenses to produce images on an image plane. One can, however, not use a pure perspective transformation to represent this mapping. The reason being the distortions produced by the camera, e.g. from its optical parts[42]:

- **Radial distortions** is the difference between a captured image and an image taken with a perfect perspective pinhole camera. It happens when light rays are bent differently than in the ideal case[43]. The radial distortion curve is a function of the focal distance of the camera[42].

- **Decentering distortions** is when the principal point on the projection plane does not lie on the optical axis[43].

- **Tangential effect** is caused by misalignments of lens components relative to the optical axis, and is equivalent to the prism effect. The tangential effect causes errors in angles based on the principal point, and is caused by imperfect glass or imperfect centering of lens components. The **prism effect** is the effect of a thin prism being placed in front of a lens[42]. The tangential effect is often only used in more comprehensive lens models.

These parameters have to be found through calibration procedures, and then used to undistort the images captured from the camera, thus making them closer to what would have been captured by an ideal camera. Radial distortion and decentering distortion can in most cases be assumed to be rotationally symmetric, and are often approximated using polynomials [44][43].

$$x = x_0 + (x_0 - c_x)(K_1 r^2 + K_2 r^4 + \ldots) \tag{5.1}$$

$$y = y_0 + (y_0 - c_y)(K_1 r^2 + K_2 r^4 + \ldots) \tag{5.2}$$

$$r = (x_0 - c_x)^2 + (y_0 - c_y)^2 \tag{5.3}$$

Equations (5.1, 5.2, 5.3) model the radial distortion of a typical lens[45]. c_x and c_y represent the image center, and x_0 and y_0 the current image location. K_i are the radial distortion parameters for the lens. Lenz[46] notes that accurate modeling of the lens distortion is important for accurate 3D measurements, and that a 2nd order polynomial is enough to model the distortion accurately enough. Adding more in terms of order does not give noticeably better results. Since the radial distortion is a function of the focal distance of the camera, the calibration has to be done for each change of focus. The easiest way to avoid this is to use fixed optics, but that also has apparent drawbacks[45].

For 3D-vision the proper choice of image center can be critical, and is not necessarily in the center of the image plane. If only coarse 3D measurements will be performed, the center is not so important, but if more accurate measurements are needed, the center should be calibrated.[46]

Calibration procedures

The goal of camera calibration is to find the relationship between 2D points captured by a camera, and their 3D counterparts[46]. There exists several different calibration procedures. These range from needing predefined objects, predefined control points or predefined viewpoints to calibration techniques that require little or no interaction from the users. Two of the most interesting techniques reqiring little interaction are self-calibration and on-the-job calibration, which have inherited much from the field of photogrammetric camera calibration.

Self-calibration - By using methods like non-linear least squares (also known as bundle adjustment[47]), it is theoretically and practically possible to simultaneously find the lens parameters and the coordinates of points in three dimensions. This is done with no explicit intervention by people except the movement of the camera in a scene. Several different viewpoints are generally needed for the calculations. The criterias needed for successful self-calibration are[42]:

1. A camera is used to take at least three images from different viewpoint of an object.
2. The camera mechanics and the object must remain static during the procedure.
3. The photogrammetric network must be strong and have a high degree of convergence.
4. One or more images must have a roll angle that is significantly different than the other.
5. The scene should contain a significant amount of highly distributed points.

If these criterias are met, a good calibration of the camera should be possible without the need for control or intervention of any kind. A problem in aerial applications is the difficulty of obtaining images with sufficiently different camera angles[42]. This does, however, not seem to be a problem so much for Augmented Reality since the camera can be placed however needed. [48] notes that self-calibration is very flexible, but not yet mature.

On-the-job calibration - This is a similar technique to self- calibration, but requires a stricter setup. A predefined pattern of control points is used as input and the camera is moved around the pattern. For each new view, the user tells the computer to save the current image and use it later for calculating the camera parameters. Bundle adjustment is also used with this configuration to calculate the parameters. Some of these being: lens distortion, focal length, and offset of the principal point. On-the-job calibration is the most common type of calibration procedure currently being used. "On-the-job" calibration is often confused with "self-calibration", where there is actually no need for control-points at all.[42]

Photogrammetric camera calibration is the classic calibration method. Zhang[37] defines this calibration procedure as being one which uses predefined calibration objects whose geometry is known with high precision. The objects used for calibration are often planar and placed at angles with each other. As can be gathered, these setups require well-defined and often expensive calibration equipment.

Basics of camera calibration

The simplest case mathematically, is the calibration of a single camera in a known scene. In this case, the world coordinates and the image coordinates for the corresponding points are known and used for calculating the camera calibration matrix.

$$\tilde{u} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} -fa & -fb & -u_0 \\ 0 & -fc & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{bmatrix} \tag{5.4}$$

The matrix part of equation (5.4) is named K, and is the camera calibration matrix[44].

$$K = \begin{bmatrix} -fa & -fb & -u_0 \\ 0 & -fc & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

Finding the constants in this matrix is what is done when performing camera calibration. Because we are working with homogeneous coordinates, it is possible to write equation (5.4) as[44]:

$$z_c\tilde{u} = z_cK \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{bmatrix} = K \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = KR(X_w - t) \tag{5.6}$$

Where R and t represent the transformation (rotation and translation) of the camera with respect to the world coordinate system, and are called **extrinsic** parameters because they are dependent on how the camera is located in the world. There are six such parameters; one parameter for each axis of rotation and translation. The matrix K from equation (5.5) consists of coefficients that can be grouped into five constants representing the **intrinsic** parameters of the camera. The intrinsic parameters are those not dependent on the relative position of the camera in the world coordinate system. These five parameters together model a scale, shear and translation. From equation (5.5), we get $\alpha_u = -fa$, $\alpha_v = -fc$ and $\alpha_{shear} = -fb$. u_0 and v_0 are kept as constants as well. Both the extrinsic and intrinsic parameters can be expressed in a single 3×4 projective matrix, M, as seen in equation (5.7)[44].

$$\tilde{u} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} = [KR| - KRt] \begin{bmatrix} X_w \\ 1 \end{bmatrix} = M \begin{bmatrix} X_w \\ 1 \end{bmatrix} = M\tilde{X}_w \tag{5.7}$$

$[U, V, W]^T$ is the homogeneous image coordinates of the world point X_w.

Single camera calibration is the process of determining the coefficients in the camera calibration matrix K (equation 5.5), or the projective matrix, M (equation 5.6). Finding K will only give the intrinsic camera parameters, while finding M will give both the intrinsic and extrinsic camera parameters. There are two basic cases of camera calibration[44]:

1. Calibration with known scene

2. Calibration with unknown scene

When the scene is known, a set of non-coplanar points is needed, and corresponding pairs of image and world coordinates are used for mapping their relationships. Each of these equations can be written as equation (5.8)[44]. One of the problems with this approach is the need for special calibration objects, and is thus the beginning of a time consuming calibration procedure.

$$\alpha_j \tilde{u}_j = M \begin{bmatrix} X_j \\ 1 \end{bmatrix} \tag{5.8}$$

The reason α is placed here, is because we are using homogeneous coordinates, and any non-zero constant can be multiplied with such a vector without altering its value when projecting it to a two-dimensional vector.

Calibrating cameras can also be done in unknown scenes, but more images from different viewpoints are needed[44]. This works because the intrinsic camera parameters will not change from view to view. Within the calibration in unknown scenes, there are two main cases:

1. **Known camera motion** - different methods exist for pure rotation, pure translation, and a mixture of the two.

2. **Unknown camera motion** - self-calibration techniques are used in order to find the calibration parameters.

When using self-calibration, at least three different images from different views have to be the basis for calculating the camera parameters.

Calibration of one camera from a known scene

Calibrating a camera from a known scene is typically done by calculating the projective matrix, M, then use this matrix to calculate the intrinsic and extrinsic camera parameters. As previously mentioned, each corresponding pair of world and camera coordinates can be written as in equation (5.8), and by expanding this equation and processing it slightly, we get equation (5.9)[44].

$$\begin{aligned} u(m_{31}x + m_{32}y + m_{33}z + m_{34}) &= m_{11}x + m_{12}y + m_{13}z + m_{14} \\ v(m_{31}x + m_{32}y + m_{33}z + m_{34}) &= m_{21}x + m_{22}y + m_{23}z + m_{24} \end{aligned} \tag{5.9}$$

This equation, with twelve unknowns (in reality elleven unknowns because of the unknown scale factor when using homogeneous coordinates), can be used to find the projective matrix, M, by rewriting it as done in equation (5.10)[44].

$$\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz & -u \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz & -v \\ \vdots & & & & & & & & & & & \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{bmatrix} = 0 \tag{5.10}$$

This matrix can and should contain several corresponding pairs of world and image coordinates. At least six such corresponding points are needed in order to find a result, but often the system is overdetermined by adding additional points. The exact solution of the equation is most probably not feasible due to noisy measurements, so an optimization algorithm like the least-square method is often used to calculate the resulting projective matrix, M. The extrinsic and intrinsic parameters are then found by using the relation of equation (5.7), rewritten as[44]:

$$M = [KR| - KRt] = [A|b] \tag{5.11}$$

Finding the translation, t from this is trivial, but calculating the K and R matrices requires a little more work. K and R are found by using the knowledge that the calibration matrix, K, is upper triangular, and the rotation matrix, R, is orthogonal. A factorization can then be performed to find the matrices. Singular value decomposition (SVD) or QR decomposition can be used for the calculation.

The calibration method described previously is often referred to as the **Direct Linear Transform** (DLT) in litterature.

Stereo camera calibration

Stereo camera calibration is based on that of single camera calibration. The main differences being the additional problems and possibilities present when adding an other camera. Stereo camera calibration is of high relevance to Augmented Reality due to the natural stereo vision of human beings.[44]

If two cameras are calibrated and are watching the same point in space, the 3D coordinates of the point can uniquely be determined by the intersection of rays from the two cameras - this is the basic principle of stereo vision.[44]

Stereo vision gives rise to epipolar geometry, as illustrated in figure (5.2). When the imaging planes are parallel, the epipolar lines also become parallel in the imaging planes. This is a special case which has been named the **canonical configuration**, and is seen in figure (5.3). It is easier for a human operator, and also a computer to search for equivalent points along a scanline instead of arbitrary lines. Having two cameras which do not have parallel epipolar lines, the images can be transformed to the canonical configuration by a process known as **image rectification**. It is, however, noted that rectification causes resampling which causes loss of resolution.[44]

Having two image planes in a canonical configuration, as illustrated in figure (5.4), it is possible to find the 3D location of a point found in images from both cameras using elementary geometry. The depth value can easily be calculated from the equation (5.12) given the distance between the optical axes of the cameras, $2h$, the focal length, f, the distance between the left optical axis and the intersection of a ray through the world point, P_l, and a corresponding P_r for the right camera[44]:

$$z = \frac{2hf}{P_r - P_l} \tag{5.12}$$

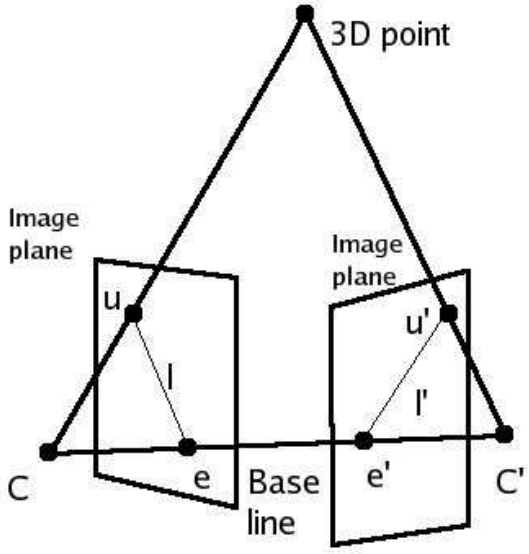

Figure 5.2: *The line connecting the two centers of the cameras is called the baseline. Any 3D point observable from the stereo cameras define an epipolar plane. The intersection of the epipolar plane and the image planes are called epipolar lines,* l *and* l'*. All such lines pass through the epipoles* e *and* e' *- the intersections of the baseline with the image planes. Any point visible to both cameras has to have projected image points lying on the epipolar lines. This is a constraint which essentially reduces the search for a corresponding points from 2D to 1D.[44]*

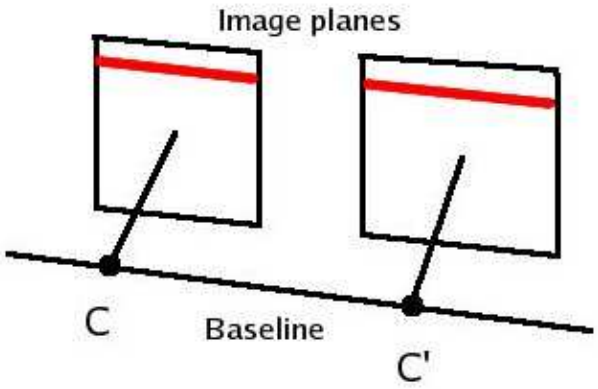

Figure 5.3: *When the imaging planes are parallel, the epipolar lines also become parallel in the imaging planes. This is a special case which has been named the* **canonical configuration***, and makes searching for equivalent points in images captured by the cameras just a search of two corresponding scanlines.[44]*

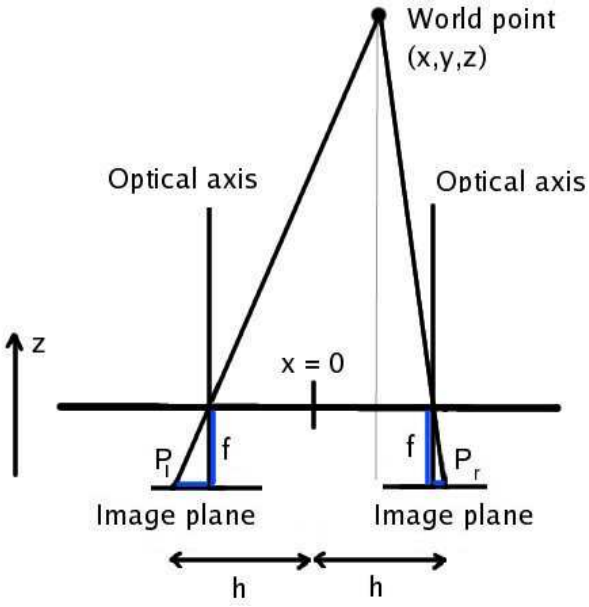

Figure 5.4: *A stereo camera setup in a canonical configuration. This figure illustrates the use of disparity to calculate the distance from the cameras to a point,* (x, y, z)*, in space visible from both cameras.* f *is the focal distance of the cameras,* P_l *is the distance between the left optical axis and the intersection of a ray through the world point and the image plane. A similar definition goes for* P_r*.* $2h$ *is the distance between the two optical axes.[44]*

$P_r - P_l$ is the effective **disparity** while observing the world point, P, and if the disparity is zero, z will be at an infinite distance from the cameras.[44]

The **fundamental matrix** is an important theoretical foundation for stereo vision. Having two cameras whose optical axes are not parallel, the coordinate system for the left view can be transformed into the coordinate system of the right view by a translation and rotation. The translation moves the left camera center, C to the right camera center, C'. This transformation is captured in the fundamental matrix, F, and the relation between two image points, u_i and u'_i, in different image planes is written as equation (5.13).[44]

$$u_i^T F u'_i = 0 \tag{5.13}$$

Having at least seven corresponding points in the left and right images enables the calculation of the fundamental matrix.[44]

Calibration using more cameras than two will not be covered in this thesis, however, fully automated methods have been developed, and one such utilizing a laser pointer is described by Svoboda in [49], and more information can be found in [44].

Different calibration groups

Calibration methods can broadly be divided into two major groups: **linear methods** and **non-linear methods**. Linear methods are efficient, but lack accuracy and robustness. Non-linear methods are accurate, but require an initial guess of parameters and are more computationally demanding. The initial guess is of high importance for the non-linear methods to converge towards an accurate solution. In order to work around this problem, a linear method can first be applied in order to find good guesses for the camera parameters. Then, the extrinsic camera parameters are calculated by the non-linear method using the calculated guesses. Linear methods, on the other hand, are known to respond badly to noise. It has been shown that if the noise rises above a certain level, and if the number of points is low, errors go through the roof. Adding additional control points can help reduce this error, but in certain applications, adding more points is not desireable. When the initial guesses are not good enough, the non-linear optimization algorithm has a problem of getting stuck in a local minimum. For this reason, genetic algorithms have been used in order to solve the camera calibration, a topic touched upon later in this thesis.[50]

Direct Linear Transform

Direct Linear Transform, or DLT for short, is a common calibration method[51] developed by Abdel-Aziz and Karara. It has since been revised in later publications. The method is based on the pinhole camera model, ignoring all distortions. The calibration method is similar to the one introduced in section (5.2.4) when describing basic single camera calibration. A calibration program using this algorithm was attempted implemented for this Master's thesis, but was not able to find a non-trivial solution to equation (5.10) without modification. This is a known problem when trying to solve the linear calibration system. In order to avoid the trival solution, a proper normalization has to be applied. The original authors set $m_{34} = 1$ and solved the equation with a pseudo-inverse technique. However, using this method, a singularity is introduced when the correct value of m_{34} is very close to zero. An other, singularity free, approach was suggested by Faugeras and Toscani, namely setting $m_{31} + m_{32} + m_{33} = 1$.[52]

Camera calibration with genetic algorithms

A non-conventional way of performing camera calibration is introduced by Li et al. in [50]. The method uses **Genetic Algorithms** (GAs) in order to accurately calibrate a camera, and shows good results compared to Tsai's procedure described in the next section. The reason such a method can be applied to camera calibration, is the way it maps to solving non-linear problems when little is known about the solution. Calibrating with genetic algorithms does not call for an initial guess of the solution, something non-linear methods do. In addition, GAs are robust and attempt to climb out of local minima. Finally, they also perform good when noise is present. Results from experiments suggest that accurate calibration can be achieved with a minimum number of

points (about seven), and that adding more points does not necessarily increase the accuracy of the calibration. An interesting and important advantage of this type of calibration is the way error in calibration is a function of noise. The GA method was said to keep the error level almost constant for certain noise levels, with errors increasing linearly with increased noise.[50]

A related calibration method using neural networks is presented by Memon et al. in [53].

Roger Tsai camera calibration

In 1987, Roger Tsai published a camera calibration procedure applicable to almost all cameras. It is a classic calibration method which is cited by most litterature on the subject. Tsai's calibration procedure is based on the pinhole camera, and models first order radial lens distortion. It works by calculating the mapping between known 3D locations and their image plane projections. The technique requires at least five or seven accurately detected points in a coplanar or non-coplanar configuration respectively. The points can be arbitrarily placed, but located at known geometric positions. The camera model has five intrinsic and six extrinsic parameters which have to be calculated. The intrinsic parameters are[54]:

- The focal length, f
- First order radial lens distortion component, K_1
- The center of the radial distortion, and the point where the z-axis intersects with the image plane, (C_x, C_y)
- A scale factor, s_x

The six extrinsic parameters are represented by the rotation and translation, each having three parameters. Additionally, there are six fixed camera constants which need to be set before starting the calibration process[54]:

- N_{cx}, the number of sensor elements in the x-direction of the camera
- N_{fx}, the width in pixels of the image captured by the camera
- d_x, the dimension along the x-axis of the camera sensor element
- d_y, the dimension along the y-axis of the camera sensor element
- d_{px}, the effective dimension along the x-axis of a pixel in the frame grabber
- d_{py}, the effective dimension along the y-axis of a pixel in the frame grabber

These constants are not the same for all cameras, thus they have to be found from the camera specifications. Consumer cameras, as the ones used for the thesis, do not normally specify these values. To get them, one usually has to contact the manufacturer. However, good calibration could be achieved by just finding the correct ratio between d_{px} and d_{py}. This value can be approximated by finding the ratio of a rectangle, $width/height$, visible by the camera. Having this ratio correct, the calibration should be able to converge.

To accurately estimate image center and radial lens distortion, world coordinates used for calibration should be sampled broadly accross the field of view, and span the depths the model is intended to work for. Finding the image plane coordinates of the world points needs to be done with subpixel accuracy in order to achive a good calibration results. Still, there are more issues which could cause erroneous results of the calibration process. The **worst-case scenario** in coplanar calibration is if all samples are placed on a plane which is parallel to the image plane. Having an angle of 30 degrees or more is recommended in order to spread the samples over a broader depth range. Non-coplanar calibration will suffer if the 3D world coordinates lie in a volume which is small relative to the distance from the camera.[54]

An interesting aspect of Tsai's calibration method is the fact that **pose**, i.e. the extrinsic camera parameters, can be calibrated in a separate step if the intrinsic camera parameters already are known. This has the potential of speeding up the calibration, and enables the easy calculation of the extrinsic camera parameters if the camera has been moved relative to the world coordinate system, while the intrinsic camera parameters remain constant.[54] However, as will be discussed in section (5.2.4), calibrating intrinsic and extrinsic parameters independent of eachother could cause inaccuracies in the calibration and should be avoided if possible.

Li et al. shows in [50] that Tsai's method is extremely accurate when the image data has little or no perturbation. However, when noise rises above a certain level, so does the error in Tsai's procedure; pixel errors were said to almost exponentially increase! The genetic algorithms method described by Li et al. from section (5.2.4), was said to keep the error level almost constant for certain noise levels, and increased linearly with increased noise. This means that when using Tsai's method, the noise has to be controlled and kept low to achive accurate calibration results. If noise levels cannot be controlled, other calibration methods could be implemented, e.g. using genetic algorithms.[50]

A variant of Tsai's calibration method, implemented by Reg Willson[54], was utilized for calibration purposes for this thesis. Software routines are provided for two levels of calibration. The first level is a direct implementation of Tsai's algorithm, where only three of the 11 camera intrinsic and extrinsic parameters are numerically optimized. The second level is an extension of Tsai's algorithm, where all 11 parameters are numerically optimized. It is noted by the author that full optimization is slower, but provides more accurate results.[54]

Zhang camera calibration

The calibration procedure presented by Zhang et al. in [37] is the calibration method implemented in OpenCV and has been used during this thesis. The method can be used even without prior knowledge of computer vision or the problems involved with calibrating cameras, and needs two or more images of a planar calibration pattern from different angles in order to find the camera parameters, including radial distortion. The calibration pattern or the camera can be moved freely without affecting the result. The calibration procedure consists of a closed-form solution which is followed by a non-linear refinement based on the maximum likelihood criterion.[37]

Zhang's method is a fairly new addition to the computer vision toolbox. When it surfaced in 1998, it significantly added to the usagespace of camera calibration. It is no longer necessary to have significant prior knowledge of camera optics, sensors, etc, and still, very good results can be achieved. The technique is shown to be both easy to use and flexible, taking camera calibration from laboratories to real world use.[37]

There already exist several different camera calibration procedures, some of which are mentioned in section (5.2.4). These include *photogrammetric camera calibration*, *self-calibration* and *on-the-job calibration*. Zhang's method is of the type on-the-job calibration because it requires a predefined calibration pattern to be present, and is referred to as a calibration method for **Desktop Vision Systems** (DVS). A DVS is a minimalistic system consisting of e.g. webcamera(s) connected to a computer. Most people do not have expensive calibration equipment, and are not willing to buy such hardware. This makes Zhang's approach very interesting for computer vision systems for a wide variety of users.

The recommended procedure of calibrating a camera by Zhang's method is[37]:

- Print a pattern and make sure it is planar
- Take images of the calibration pattern at different angles. This can be done by either moving the camera or the calibration pattern.
- Find the gridpoints in the images
- Estimate the intrinsic and extrinsic parameters
- Estimate the radial distortion coefficients
- Refine all the parameters by minimization

Zhang models the two first radial distortion parameters, and notes that adding more parameters does not give much higher accuracy, but could actually cause numerical instability[37]. The intrinsic camera parameters calculated by Zhang's method are[38]:

- Focal length
- Center of image

- Pixel size
- Radial distortion

The extrinsic parameters include world rotation and translation, together specifying the transformation between the camera and world reference frames. The relationship between an image point and its corresponding 3D point is represented by the following equation[38]:

$$s\tilde{m} = A[R|t]\tilde{M} \tag{5.14}$$

Here, s is an arbitrary scale factor which can be used because $\tilde{m}$ is a homogeneous coordinate representing an image point. $\tilde{M}$ is also a homogeneous coordinate, but represents a 3D coordinate which projects to the point $\tilde{m}$ on the image plane. R and t are the extrinsic camera parameters (rotation and translation respectively), and transforms from the world coordinate system to the camera coordinate system. $[R|t]$ is a matrix composed of the 3×3 rotation matrix and the 3×1 translation vector, making the resulting matrix 3×4. A is a matrix of the intrinsic camera parameters and is defined as follows[38]:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

u_0 and v_0 represents the principal point, meaning the center-point of the image, and α and β are scale factors in the u and v axes of the image. γ is the skewness of the image axes.[38] Looking back at the introduction of camera calibration and equation (5.5), it is easy to see the similarities between the matrices A and K. This is no coincidence, A is actually the same matrix as K, rewritten using different constants. So, it comes as no surprise that equation (5.14) represents the same projection as found in equation (5.6). The two projections just use slightly different mathematical syntax.

The extrinsic parameters are represented by a rotation matrix, R, and a translation vector, t. They are stored in the following format[38]:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{5.16}$$

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{5.17}$$

An important observation about the method, is that just translating the calibration pattern, or just translating the camera does not give the additional constraints needed when trying to calibrate the camera. It is therefore very important to have rotational components different in the calibration images.[37]

Results from calibration attempts with real and constructed data show that calibration error is decreased when using many images. Especially, having three images

instead of two significantly reduced the error. Additionally, having significant angles between the model plane and the camera is important for good performance. An angle of 45 degrees was shown (by experiments) to give the best results. However, when the angle between the model plane and the camera increases, the area of the plane visible to the camera is reduced, thus making corner detection less accurate. Testing corner detection when rotating the pattern was not implemented in the experiement conducted by Zhang et al., and it may be possible that angles around 30 degrees could give better results in practical implementations.[37]

Having a model pattern which is not planar is also a problem for the technique. However, if the non-planarity of the pattern is less than 3%, the results are still good. One should in any case try to make a planar pattern, printed on a high-quality printer where aspect ratio and other dimensions are preserved.[37]

Accuracy of calibration

In [55], Gonzales et al. compare different camera calibration methods, including the ones developed by Zhang and Tsai. The accuracy of point reconstruction was measured. Additionally, the stability of the calibration was measured when the pattern was moved relative to the camera, and when the intrinsic camera parameters were altered. Two types of stabilities were defined: if only the extrinsic parameters were altered, then the intrinsic parameters should stay the same. Conversely, if only the intrinsic parameters were altered, then the extrinsic parameters should stay the same. The study was performed using both real and simulated data, with varying number of points and varying locations. Errors were measured in 2D and 3D. The errors were respectively measured as the mean distance between real and reconstructed image coordinates and mean distance between real and reconstructed point coordinates. Because the real parameters of a camera are unknown, it was not possible to know if the calibration using a particular method came close to the correct result. For this reason, data which simulates a known camera was generated and used for simulation. Noise was added to the simulated data in order to test the influence on the calibration methods.[55]

To be able to test the stability of the extrinsic parameters, the camera was zoomed, thus altering the focal length of the camera. The stability of the intrinsic parameters were tested using two different methods; displacing the calibration pattern parallel to the camera, and placing the pattern at different locations and orientations relative to the camera.[55]

The very important result from the experiments, is the fact that most of the calibration methods are unstable - that is, they do not result in the same intrinsic parameters when the calibration pattern was moved relative to the camera. A similar observation was made in the extrinsic case. When just the intrinsic camera parameters were varied, the parameters related to the altered parameters became affected by the change. An example is that change in focal distance caused the extrinsic parameters also to vary.

The influence of noise was measured on simulated data only. Most of the methods tested attempted to minimize the the image space error, and errors were compareable

with the methods. The relative equality of error measurements did, however, not result in the same camera parameters being calculated by the different methods.[55]

The conclusion of the experiments was that the values generated by the different calibration methods are not necessarily comparable or pointing towards the same, ground truth, solution. In addition to this, the experiments also showed a strong coupling between intrinsic and extrinsic parameters calculated by one specific calibration method, and it was noted that when moving the calibration pattern or moving the camera, even though the intrinsic parameters in theory should stay the same, this is not always the case. If an accurate calibration is wanted, both intrinsic and extrinsic parameters should be estimated at the same time.[55]

Kalman filter

Errors in the reported output from the tracking and sensing systems are often the most serious type of static registration errors[10]. These errors are difficult to characterize and handle, but trying to filter the signal to reduce noise is possible, and is exactly what the Kalman filter does. In his rather famous paper from 1960, R. E. Kalman described a recursive solution to the discrete-data linear filtering problem[60]. Welch and Bishop[63] describe the Kalman filter in the following way:

"The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimation of past, present and even future states, and it can do so even when the precise nature of the modeled system is unknown"

The process described by the Kalman filter addresses the general problem of estimating the state $x \epsilon \mathcal{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{5.18}$$

with a measurement $z \epsilon \mathcal{R}^m$ that is:

$$z_k = Hx_k + v_k \tag{5.19}$$

w_k and v_k are random variables with w_k representing the process noise and v_k representing the measurement noise. They are also assumed to be independent of each other, white, and with normal probability distributions.

What the Kalman filter does, is to estimate the state of a process at a particular instant in time. Feedback is given by newly obtained measurements (noisy), which are used to create an a posteriori estimate of a previously calculated a priori estimate. This corrected estimate is, for the next iteration, used to calculate the new a priori estimate, and so on. This recursive property of the Kalman filter is what makes it so appealing, because it makes practical implementations much more feasible than e.g. an implementation of the Wiener filter (which is constructed to work on all available data

directly for each estimate). The equations fall into two main categories: time update equations and measurement update equations. The time update equations predict the future, and measurement update equations updates the state on which the predicted estimates are based. Time update equations can be thought of as predictor equations, and measurement update equations can be thought of as corrector equations[63].

For a better introduction to the Kalman filter, the reader is referred to [63].

5.2.5 Results

Overview

Having covered a lot of topic in the procedure section, it is time to sum things up a bit. Not all parts mentioned have been implemented, and clarifying this issue seems appropriate. An overview of what has been implemented is seen in the following list:

- Calibration of intrinsic camera parameters using Zhang's method implemented using OpenCV.
- Calibration of the extrinsic camera parameters using Tsai's method implemented by Reg Willson.

Intrinsic camera calibration is conducted by taking a series of pictures of a chessboard calibration pattern from different angles. The corners of the squares on the pattern are found by algorithms within OpenCV at subpixel accuracy. *Extrinsic camera calibration* is performed by moving a tracked pen slowly in front of a camera attached to a tracked HMD. On the tip of the pen, a 2×2 chessboard pattern is attached to make it possible to automatically find the projected location of the pen on the image plane. When the pattern is recognized, both the location of the pen relative to the head tracker and the image plane projection are saved for use in Tsai's calibration algorithm. Both stereo and mono calibration is possible for the two calibration methods.

Calibration takes place in Augmented Reality, and is built on the Studierstube framework. Thus, calibration has to produce camera parameters valid for the **SoOffAxisCamera** model used by Studierstube. SoOffAxisCamera has several parameters representing the state of the camera[56]:

- **eyepointPosition** is the location of the eye
- **position** is the position of the center of the image plane
- **orientation** is the orientation of the image plane
- **size** is the size of the image plane

These parameters together model a camera illustrated by figure (5.5). The goal of the camera calibration is to find the intrinsic parameters in addition to the translation and rotation of the camera relative to a headtracker, illustrated in figure (5.6).

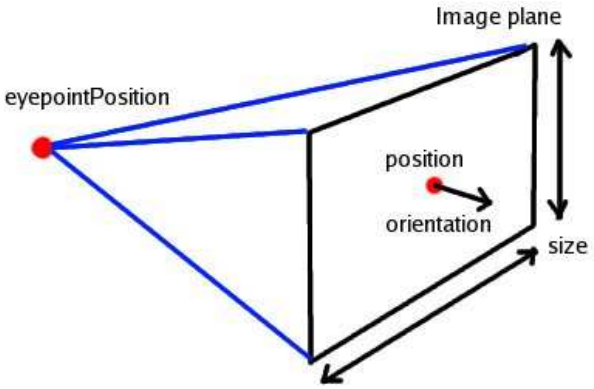

Figure 5.5: *Illustrates the SoOffAxisCamera model used in Studierstube. Eyepoint and image plane are two separate entities which can be moved independently. The orientation vector should actually point in the opposite direction, but was drawn this way for illustrational purposes[10]*

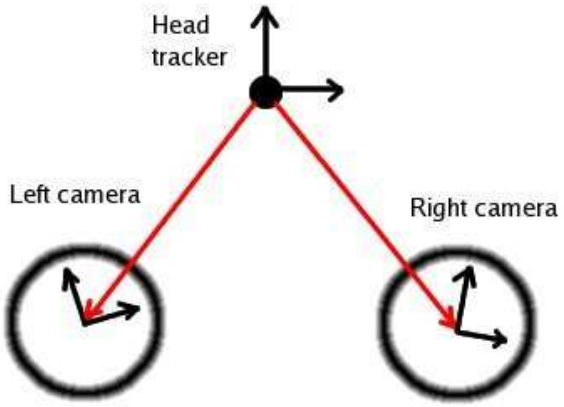

Figure 5.6: *The goal of extrinsic camera calibration is the calculation of the relative transformation between the head tracker and the cameras. The transformation can consist of both rotation and translation.*

Implementations of the *Direct Linear Transform* method described earlier was attempted, but was not finished due to problems related to normalization and the trivial solution. An attempt was also made on *calibrating both the intrinsic and extrinsic camera parameters using Zhang's method from OpenCV*. This was not finished due to problems related to accuracy discussed in the next section. Also, an implementation of the *Kalman filter* to predict future tracking positions was also started, but lack of time and priority caused it to be dropped.

Zhang camera calibration

Calibrating the cameras using Zhang's method to calculate both intrinic and extrinsic camera parameters is one way to find them. This approach is described in Algorithm 1. However, there are problems related to this approach which need to be addressed. The problems relate to accuracy and are further presented in the next paragraph.

Algorithm 1 was partially implemented, but not finished. The reason for this relates to the extrinsic camera parameters which could come out wrong. This is because the extrinsic parameters found by Zhang's method are only relative to the calibration pattern coordinate system, not the tracker coordinate system. In order to convert to the tracker coordinate system, the location of the calibration plane in tracker space has to be found. This is of course possible, and could be done by using a tracked pointer to sample the position of the plane corners in order to define a coordinate system in which the plane is located in 3D space. This sampling could be noisy, something which could be reduced by sampling the calibration pattern with the pen many times and projecting the coordinate axes into the plane. However, due to the nature of rotational errors in a camera model, any small mistake here will propogate and increase further away from the calibration pattern, as illustrated in figure (5.7). This makes it difficult to get an accurate position of the camera relative to a head tracker.

The calibration is performed from images collected of a chessboard pattern, as shown in figure (5.8). After collecting about 15 of these images in stereo or mono, the intrinsic camera calibration is performed, and this calibration can be applied to the Studierstube camera(s). The images collected from the cameras by OpenCV can easily be undistorted using internal functions in the library.

In order to make sure the extrinsic camera parameters come out accurately and not so much affected by errors in the sampling, Tsai's method could be used for the extrinsic parameter calibration, as described in the following section.

Tsai camera calibration

Tsai's procedure, as described in Algorithm 2, is in the need for a set of quintuples containing the pen position in the tracker coordinate system as well as the projected coordinate of the pen on the screen. The tracked pen was chosen as an input device instead of a calibration object because such calibration objects are costly, difficult to set up and have to be calibrated themselves. The sampling process consists of moving

Algorithm 1 : Zhang calibration of intrinsic and extrinsic camera parameters

$i \leftarrow 0$
while more images of calibration pattern to sample **do**
  $pattern \leftarrow sampleImageOfCalibrationPattern()$
  $corners \leftarrow findCalibPatternCorners(pattern)$ {Find corners at subpixel accuracy}

  {If all $corners$ were found in the image, save the corners along with the head tracker transformation. This transformation will be used to find the head tracker relative extrinsic camera parameters}
  if found $corners$ **then**
    $headtransforms[i] \leftarrow sampleHeadTrackerTransform()$
    $samples[i] \leftarrow corners$
    $i \leftarrow i + 1$
  end if
end while
$camparams \leftarrow calibrateCameraZhang(samples, i)$
$intrinsic \Leftarrow extractIntrinsic(camparams)$;
$extrinsic \leftarrow extractExtrinsic(camparams)$; {Pattern relative}
{Have to sample the calibration plane position and orientation in the tracker coordinate system in order to find the extrinsic camera parameters in the tracker coordinate system}
$i \leftarrow 0$
while need to sample tracker location on calibration pattern **do**
  $points[i] \leftarrow sampleTrackerPositionOnCalibrationPattern()$
  $i \leftarrow i + 1$
end while
$plane \leftarrow findOptimalPlane(points, i)$

{Now, sample the location tracker positions of origo, positive x-axis and positive y-axis of the calibration pattern}
$origo \leftarrow sampleCalibPatternOrigo(plane)$
$xaxis \leftarrow sampleCalibPatternXAxis(plane)$
$yaxis \leftarrow sampleCalibPatternYAxis(plane)$

{Generate a transform which takes us from pattern space to tracking space}
$patt2tracker \leftarrow generateTransform(origo, xaxis, yaxis)$

{Find the head-tracker relative extrinsic camera parameters}
$extrinsic \Leftarrow transformToTrackerCoord(patt2tracker, headtransforms, i, extrinsic)$

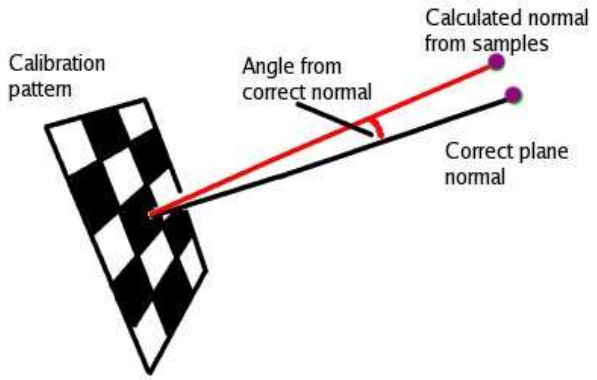

Figure 5.7: *Illustrates the propogative nature of rotational errors in the extrinsic camera parameters. The farther away the calibration pattern is, the distance between the real point on the plane normal and the calculated one increases in size. Calculating the camera position this way could lead to the camera center being wrongly placed.*

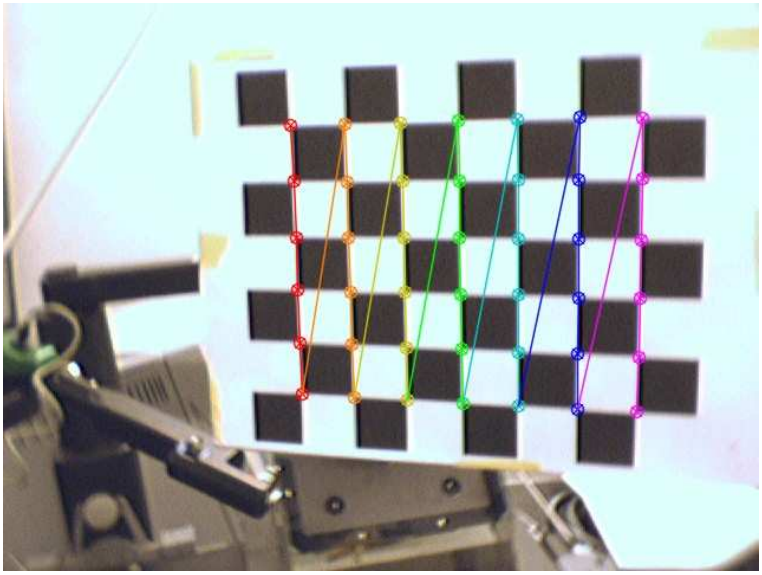

Figure 5.8: *This figure display the calibration pattern used for calibrating camera parameters using OpenCV. 10-20 images are captured during the procedure, and the corners in the pattern are automatically detected at subpixel accuracy*

Figure 5.9: *The tracked pen with a* 2×2 *chessboard pattern attached. The pattern has to placed with the center at the tip of the pen. Calibration is performed by slowly moving the pen in front of the camera while both the head and pen are being tracked.*

a tracked pen with a 2×2 calibration pattern pointing towards the camera being calibrated, as seen in figure (5.9). It is assumed the tracking system is very accurate. If the calibration pattern is found by image processing techniques, the tracker location is saved along side the image coordinate marking the center of the calibration pattern.

When enough samples has been gathered, the calibration process starts and the algorithms try to find the best camera model to fit the sampled data. After calibration is done, a SoOffAxisCamera is created based on the result.

Calibrating intrinsic and extrinsic parameters with different procedures

In some cases the calibration process is best split into two different parts: calculating the intrinsic camera parameters and calculating the extrinsic camera parameters. These parameters can be found by different calibration methods if wanted. Such a composite calibration method was implemented for this thesis by using Zhang's method to calibrate the intrinsic camera parameters and Tsai's method to calibrate the extrinsic camera parameters. This method is illustrated in Algorithm 3.

But, as noted in the section about accuracy, calibrating using different calibration methods could introduce errors due to the different algorithms being used to calculate the parameters. That is, the intrinsic and extrinsic parameters might not be compatible. This will become an issue if calibration accuracy is very important.

Calibrating a Studierstube SoOffAxisCamera

In order to be able to use the previously introduced calibration methods in Studierstube, a way of mapping the intrinsic and extrinsic parameters calculated by each procedure into a *SoOffAxisCamera* is needed. The problem revolves around to having a matrix for the intrinsic camera parameters as seen in equation (5.20), and values for the extrinsic

Algorithm 2 : Tsai's method for calibrating intrinsic and extrinsic camera parameters

$i \leftarrow 0$
while more images of the pen with the 2×2 calibration pattern to sample **do**
  $pattern \leftarrow sampleImageOfCalibrationPattern()$
  $corner \leftarrow findCalibPatternCorner(pattern)$ {Find the projection of the pen tip}
  if found $corner$ **then**
    $penpos \leftarrow samplePenTrackerPosition()$
    $headpos \leftarrow sampleHeadTrackerPosition()$
    $headrot \leftarrow sampleHeadTrackerRotation()$

    $relpos \leftarrow headrot^{-1} * (penpos - headpos)$ {Pen position relative to head tracker}

    $samples[i] \leftarrow (corner, relpos)$ {Save sampled quintuple}
    $i \leftarrow i + 1$
  end if
end while
$camparams \leftarrow calibrateCameraTsai(samples, i)$
$intrinsic \Leftarrow extractIntrinsic(camparams)$;
$extrinsic \Leftarrow extractExtrinsic(camparams)$;

camera parameters as seen in equation (5.21).

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{5.21}$$

How do these relate to each other? Taking the intrinsic camera parameters first, the proper way to map these to a SoOffAxisCamera is illustrated in Algorithm 4. The results of this algorithm makes it possible to set up a *SoOffAxisCamera* as follows[57]:

- **size** set to $size$ returned from Algorithm 4
- **position** set to $position$ returned from Algorithm 4
- **eyepointPosition** set to $(0, 0, 0)$
- **orientation** is set to identity rotation

Algorithm 3 : Zhang's method for intrinsic and Tsai's method for extrinsic calibration

{Find intrinsic parameters using Zhang's method}
$i \leftarrow 0$
while more images of calibration pattern to sample **do**
  $pattern \leftarrow sampleImageOfCalibrationPattern()$
  $corners \leftarrow findCalibPatternCorners(pattern)$ {Find corners at subpixel accuracy}

  if found $corners$ **then**
    $samples[i] \leftarrow corners$
    $i \leftarrow i + 1$
  end if
end while
$camparams \leftarrow calibrateCameraZhang(samples, i)$
$intrinsic \Leftarrow extractIntrinsic(camparams)$;

{Find extrinsic parameters using Tsai's method}
$i \leftarrow 0$
while more images of the pen with the 2×2 calibration pattern to sample **do**
  $pattern \leftarrow sampleImageOfCalibrationPattern()$
  $corner \leftarrow findCalibPatternCorner(pattern)$ {Find the projection of the pen tip}
  if found $corner$ **then**
    $penpos \leftarrow samplePenTrackerPosition()$
    $headpos \leftarrow sampleHeadTrackerPosition()$
    $headrot \leftarrow sampleHeadTrackerRotation()$

    $relpos \leftarrow headrot^{-1} * (penpos - headpos)$ {Pen position relative to head tracker}

    $samples[i] \leftarrow (corner, relpos)$ {Save sampled quintuple}
    $i \leftarrow i + 1$
  end if
end while
$camparams \leftarrow calibrateCameraTsai(samples, i)$
$extrinsic \Leftarrow extractExtrinsic(camparams)$;

Hence, a *SoOffAxisCamera* can easily be construced if the intrinsic camera parameters are available. The intrinsic camera parameters have all be converted into a Studierstube camera, but the conversion is not over yet. The extrinsic camera parameters have to be considered.

Algorithm 4 : Convert intrinsic parameters to a SoOffAxisCamera

Require: A to be a 3×3 matrix containing intrinsic camera parameters
Require: $width$ to be the width of video stream in pixels
Require: $height$ to be the height of video stream in pixels
Ensure: $position$ is the calibrated position of the image plane
Ensure: $size$ is the calibrated size of the image plane

{Find the coordinates over the image points on the image plane. The columns of the right matrix are homogeneous coordinates marking the boundary of the image in pixels}

$$Ipc \leftarrow A^{-1} * \begin{bmatrix} 0 & width & 0 & width \\ 0 & 0 & height & height \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

{Copy 2×4 values from the Ipc matrix into a new matrix. The values not copied are all 1's}
for $i = 0$ to 3 **do**
  $Imageplanecoords(0, i) \leftarrow Ipc(0, i)$
  $Imageplanecoords(1, i) \leftarrow Ipc(1, i)$
end for
{Find the center of the image plane}
$Posmat(0, 0) \leftarrow 1/4 \sum_{i=0}^{3} Imageplanecoords(0, i)$
$Posmat(1, 0) \leftarrow 1/4 \sum_{i=0}^{3} Imageplanecoords(1, i)$
{Subtract the center from the image plane coordinates, making the resulting image plane coordinates relative to origo}

$$Sizemat \leftarrow Imageplanecoords - Posmat * \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

{Extract position and size from matrices. The position of the image plane is always at a distance -1 in the z-axis. This is because the different focal lengths are compensated for in the size of the image plane.}
$position \Leftarrow (Posmat(0, 0), Posmat(1, 0), -1)$
$size \Leftarrow \mathbf{SVD}(Sizemat)$

The extrinsic camera parameters are, as you might remember from section (2.1.3), those which depend on the position and orientaton of the camera. Looking back at our camera model, the parameters related to position and orientation are:

- position
- eyepointPosition

- orientation

In the intrinsic camera calibration stage described earlier, these values were altered to match the camera parameters, but these values need to be changed in order to place the camera at the correct location in the head tracker relative coordinate system. This basically means that $eyepointPosition$ and $position$ have to be transformed to the correct locations, and $orientation$ has to be rotated in order to apply the extrinsic camera parameters. Basically, what we are trying to do was earlier written as equation (5.22), which is a rotation and translation followed by a projection.

$$s\tilde{m} = A[R|t]\tilde{M} \tag{5.22}$$

A problem with the *SoOffAxisCamera* camera model which becomes apparent now, is the fact that intrinsic and extrinsic camera parameters are not separated; changes to the intrinsic parameters affect the extrinsic and vise versa. This is an unattractive feature when performing calibration using separate methods, or when just the intrinsic or extrinsic camera parameters should be recalibrated. Still, this is the camera model currently implemented in Studierstube, and is also adopted for this thesis. Hence, the search for the calibrated *SoOffAxisCamera* continues. However, a new camera model more closely related to the logical separation of intrinsic and extrinsic camera parameters is a wanted feature in future versions of Studierstube.

Adding the extrinsic camera parameters cannot be directly performed as illustrated in equation (5.22). The extrinsic camera parameters have to be directly applied to the camera, and the needed transformation is the inverse extrinsic transformation. This transformation is applied to the SoOffAxisCamera by transforming the $eyepointPosition$, $position$ and $orientation$ as illustrated in Algorithm 5. The *SoOffAxisCamera* looks by default down the negative z-axis (y-axis goes up and x-axis to the right), and this has to be considered when applying the transformation to the camera.

Stereo camera calibration

The calibration framework implemented has also support for stereo camera calibration. In the case of calibrating by Zhang's method using OpenCV, two images will be captured while sampling instead of one. The calibration pattern has to be found in both images in order for the sample to be valid. When stereo calibration is performed, several interesting features are accessible through OpenCV. These include *image rectification*, *calculation of depth maps*, and much more.

Sampling stereo images has to be done with care. The cameras are most likely not synchronized with eachother, and movement between capturing the images is not a desired event. So, calibrating a stereo camera system should be conducted while having the HMD statically mounted when sampling.

When calibrating in stereo, the images can be rectified using builtin functionality in OpenCV. This is, however, a problem. The problem being that only the images

Algorithm 5 : Apply extrinsic parameters to a SoOffAxisCamera

Require: $[R|t]$ to be a 3×4 matrix containing extrinsic camera parameters
Require: $position$ is the intrinsic calibrated position of the image plane
Require: $orientation$ is the intrinsic calibrated orientation of the image plane
Ensure: $position$ is the calibrated position of the image plane
Ensure: $eyepointPosition$ is the calibrated position of the eyepoint
Ensure: $orientation$ is the calibrated orientation of the image plane

{Transform the $eyepointPosition$, $position$ and $orientation$ parameters of the camera. $R_x(\Pi)$ is needed in order to compensate for the fact that the *SoOffAxisCamera* looks down the negative z-axis. It is a transformation which makes it possible to map one camera-model to another. In this case, Tsai's model is mapped to a *SoOffAxisCamera*}

$$eyepointPosition \Leftarrow \left[\begin{array}{c|c} R & t \\ 0 & 1 \end{array}\right]^{-1} * R_x(\Pi) * \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array}\right]^T$$

$$position \Leftarrow \left[\begin{array}{c|c} R & t \\ 0 & 1 \end{array}\right]^{-1} * R_x(\Pi) * position$$

$$orientation \Leftarrow R^T * R_x(\Pi) * orientation$$

collected by OpenCV from the cameras are rectified, not the composite images of the virtual scene and the background which is needed in order to correctly register the virtual objects with the rectified scene. A way to solve this is to perform the rectification in OpenGL after rendering the scene to a buffer, and then applying the rectification and rendering the rectified images. This has not been implemented, but remains labeled "work to be done".

It is also possible to perform calibration using Tsai's method in stereo. This, however, is currently not a recommended approach because of the time it takes to search for the calibration pattern in two images in real-time.

Accuracy of calibration

As discussed in the procedure section, there are issues with calibrating intrinsic and extrinsic camera parameters using separate methods, but before this becomes the real issue, the calibration should give fairly accurate results. While the calibration of the intrinsic camera parameters using OpenCV remained accurate, the extrinsic calibration using Tsai's method introduced errors. These spawned from the sampling process; moving the tracked pen slowly in front of the camera and saving the head-tracker relative transformation was shown to not generally give good and reproducible results. This problem should be easy to mend by sampling only when both the head-tracker and the pen remain static. Finding the average of several samples over a time-interval should also help to solve the problem. However, when this problem is solved and if use of two calibration methods becomes a problem, calibrating both intrinsic and extrinsic

Figure 5.10: *The calibration process is performed in Augmented Reality and starts with this menu..*

parameters using Tsai's method could be implemented and should give good results if accurate sampling is achived.

User interface

As mentioned earlier, the calibration process is performed in Augmented Reality, making calibration an interactive process for both mind and body. The calibration starts with the menu seen in figure (5.10) and continues as illustrated in figure (5.11). By pressing buttons, the user chooses basic calibration options. For more specific control, the manipulation of an Open Inventor file is needed.

The calibration process by itself is quite easy to perform. One only has to do certain tasks which can easily be learned by anyone.

5.2.6 Discussion

Calibrating the extrinsic camera parameters with Zhang's method can be achieved if additional input for finding the head tracker relative transformation is found. Though not fully implemented, the procedure has a potential of becoming a fast and easy, but somewhat noise dependent, way of calibrating an HMD while in Augmented Reality. This because finding a plane in the tracker coordinate system can be performed with a fair amount of accuracy, and could be conducted in a fast and easy manner. Hence, this method could be used by inexperienced users not in the need of very accurate

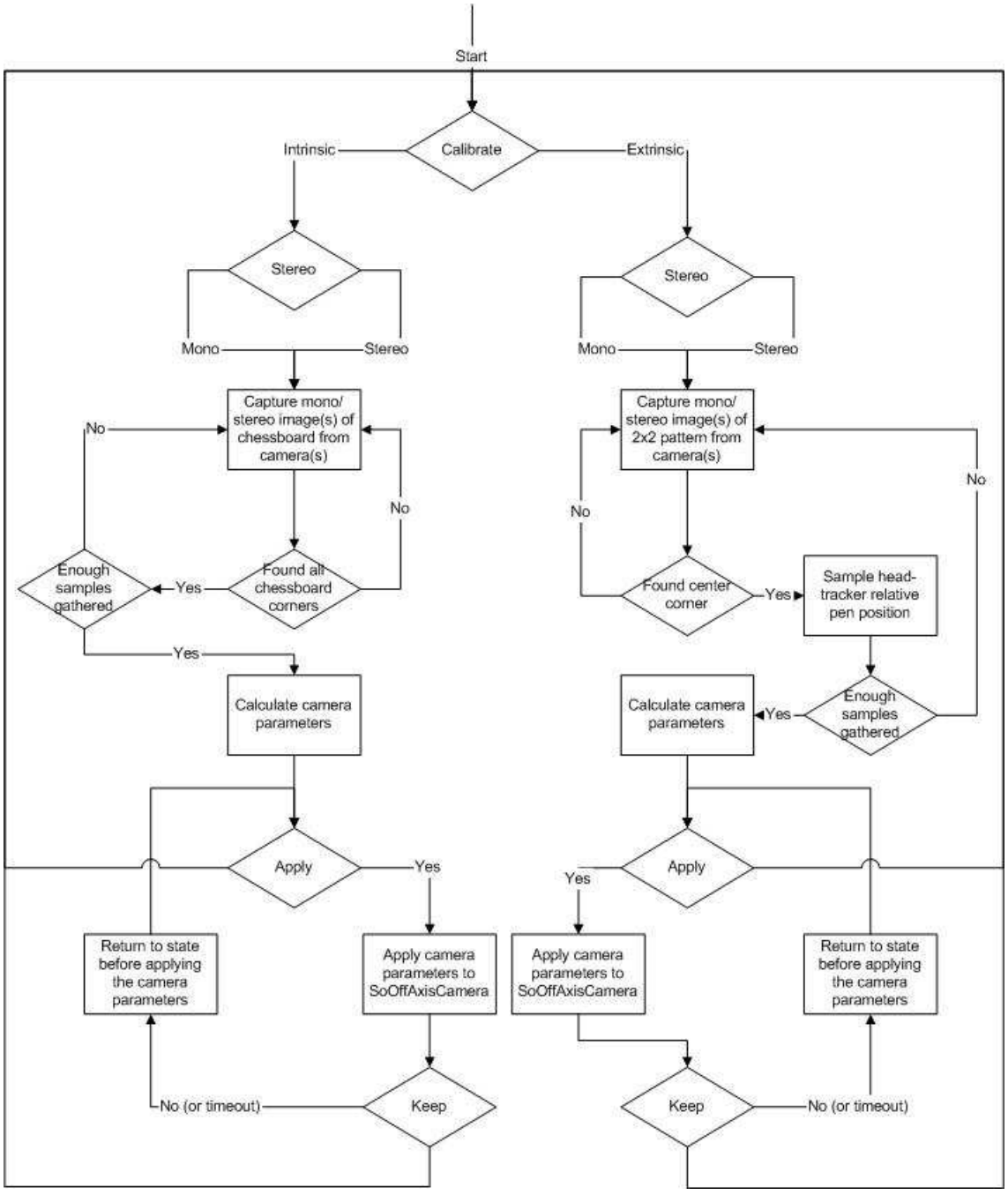

Figure 5.11: *The Augmented Reality application for calibrating intrinsic and extrinsic camera parameters follows a certain execution path which is illustrated in this figure..*

calibration.

Tsai's method for extrinsic camera calibration was adopted in the hope that it would lead to accurate results. Though Tsai's method is fairly accurate, the sampling of quintuples for the calibration process experienced noise. This was due to the unsynchronized nature of the tracking and image acquisition devices. It is very hard to sample accurate data when movement of head and other tracked devices is allowed during sampling. This is why static sampling is a desired feature if accuracy is important. However, if static sampling is not an option, using calibration methods less affected by noise is desired. One such method, based on genetic algorithms, was introduced in section (5.2.4).

It is noted by [58] that accurate calibration should be conducted using a calibration object whenever possible. The same accuracy is usually not otherwise achieved. These objects most often consist of two or three orthogonal planes. It is also possible to perform such a calibration using a plane undergoing known translation. These calibration objects are, on the other side, an added problem if a system is to work on multiple locations without having a calibration object available. However, when having a tracking system, 3D points can easily be attained and used for calibration. These should be sampled multiple times for each point in order to minimize the errors in the tracker data.

It was earlier mentioned that *SoOffAxisCamera* does not separate between intrinsic and extrinsic camera parameters. Having a camera model separating these is desired because it allows for an easier and more versatile calibration process, giving the user the choice of calibrating either the intrinsic or extrinsic parameters independently.

5.3 Dynamic registration

5.3.1 Overview

[2] End-to-end system latency is what Azuma[13] classifies as dynamic registration, and is the time it takes from the moment the tracking system registers the position and orientation of the HMD until the view is updated to correspond with the tracking information. The dynamic registration errors are not visible until the user changes the position or orientation of his head, but is still the largest source of registration errors. This type of error is observable when delay causes virtual objects to lag behind their real counterparts when the user moves his head.

The goal of dynamic registration procedures is to accurately predict the location of the users head at the time the displays are updated. There are two kinds of systems for handling dynamic registration errors, **open-loop systems**, and **closed-loop systems**. Open-loop systems do not have a feedback loop, so the accuracy of the technique depends on the tracking equipment and calibration. Closed-loop systems have feedback

[2]Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

mechanisms which make it possible to calculate the goodness of a solution to be able to minimize the error[61].

Tracking data received by sensors are usually full of error. Accurate positioning of objects is a difficult process that is in need of special, and often expensive hardware. Not only the positioning has to be correct, but also the timing - waiting several milliseconds for a tracking input to arrive cannot be compensated by the accuracy of the tracking signal. Also, real and virtual objects need to be registered to their counterparts. Procedures for compensating dynamic errors fall into four main categories[13]:

- Reduce system lag
- Reduce apparent lag
- Match temporal streams (with video-based systems)
- Predict future locations

Reduce system lag

System lag is what causes dynamic error, and by removing this source dynamic error will also be removed. This is, however, not an easy task. To reduce system lag, refresh rate of the monitor must be very high, and reducing the latency of a system will also reduce the maximum throughput. On current systems, this is not a practical solution due to the need for much faster hardware for processing the data.[13]

Reduce apparent lag

This can be done by a technique called “image deflection”. With this technique, the scene is rendered into a buffer which is bigger than the actual size needed to fill the screen. Then the most recent tracking coordinates are read just before the image is to be rendered to the screen, and the orientation values are used for translating the image to reduce the apparent registration error.[13]

Match temporal streams

When using video cameras for collecting images of the real world, the capturing and digitization processes cause a delay, but this could actually be utilized. The virtual scene could dynamically be transformed to match the captured images, causing the relative registration error to be removed. When the user moves his head, his movements will not match what he sees in his displays, so registration error is still present.[13]

Predict future locations

To predict future locations has shown to be a very promising technique for reducing the registration error, especially when the system lag is below 80ms. Predictors using inertial sensors have shown to reduce lag by a factor of 2-3. Predictors can make mistakes, but in general they are able to predict close to correct positions, making them very useful in Augmented Reality applications.[13]

5.3.2 Previous work

Hybrid tracking systems

Hybrid tracking systems utilizing inertial sensors to predict future position and orientation, can greatly reduce dynamic registration errors. Azuma has shown that hybrid tracking systems using inertial sensors are approximately 5-10 times more accurate than doing no prediction at all[9]. A problem with inertial tracker is the drift they experience as time goes on.

Using only tracking data from the tracking system gives no way to check the accuracy of the data, and without feedback on the accuracy, good registration is difficult to achieve[13]. Video-based approaches can be used for calculating the match between the worlds, but can be quite expensive with regards to processing. Patterns recognizable by a computer vision program could be used for tracking head and object locations. Even images taken of objects from different viewpoints could be used for recognizing and tracking objects.

Other possible sensors to help registration are depth sensors, e.g. laser rangefinders. They can create depth maps which can be matched with the depth maps of the virtual model, and thus improve registration. Magnetic trackers are also being used, but suffer from problems when metal is introduced into the environment.

If just one technology is to be looked at, optical tracking is probably the most promising one because of the increasing resolution of digital cameras, real-time photogrammetric techniques and structured light sources giving for more light at longer distances.[13] But hybrid tracking systems is probably the way of the future because a multitude of sensors can combine their strengths to overcome their weaknesses.

Computer vision based techniques

[61] describes a process of dynamically measuring 2D registration error which in turn improves registration in the combined images. It is noted that a similar error correction process will probably be a key component in future video-based Augmented Reality systems. By using video images for correcting registration error, a possible perfect registration is possible if the user can tolerate a certain delay from processing the images.

Robotics

For robots to autonomously be able to perform task and navigate, they need accurate information on where in the world they are located. This information can not only be gathered from GPS systems, due to their lack of accuracy. The problems related to the positioning of a robot can be divided into two main problem areas[48]:

- Relative localization
- Absolute localization

The relative localization information is gathered through a multitude of sensors, e.g. inertial sensors, to integrate the position and orientation from the initial position. The absolute localization is usually based on GPS systems, navigation beacons, etc. These systems are, unfortunately, very much corrupted with noise, and methods to handle this have to be used.

A promising way to correct noisy data from tracking sensors, is to use them in combination with other sensors, like inertial tracking devices, and computer vision based tracking[48]. [48] propose a hybrid tracking system based on accelerometers, gyroscopes, a compass, a video camera and a differential GPS. The position of the camera is calculated from known objects in the real world using line matching. They note that the process of finding the correct match is quite expensive.

Predictors

Azuma[9], suggests a setup where separate predictors are used for position and orientation. First the current position, velocity and acceleration are calculated. Then this data is used to calculate where the HMD will be at the next redraw. The predictor used for this is the Kalman filter, which is a highly popular filter for e.g. image processing. The filter also has a tendency to behave good even though data is received that does not meet the assumptions on which it is based. In order to do accurate predictions, the tracking data should preferably be timestamped[9]. This is, however, not commonly done, even by modern tracking equipment/software.

5.3.3 Problem statement

Magnetic Resonance scanners are highly sensitive to metallic substances in their vicinity, and will not function properly if this is the case. Also, most electronic equipment will not survive the extremely magnetic environment the MR-scanner causes. The magnetic fields could cause heating of wires, and induction of electricity within electrical and metallic devices[62]. There exist methods to create electronic devices that work in an MRI environment, but most tracking systems do not comply to this. Therefore, at the Interventional Center, an optical tracking setup is used. A hybrid tracking

system based on e.g. inertial integration is not so plausible without the correct equipment.

The problem of interest in this part of the thesis, is how to reduce noise from tracking data for more stable and accurate results. The approach used differs from other approaches in that it is to be used in a tracking framework named OpenTracker (which is used by Studierstube). It does not use the video information coming from the video-see-through HMD to reduce registration errors, this mainly because such calculations could reduce the performance of the system (though it probably will be an important way of reducing dynamic registration errors in future applications).

5.3.4 Procedure

Image deflection

Image deflection, as introduced in section (5.3.1), is used to reduce registration errors caused by change of orientation. This is done by rendering images that are bigger than the actual display to an offscreen buffer, and translating the image just before rendering to the display. The translation is performed using the newest tracking coordinates from the tracking system[13]. This procedure can give better apparent registration if the delay caused by rendering a bigger image can be tolerated. The method exploits the fact that rendering of scenes take time, and newer tracking signals will most probably be available when rendering is finished.

Video-based registration correction

Even though a video-based approach might not be a good solution for this project, some focus seems appropriate. The system developed uses video-see-through HMDs, which have both pros and cons. On the negative side, the video camera(s) and digitization hardware will cause even more delays in the system. On the positive side, video-see-through HMDs have the opportunity of reducing the dynamic registration errors through computer vision techniques.[13]

A problem with this approach is the assumption made of the virtual world being a close model of the real world. This is, however, not always the case. Often, the world is augmented, not copied. Virtual objects are placed in the real world, they do not replace the real world. Volume data contains lots of information which could be used for finding a good registration. However, when rendering the volume with different alpha-mappings, the outer details of the volume will disappear and the image is no longer usable for registration purposes. This could be solved by creating a surface approximation of the outer boundary of the volume which is rendered into a separate buffer and only used for registration purposes.

When using stereo video input, some thought also has to be put into how registration correction should occurs with relation to each eye in order to maintain the correct relationship between the eyes and the images.

Average filter

An averaging filter was implemented for testing purposes autumn semester 2004. Using an ultrasound tracking system, the position of a tracked static device was recorded for about ten seconds with and without the average filter, as seen in figure (5.12). The figures show the reduction of static errors when using an averaging filter, but when moving the device, as shown in figure (5.13), the averaging filter returns tracking coordinates lagging behind the tracked device coordinates, actually increasing the dynamic errors.

Because we want to reduce both dynamic and static registration errors, the average filter does not suffice. An other and more mathemathical approach is needed, which brings us to the next sections.

Bayesian filtering

Bayesian Filtering is a technique using probabilities for fusing data. Such a system is mathematically formulated and changes state as time goes on. The state is represented by the probabilities currently calculated by the system. This state can be utilized to predict future states. Such a predictor is called an **estimator**, and calculates an estimate of the system state with each observation provided to the system. The Kalman filter is a frequently used filter of this type. [59]

Kalman filter

The Kalman filter should already be known from static registration (section 5.2.4). The filter can, as mentioned, also be used to predict the future, which can be utilized for improving the dynamic registration. Though, it is assumed the end-to-end system latency is more or less predictable. If this is the case, a fairly accurate predicted position/orientation can be calculated using the filter.[63][9]

5.3.5 Results

Dynamic registration errors are very important and cause the biggest errors compared to static registration errors. However, good static registration is the basis for good dynamic registration, and has for this reason been more researched during this thesis.

It has been shown that the use of hybrid tracking systems based on intertial sensors can produce far better results than just using one tracking system alone, and the use of predictors could help reduce the problem even further. Looking at the future, using image-based methods will probably be a major part of handling dynamic registration in years to come.

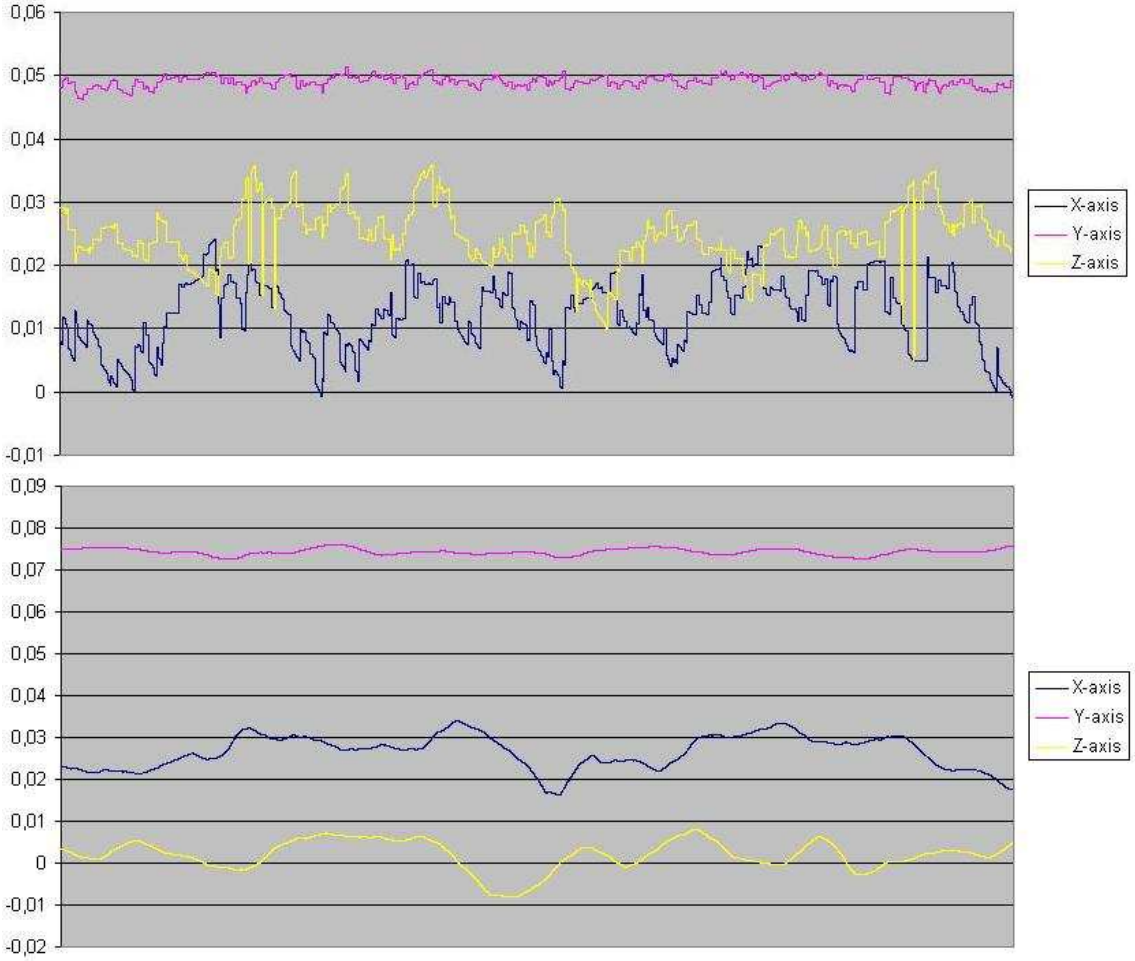

Figure 5.12: *The upper graph shows the reported tracking coordinates for an HMD tracked by a ultrasound tracking system. The device remains static (not moving), and the correct output would be a straight line for all axes, but this is not the case due to noise. The final graph shows the another output that is smoothed using an averaging filter. As seen, noise is reduced in this static case. Measurements are in meters, and sampling was performed for about ten seconds.*

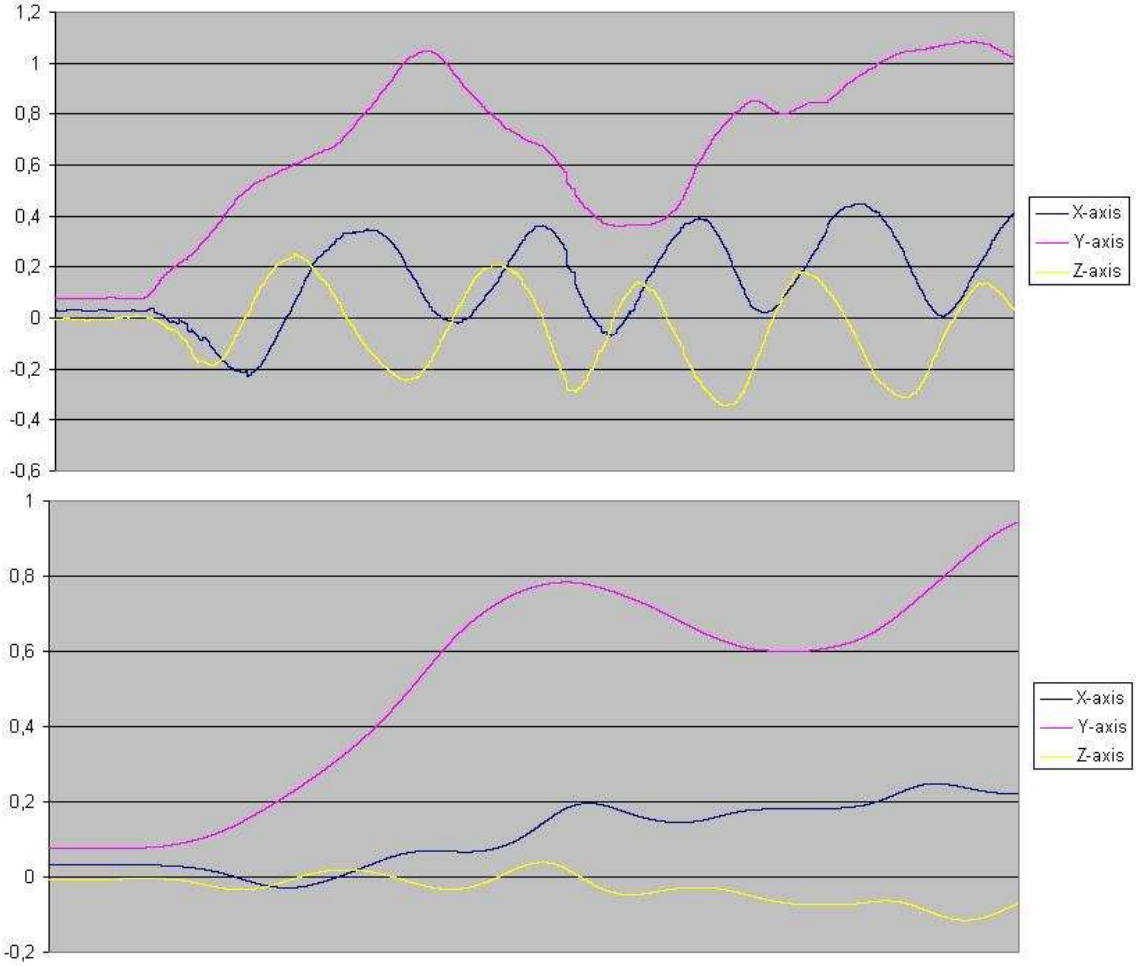

Figure 5.13: *The above graphs are two different views of the same tracking data. At the top, a raw dump of a device which is moved around and around is illustrated. Below, the same data is averaged to produce a smoother curve. It is apparent from this example that averaging is a low-pass filter which causes reaction delays, but produce smooth movement. Measurements are in meters, and sampling was performed for about ten seconds.*

5.3.6 Discussion

Not much work was conducted on implementing dynamic calibration procedures. Just a little time was spent trying to implement a Kalman filter to predict tracker coordinates, but this was never finished. The theory presented in the previous sections is meant as an introduction to problems and solutions involved in dynamic registration.

5.4 Registering a virtual object on a patient

5.4.1 Overview

Having calibrated both the intrinsic and extrinsic camera parameters, there are even more registration problems left to solve. If virtual objects are to be placed in the scene, they should be placed in relation to real objects. If the virtual object has a real counterpart, these have to coincide. There are many cases in which such a registration is important, e.g. the registration of an ultrasound image on top of a patient, performing breast biopsy using Augmented Reality, visualizing the part in a printer which needs to be changed, and many more. Wherever there are virtual objects which have to be registered with a physical ones, the registration problem is bound to appear.

5.4.2 Previous work

Registering hybrid tracking systems

Schwald et al.[64] describe a method for registering a virtual volume on top of a patient using a set of points. The method described is also used to correctly register tracking systems with eachother, as described in forthcoming section (5.5.2).

The registration starts with the placement of a set of small spherical markers on the patient. Then, a volume scan of the patient is acquired from either MR or CT scanning. The number of markers necessary is minimum four, and usually below ten, and they have to stay on the patient until the Augmented Reality system is used. After the scanning, the position of the spheres on the patient is sampled using a handle with a special electromagnetic sensor which is placed on the top of all spheres, one by one. The sensor has a concave tip which enables it to focus on the center of the spheres. To minimize the errors inherent in the tracking system, a number of samples are taken for each sphere position. Typically around 100 samples per sphere is used, and the result is the average of all positions. The spheres are sampled starting from the marker closest to the feet of the patient, and ending with the markers closest to the head, thus sampling is well-defined.[64]

Calculating the registration transformation is now performed by finding the rotation and translation which will transform the virtual volume on top of the patient in the tracker coordinate system. This problem is called the **absolute orientation problem**, and is described further in sections (5.4.4 and 5.5.2). The article does, however, not

mention how the positions of the spherical markers are found in the CT and MR data. Finding these could probably be performed automatically by special algorithms or manually by locating them in the volume. Results of the tests conducted by Schwald et al. show that high accuracy is achieved with their algorithm. The mean translation errors are usually below $2mm$; residing between $1mm$ and $2mm$. Though it is possible to find a registration just using four markers, having more markers will usually result in a better registration due to the measurement errors of the tracking system. As many as $8-10$ markers are proposed in order to get high accuracy and reproducible registration in a fast and easy manner. Improving the accuracy even further is said to be possible with the advent of new and better tracking systems and sensors. Also, the fusion of sensor data could help reduce the errors in the tracking system.[64]

Object calibration for Augmented Reality

In their publication, Whitaker et al.[65] describe an object calibration method based on **landmarks**. Landmarks being points placed in easily recognizable places such as corners and creases. The calibration procedure consists of locating the corresponding points on the virtual as well as real model. This is followed by the calculation of an object-to-world transformation based on the corresponding points. Two methods to perform the calibration are presented[65]:

- Image-based object calibration
- Pointer-based object calibration

Image-based object calibration assumes the camera has been calibrated and attempts to calculate the object-to-world transformation of a single and known geometric object. The calibration starts by capturing an image of the object and preprocesses it in order to find landmarks. The calibration approach described by Whitaker et al., does not make many assumptions about the scene, and does not need to. This is because manual intervention, by using a mouse pointer, is needed in order to find the landmarks. A rigid 3D transformation and a projection by a pinhole camera is assumed to map 3D world coordinates into image coordinates. It is noted that this is a simplification of the optics of a camera and does generally not apply.

The registration was solved as a minimization problem in a short amount of time, and the reprojection error was generally kept between $2-3$ pixels. However, even though the reprojection error is kept small, rotating the camera around the object could potentially show a problem. The errors in the z-direction when performing image-based calibraton can be very large. This **depth problem** was thought to be caused by the assumptions in the simplified camera model. Because of this problem, a method based on a tracked pointer was developed for object calibration.[65]

For **pointer-based calibration** to work, a pointing device capable of delivering world coordinates of its position is needed. The challenge is then to calculate the transformation between a set of 3D point pairs collected by the device. The world

points collected, P_i^w, are then used to calculate the transformation from object coordinates, P_i^l, to world coordinates. The relationship between P_i^w and P_i^l can be written as[65]:

$$P_i^w = P_i^l R + T \tag{5.23}$$

A linear system of twelve unknowns is created from equation (5.23). To solve this system, at least four points have to be sampled, and the least-squares error solution has to be calculated. This method of registering points with eachother can produce non-rigid transformations which are undesireable. Forcing the rotation to be rigid can be performed by a non-linear optimization problem[65]:

$$\min_x ||P_i^w - P_i^l R - T||^2 + \alpha ||RR^T - I||^2 \tag{5.24}$$

The procedure presented was able to calculate the object-to-world registration from the minimization problem of equation (5.24) in a short timeperiod.

It is noted that this method of registering, in contrast with the image-based method, does not have problems with depth, and that when working with large physical objects, the image-based approach is often better because it is more practical than the pointer-based approach. The pointer-based method is better when having a smaller object or when accuracy is desired.[65]

Non-rigid volume registration

Non-rigid volume registration is a very much bigger problem to solve than the previously mentioned rigid registration techniques. It is a well-known problem within image guided brain surgery that the brain actually "deflates" about one centimeter when opened. The cortical surface shifts of approximately 10mm, and subsurface shifts in the range of 6mm. This causes problems when trying to register pre-operative volume data correctly on top of the patient.[66]

The main causes of brain shift are[66]: the loss of cerebro spinal fluid, the usage of anaestetics and the work of the surgeon. This makes the preoperative and preprocessed images less useful for the neurosurgeon during the operation, and a better way of registering the preoperative volume data is needed.[66]

The non-rigid volume registration techniques can be separated into two main blocks: techniques using **image based models**, and techniques using **biomechanical models**. The image based models are often used if such images are possible to aquire during surgery.[66]

Transformations can be rigid, affine, projective and curved, and can be either global or local. Rigid transformations imply only rotation and translation effects are of concern. Affine also emphasizes the scaling and shearing effects which may occur. Projective transformations do not necessarily preserve parallel lines, and elastic transformations give the additional feature of being able to register lines with curves. Rigid and affine transformations are often characterized as global, since they affect globally.[67]

Non-rigid volume registration is a large and complex topic which will be researched for still many years. This section was meant as an introduction to the very important registration topic.

5.4.3 Problem statement

Calibration and registration of cameras and objects is an essential part of Augmented Reality. Without good registration, convincing demonstrations of the technology is not possible. Good registration enables effects such as occlusion and collision detection[65], and makes it possible to register a virtual object on top of a object with high accuracy.

A problem when dealing with volume data gathered from an MR-scanner, is the placement of the volume in world coordinates. Registering the volume with the patient is a must to accurately be able to perform surgery based on the data.

The problem to be solved in this part of the thesis is the registration of virtual objects on top of their real counterparts in a way which minimizes the error and can be performed fast and easily by a human operator.

5.4.4 Procedure

3D point registration

The registration could be performed using either image-based object registration or pointer-based object registration, as mentioned in section (5.4.2). Because of the depth and accuracy problem of the image-based method, the choice for this report is the pointer-based object registration.

When performing the sampling of points, the samples are bound to be off by some degree. A popular method for achieving this kind of object registration is the **Iterative Closest Point** (ICP) algorithm. It is an iterative procedure which requires an initial registration. For each step in the algorithm, two tasks are executed[68]:

- The closest neighbouring points are assumed to be the same points.
- The registration error is attempted minimized using a least squares algorithm which minimizes the distance error between the corresponding points.

This registration technique can be shown to converge towards a solution of minimal residual error, but the initial registration is of high importance for it to converge accurately and within a timeframe.[68]

The ICP algorithm is more often used when having a large number of samples, but registering a virtual object with a real should be performed by a human operator in a relatively short amount of time. Hence, having the operator sample many points is not desireable. Additionally, in contrast with the ICP algorithm, the mapping between corresponding points in the two coordinate systems is known prior to executing the algorithm. These properties of the problem indicate that another method for solving the

problem is needed. A simplified version of the ICP algorithm should to be investigated. But before that, some more information on the ICP algorithm seems appropriate.[68]

Iterative Closest Point algorithm

The Iterative Closest Point (ICP) algorithm is a popular method for aligning geometric models with eachother. The algorithm starts with an initial guess for the registration of the data, and iterates over the data until a good registration is found by minimizing an error metric.

Normally, the ICP algorithm has two sets of points as input, where the number of points in one set can vary from the number of points in the other set. The sets can be mathematically written as $P = \{p_i\}_1^N$ and $M = \{m_i\}_1^M$, where usually $N \neq M$. The problem, given the input parameters, is to compute the best transformation (rotation and translation) which aligns P with M. Equation (5.25) illustrates the transformation.[68]

$$m_i = Rp_i + t \tag{5.25}$$

The closest neighbouring points are assumed to correspond with eachother in the two sets[68]:

$$cp(p) = \min_{m \epsilon M} ||m - p|| \tag{5.26}$$

$cp(p)$ in the previous equation matches the closest point in the M set with the point p in the P set. Having this tool, the ICP algorithm can be expressed as follows[68]:

1. *Find a set of closest points. Can be achieved by an exhausting search, or by heuristical methods and smart algorithms.*

$$Y = \{m \epsilon M : p \epsilon P \wedge m = cp(p)\} \tag{5.27}$$

2. *Compute the optimized transformation from P to Y*

$$(R, t) = \min_{R,t} \sum_{i=1}^{N} ||y_i - Rp_i - t||^2 \tag{5.28}$$

3. *Apply the transformation to all points in P:*

$$p_i = Rp_i + t \tag{5.29}$$

4. *Continue at 1 if the stop criterion has not yet been satisfied.*

The algorithm stops as soon as any of the following conditions are satisfied[68]:

- The mean square error, equation (5.30), is low and below a threshold value.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} ||y_i - p_i||^2 \tag{5.30}$$

- The mean square error between two iterations is relatively small.
- The maximum number of iterations has been reached.

Step number two of the ICP algorithm is the focus of the remaining parts of the "Procedure" section, and is more generally known as the **absolute orientation problem**. For more information on the ICP algorithm, the interested reader is referred to documents ([69] and [70]).

Absolute orientation problem

In many applications, the need for finding the transformation between coordinate systems exist. Having two sets of feature measurements and finding the transformation between them is what constitutes the absolute orientation problem. There are many features which can be used, e.g. lines, surfaces, etc, but points are the most commonly used feature in practical applications. There are two general methods for solving these problems: **closed form solutions** and **iterative methods**, where the closed form solutions have shown to produce generally better results because they are not trapped in local minima.[71]

First it is assumed there coexists two point sets with corresponding points in each set, $\{m_i\}$ and $\{d_i\}$, where $i = 1, 2..N$. The point sets are related by the transformation[71]:

$$d_i = Rm_i + T + V_i \tag{5.31}$$

As suspected, R is a rotation matrix and T is a translation vector. These basic blocks define the transformation between the coordinate systems. V_i is a noise component which is related to each of the corresponding points in the sets. Finding the optimal rotation and translation components which minimizes the error of the correspondence very often requires minimizing a least squares criterion, as seen in equation 5.32.[71]

$$\epsilon^2 = \sum_{i=1}^{N} ||d_i - \hat{R}m_i - \hat{T}||^2 \tag{5.32}$$

However, it is noted that if outliers exist in the data sets, the use of this minimization is not optimal and other methods should be applied. There also exist several slightly different methods to calculate the optimization, but a choice was taken, and the method used in this thesis is that of Arun et al. (1987), and was designed to minimize equation (5.32).[71]

Equation (5.32) requires the point data to have the same centroid for the rotation part to work correctly. The centroids are defined as seen in equations (5.33, 5.34).

$$\begin{aligned} \overline{d} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} d_i \\ d_{c_i} &= d_i - \overline{d} \end{aligned} \tag{5.33}$$

And for the other point set:

$$\begin{aligned} \overline{m} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} m_i \\ m_{c_i} &= m_i - \overline{m} \end{aligned} \tag{5.34}$$

Basically, the centroid is just the mean of the point positions of all the points in the set. The points in the sets are then subtracted the centroid to find the relative positions d_{c_i} and m_{c_i}. This computation makes it possible to reduce equation (5.32) to equation (5.35).[71]

$$\begin{aligned} \epsilon^2 &= \sum_{i=1}^{N} ||d_{c_i} - \hat{R}m_{c_i}||^2 \\ \epsilon^2 &= \sum_{i=1}^{N} (d_{c_i}^T d_{c_i} + m_{c_i}^T m_{c_i} - 2d_{c_i}^T \hat{R} m_{c_i}) \end{aligned} \tag{5.35}$$

To minimize equation (5.35), the last term of the equation has to be maximized. This is equal to maximizing Trace$(\hat{R}H)$, where H is a correlation matrix defined by equation(5.36)[71]:

$$H = \sum_{i=1}^{N} m_{c_i} d_{c_i}^T \tag{5.36}$$

A singular value decomposition (SVD) is then performed on the correspondence matrix, H, which in turns yields a result of $H = U\Lambda V^T$. From this, the optimal rotation, $\hat{R}$ can be calculated as seen in equation (5.37).[71]

$$\hat{R} = VU^T \tag{5.37}$$

The solutions calculated by this method have been experienced to hold also in cases where both model and data points contain noisy data, which is just what is needed in our implementation of rigid registration. Thinking back to our problem, the markers have to be placed by human hand, and each point has to have a corresponding point in the other point set. Though errors should be fairly low, they will be present to some degree.[71]

Currently we have found an optimized rotation, but have yet to find an optimal translation. This translation is one which aligns the centroid of point set $\{d_i\}$ with the rotated point set of $\{m_i\}$. Mathematically, this is written as equation (5.38).[71]

$$\hat{T} = \overline{d} - \hat{R}\overline{m} \tag{5.38}$$

If the determinant of $\hat{R}$ is $+1$, there are no problems, but if the points in the data sets are planar, or large amounts of noise exists in the data, the determinant may become -1. This tells us there was a reflection instead of just a rotation conducted. Under these circumstances, the rotation matrix is found by calculating $\hat{R} = V'U^T$, where $V' = [v_1, v_2, -v_3]$ (values taken from V) has v_3 where the column corresponding to the singular value of H that is zero.[71]

5.4.5 Results

Overview

The registration of a point set with another point set is needed in order to correctly register a virtual objects on a patient. A clinical setting in which this is needed, is the registration of e.g. a volume data scan of a patient with the corresponding anatomy

of the actual patient. How such a registration can be performed mathematically was described in the procedure section, and will be refreshed in the Algorithms section coming up next. The registration is performed with a simplified version of the *Iterative Closest Point* (ICP) method - solving an issue called *the absolute orientation problem*. Two algorithms solving this problem have been implemented in Studierstube during the thesis:

- Register 3 points on a virtual object and 3 points on a patient after placing them in Augmented Reality .
- Register $N > 3$ points on a virtual object and N points on a patient after placing them in Augmented Reality.

The main difference between these two procedures is the underlying algorithm being used. The first procedure uses basic knowledge of vectors and matrices, while the last approach minimizes an error metric in order to calculate the transformation. Using just three points will make the transformation very noise-dependent, while the algorithm registering N points is able to handle noise much better. In any case, ensuring that as little noise as possible is present will help calculate a more accurate results.

Algorithms

Two calibration methods were implemented, however, the basic procedure to register a virtual object with a patient remains the same:

- Place a set of points on landmarks on the virtual object
- Place a set of points on the corresponding landmarks on the patient
- Press a register button which causes the registration to be calculated and applied to the virtual object.
- The virtual object should now be correctly placed on the patient unless there was noisy data.

Registering three points in one set with three points in another set can be done as seen in Algorithm 6. Registering two corresponding sets with N points in each is a more general process, taking advantage of algorithms developed to solve the *absolute orientation problem* previously mentioned. Performing this registration was implemented during this thesis and the registration procedure can be found in Algorithm 7.

Accuracy

The accuracy of the registration highly depends on the quality of the point sampling. If sampling is performed by moving the points using a 3D pointer in Augmented Reality,

Algorithm 6 : Three corresponding points virtual-real registration

Require: $M = \{m_i\}_{i=0}^{2}$ to be a set of non-planar points placed on a virtual object
Require: $D = \{d_i\}_{i=0}^{2}$ to be a similar set of non-planar points placed on a patient
Require: Both sets are located in the tracker coordinate system
Ensure: $T_{M \to D}$ is the transformation from M to D

{Create vectors in order to set up a coordinate system}
$a_1 \leftarrow normalize(m_2 - m_1)$, $a_2 \leftarrow normalize(m_3 - m_1)$
$b_1 \leftarrow normalize(d_2 - d_1)$, $b_2 \leftarrow normalize(d_3 - d_1)$

{Cross-products follow the right hand rule.}
$a_3 \leftarrow a_1 \times a_2$
$b_3 \leftarrow b_1 \times b_2$

{Do another cross-product to make sure the vectors really are orthogonal}
$a_2 \leftarrow a_3 \times a_1$
$b_2 \leftarrow b_3 \times b_1$

{Find the translation which takes us from M to D.}
$t_{m_1 \to d_1} \leftarrow d_1 - m_1$;

{The rotation is found from the coordinate systems previously calculated.}

$$R_a \leftarrow \begin{bmatrix} a_1[0] & a_2[0] & a_3[0] & 0 \\ a_1[1] & a_2[1] & a_3[1] & 0 \\ a_1[2] & a_2[2] & a_3[2] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , R_b \leftarrow \begin{bmatrix} b_1[0] & b_2[0] & b_3[0] & 0 \\ b_1[1] & b_2[1] & b_3[1] & 0 \\ b_1[2] & b_2[2] & b_3[2] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

{R_a and R_b are rotation matrices of the two corresponding coordinate systems. We now need to find the relative rotation which takes us from R_a to R_b.}
$R_{a \to b} \leftarrow R_b * R_a^{-1}$

{Set up translation matrices}

$$T_{m_1 \to d_1} \leftarrow \begin{bmatrix} 1 & 0 & 0 & t_{m_1 \to d_1}[0] \\ 0 & 1 & 0 & t_{m_1 \to d_1}[1] \\ 0 & 0 & 1 & t_{m_1 \to d_1}[2] \\ 0 & 0 & 0 & 1 \end{bmatrix} , T_{origo \to m_1} \leftarrow \begin{bmatrix} 1 & 0 & 0 & m_1[0] \\ 0 & 1 & 0 & m_1[1] \\ 0 & 0 & 1 & m_1[2] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

{The resulting transformation can now be calculated.}
$T_{M \to D} \Leftarrow T_{m_1 \to d_1} * T_{origo \to m_1} * R_{a \to b} * T_{origo \to m_1}^{-1}$

{All homogeneous points, $\tilde{p}_m$, located in M's coordinate system can be transformed to D's coordinate system by a simple matrix multiplication: $\tilde{p}_d = T_{M \to D} * \tilde{p}_m$}

Algorithm 7 : N corresponding points virtual-real registration

Require: $N > 3$
Require: $M = \{m_i\}_{i=0}^{N-1}$ to be a set of N points placed on a virtual object
Require: $D = \{d_i\}_{i=0}^{N-1}$ to be a similar set of N points placed on a patient
Require: Both sets are located in the tracker coordinate system
Ensure: $T_{M \to D}$ is the transformation from M to D

{Calculate the mean/centroids of both sets}
$\overline{d} \leftarrow \frac{1}{N} \sum_{i=0}^{N-1} d_i$
$\overline{m} \leftarrow \frac{1}{N} \sum_{i=0}^{N-1} m_i$

{Subtract the respective centroid from all points in M and D and place the resulting points in $\{m_{c_i}\}_{i=0}^{N-1}$ and $\{d_{c_i}\}_{i=0}^{N-1}$ respectively.}
for $i = 0$ to $N - 1$ **do**
  $d_{c_i} \leftarrow d_i - \overline{d}$
  $m_{c_i} \leftarrow m_i - \overline{m}$
end for

{H is a 3×3 correlation matrix}
$H \leftarrow \sum_{i=0}^{N-1} m_{c_i} * {d_{c_i}}^T$

{Decompose H by Singular Value Decomposition}
$U \Lambda V \leftarrow SVD(H)$

{Calculate the optimal rotation, $\hat{R}$}
$\hat{R} \leftarrow V * U^T$

{Calculate the optimal translation, t}
$\hat{t} \leftarrow \overline{d} - \hat{R}\overline{m}$

{Create a homogeneous transformation matrix, $T_{M \to D}$}
$T_{M \to D} \Leftarrow \left[\begin{array}{c|c} \hat{R} & \hat{t} \\ 0 & 1 \end{array} \right]$

{All homogeneous points, $\tilde{p}_m$, located in M's coordinate system can be transformed to D's coordinate system by a simple matrix multiplication: $\tilde{p}_d = T_{M \to D} * \tilde{p}_m$}

the quality of the samples could be bad. This is because it is very difficult to keep the hand steady enough when releasing the point near the desired location.

Currently, the placement of points on landmarks is performed in Augmented Reality for both the virtual object and the patient. Placing the points on the patient in Augmented Reality should not be difficult to do with a certain amount of accuracy because the patient is a physical entity with a surface on which landmarks can easily be found using a tracked pen.

To perform the registration, a set of easily identifiable points on the real patient is needed. In section (5.4.2), a method which used small spheres placed on the patient before MR or CT scanning was discussed. This technique is probably very accurate and should be tested in future implementations. A method which does not use spheres, but identifiable contours of the human body could also be implemented. The idea is to take an MR-image of the patient, find landmark points in the volume and then locate the landmark points on the patient. The points in the volume are found by looking at 2D slices from different angles.

In most cases, having more than four points could increase the precision. Iterative Closest Point implementations generally use hundreds of points to find a good approximated transform. However, having to place the points manually, using a number of points between eight and ten was said by Schwald[64] to give good results. Outliers should not be an issue in this context beacuse they are avoidable if sampling is carefully performed.

If accurate sampling can be assumed, there is no real limitation in the quality of the calibration except perhaps floating point accuracy. The method described in Algorithm 7 can be shown to converge towards the optimal solution.

User interface

The test-application developed does not need a real patient to be present. It registers two dummy-objects in 3D with eachother using a set of points which are moved by the user and placed on the corresponding locations on the two objects. Initially the points are placed in the scene as seen in figure (5.14). The blue points are to be placed on the virtual object, while the green points are placed on the real patient (which also is a 3D object in the case of the test-application).

A point is moved with the pen by pressing the pen button near the point, then relocating it to the desired position and releasing the pen button. The points change size when the pen enters their bounding volumes. This gives the process an intuitive feel and makes picking easier.

An example placement of the points is shown in figure (5.15). The locations do not have to be exactly the same as that of the reference model, but should be very close approximations if a good registration is needed.

After the points have been placed on corresponding locations on the virtual and real patient, the registration can be calculated. This is performed by clicking on the “Register” button located in the scene. The registration is then calculated and the

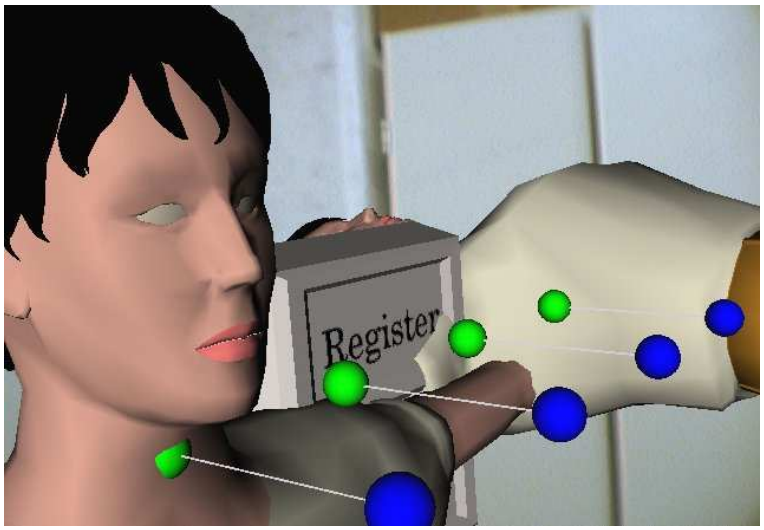

Figure 5.14: *Shows the setup currently used for registering a virtual and a real patient. The operator has to move the points manually in Augmented Reality using a pen. The blue points are placed on the virtual patient, and the green ones are placed on the real patient*

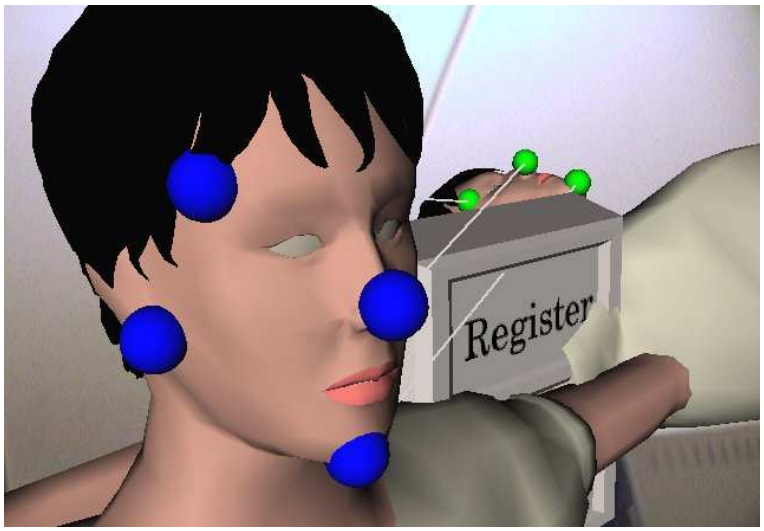

Figure 5.15: *Shows the points being placed on the virtual and real face, thus enabling a transform to be calculated and optimized. By pressing the "Register" button, the transformation will be calculated and applied - effectively registering the virtual object on top of the real*

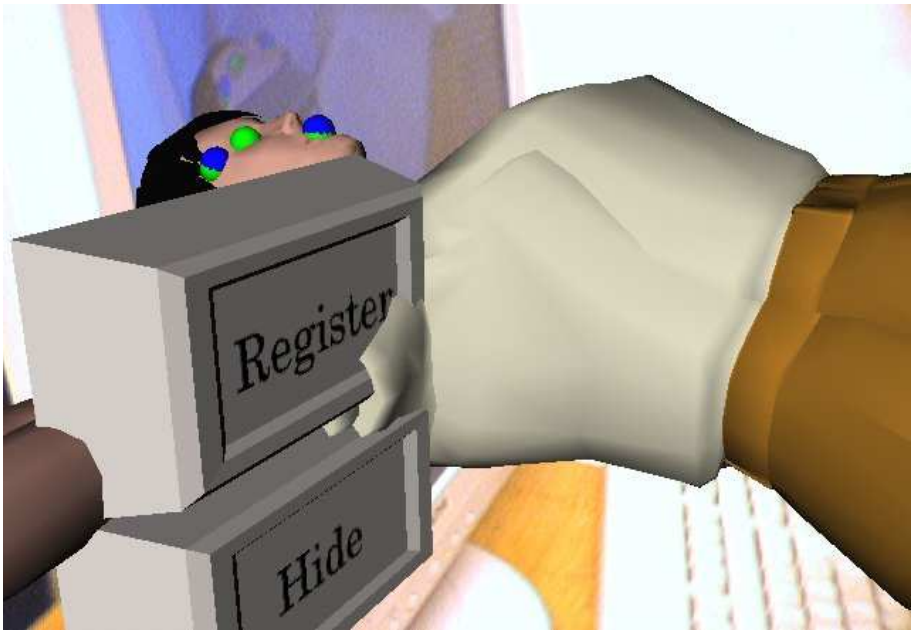

Figure 5.16: *The virtual object has been transformed to fit the real patient. The fit is not exact because the points have not been placed on the exact same relative locations in both models.*

virtual object is aligned (rotated and translated) to optimize the fit of the points with those on the patient. An example result with points not placed exactly correct is seen in figure (5.16)

If the calibration came out wrong, it is possible to move the control points again and the registration can be recalibrated, and hopefully improved upon.

5.4.6 Discussion

The method developed for registering virtual objects with real ones, has proved itself very general purpose and easily applicable. However, the way it was applied in this thesis is not how it will be used in a clinical setting. The surgeon will not have to place the points on the virtual objects in Augmented Reality. The reason why this is not a desired feature, is that accuracy suffers. It is much easier to place a point accurately on a 2D image with a mouse than in a 3D volume using a tracked pen. Further developments will have to address this issue.

Though two different methods to perform the virtual-real registration were introduced, only one is generally applicable. Being restricted to two sets with three corresponding points is a major limitation. Hence, the best method in most cases is the N point virtual-real registration algorithm.

5.5 Registering overlapping tracking systems

5.5.1 Overview

Overlapping tracking systems is here defined as tracking systems with overlapping tracking volumes. In these volumes, two or more tracking systems deliver tracking signals, but not necessarily in the same coordinate system or at the same time.

Having multiple overlapping tracking systems causes trouble because they, most likely, will not deliver tracking coordinates in the same coordinate system. Hence, a way to transform the tracking coordinates into a reference coordinate system is a desired feature, and will be discussed in this part of the thesis.

The previous sections dealt with the registration of virtual objects on top of a patient. This was performed by placing a set of points on both a virtual object and on the patient. As it turns out, fortune has favoured the brave - the virtual-real registration problem is related to the problem of registering overlapping tracking systems.

5.5.2 Previous work

Registering hybrid tracking systems

Schwald et al. proposes in [64] procedures for registering several tracking systems with eachother, making them produce tracking signals in the same coordinate system. The paper describes procedures for utilizing the capabilities of separate and overlapping tracking systems in a **hybrid tracking system for medical Augmented Reality**. The tracking components consist of a video-based infrared tracking system, and an electromagnetic tracking system. The optical tracking system is used for tracking the head of the surgeon as well as a semi-transparent display mounted on a mechanical arm, while the magnetic tracking system is used for tracking the other instruments and the patient. Several problems, including timing issues, are discussed.

Transforming from one coordinate system to the other ensures that all tracking signals are registered in the same coordinate system. The alignment process consists of moving an instrument, both magnetically and optically tracked, for about 20 seconds, or until enough samples have been gathered; typically around 250 are taken from each tracking system. When sampling is finished, the transformation between the coordinate systems is found and optimized. The rotation, R and translation, T, are found by minimizing equation (5.39).

$$err = \sum_{i=1}^{n} |RP_i + T - Q_i|^2 \tag{5.39}$$

This minimization problem is solved by maximizing $trace(RK)$, where K is defined as equation (5.40).

$$K = \sum_{i=1}^{n} \hat{Q}_i \hat{P}_i^T \tag{5.40}$$

P_i and Q_i are corresponding points from the electromagnetic and optical tracking systems respectively. $\hat{P}_i = P_i - \overline{P}$ and $\hat{Q}_i = Q_i - \overline{Q}$ are a set of points from P_i and Q_i with shifted centers to the centroids, $\overline{P}$ and $\overline{Q}$, of the point sets (average of all points in the sets) respectively. $\overline{P}$ and $\overline{Q}$ are defined to be: $\overline{P} = \frac{1}{n}\sum_{i=1}^{n} P_i$ and $\overline{Q} = \frac{1}{n}\sum_{i=1}^{n} Q_i$. The rotation part of the transformation from P_i to Q_i can be found by *Singular Value Decomposition* (SVD) of the matrix K:

$$K = VDU^T \tag{5.41}$$

$$R = VU^T \tag{5.42}$$

The optimal translation vector can now be calculated as:

$$T = R\overline{P} - \overline{Q} \tag{5.43}$$

All the corresponding points (P_i, Q_i) are then entered into equation (5.44) and the points are only used if the distance is below a threshold value. This ensures the **removal of outliers** caused by bad tracking data.

$$d = |P_i R + T - Q_i| \tag{5.44}$$

The algorithm is now run another time to ensure the bad values have no influence on the solution. The resulting rotation and translation after this second step is the final transformation.[64]

Having walked through the mathematics, the perceptive reader might have noticed the similarities between this procedure and the one used in section (5.4.4) to solve the absolute orientation problem. In fact, nearly the same algorithm was implemented when solving the virtual-real registration problem, and can be seen in previously introduced Algorithm 7.

Schwald et al. also tested an other way of calculating the transformation: nonlinear optimization with the Levenberg-Marquardt (LM) method. The algorithm resulted in a slightly different mean error, but the solution described above was chosen because of its simplicity.

A problem which will become appearent when working with different tracking systems, is the **time of which the samples are gathered**. The algorithm mentioned above assumes the corresponding points are gathered at exactly the same moment in time. This will, however, not be the case of most tracking systems. It may not even be the case that the tracking data is distributed in even intervals due to the thread priority of the computer or issues with the tracking system. These timing problems could be solved in this case by only sampling data when the trackers are completely still, but this is not a satisfactory solution in an interactive Augmented Reality application[64].

When motion is allowed, the usage of time stamps is needed in order to find compatible samples from the two tracking systems. This is done by finding the offset time, t_{offset}, between samples from the different systems. This value has to be within an epsilon time value, Δt. The ideal case would be to have $\Delta t = 0$, but this is essentially

impossible due to the synchronization issues. However, it is noted that Δt should be significantly lower than the update rate of the tracking system with the highest frequency. This coupling restricts the number of samples usable for registration. The Δt which gives the least error in equation (5.39) can be found by testing with different values of Δt.[64]

Other work

As mentioned in the overview, the problem of registering overlapping tracking systems is related to the problem of virtual-real registration from section (5.4). The problem also has many similarities with methods where the transformation between two sets of points is needed, as is the case when applying the *Iterative Closest Point* algorithm (ICP) introduced in section (5.4.4). Because of the power and popularity of the ICP algorithm, there are many implementations spanning different domains and problem areas utilizing the algorithm. It is up to the interested reader to investigate this topic further.

5.5.3 Problem statement

At the Interventional Center, there is an MR-scanner with an attached FlashPoint 5000 optical tracking system. In the central parts of the scanner, there is a magnetic tracking system which allows for tracking when occlusions cover the optical trackers. Also, it is possible to use more than one optical tracking system to further increase the possibility of at least one tracker being visible at a moment. Funds have been granted to the Interventional Center to get an additional optical tracking system, which will further increase the problem of having multiple tracking units working in union. To be able to use overlapping tracking systems together, they have to be registered to the same coordinate system.

The problem in this part of the thesis is the calibration and registration of overlapping tracking system so that tracking signals from all trackers are transformed into a reference coordinate system.

5.5.4 Procedure

Calculating the registration

Having two sets of corresponding points in different coordinate systems is visualized by figure (5.17). The basic idea behind the procedure, is to transform all points in one set so that the centroid of the set (average of positions of all points) becomes the origo. Now, the rotation has to be calculated between the points. This is calculated by solving the *absolute orientation problem* problem described in sections (5.5.2 and 5.4.4). A short review of the algorithm is seen in the following list:

- Find the centroids of both point sets.

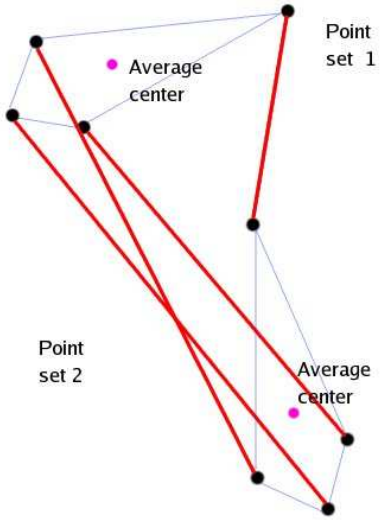

Figure 5.17: *Illustrates the problem of registering points from two different coordinate systems. All points in one set have corresponding points in the other set. A transformation between one set and the other is needed to properly register the sets with eachother. The goal is to calculate the transformation where the corresponding points, indicated by red lines, are located exactly at the same locations in a reference coordinate system*

- Subtract the controid from all points in the respective set from which it was calculated.
- Find the optimal rotation from one set to the other
- Find the optimal translation from one set to the other
- For all subsequent tracking signals which should be in the reference coordinate system, apply calculated transformation by matrix-vector multiplication.

There are several issues when working with two possibly heterogeneous tracking systems. One issue is the timing and rate at which the samples are returned. Another problem is the possibility of receiving bad tracker data. Handling such cases is important in order to achieve good and reproducible registration. These problems were introduced earlier in the document when referring to work done by Schwald et al. in section (5.5.2). The next sections will dive a little deeper into the timing and noise management issues.

Timing issues

When registering virtual objects with real ones or just two virtual sets of points with eachother, timing is not of concern. The positions are “timeless” - meaning they are

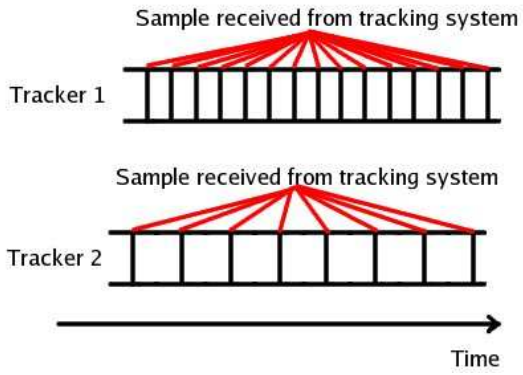

Figure 5.18: *Illustrates the timing issue when sampling from two tracking systems where the coordinates are time-dependent, meaning movement of the tracker is possible during sampling. The two tracking systems have different update rates. Thus, finding corresponding points from the two tracking systems is not a trivial task. Schwald et al.[64] notes that the time offset between corresponding samples should be very low, a time threshold significantly lower than the update rate of highest frequency tracker. This ensures the positions gathered are close enough in time to assume they have been sampled at the same moment*

not dependent on time information. However, when receiving a continuous stream of samples from the tracking systems, timing becomes an important issue. Especially when receiving tracking data from two or more devices for calibrating their coordinate systems. If motion is prevented when sampling, the timing does not become a problem. But if movement is allowed, timing becomes a relevant factor which requires some more thought. An illustration of the problem is found in figure (5.18).

One way of finding corresponding samples from the two tracking systems is to make sure the samples are **close in time**. Such a procedure was presented by Schwald in the "Previous work" section. Another theoretical approach is to perform some kind of **interpolation between two following samples** to approximate the tracking information at a specific time. The samples have to be close in time for this to work. Interpolation is conducted in such a manner that it predicts the signal sent by the other tracking system between the two samples. However, this setup requires a tracking system with low latency.

Handling noisy samples

Tracking systems have many sources of inaccuracies, and noisy measurements can easily occur. Investigating methods for handling these circumstances is necessary in order to achieve accurate registration.

The method introduced by Schwald et al. in the "Previous work" section handles outliers in the tracking signals by removing their influence on the resulting transforma-

tion. More details on how the registration could be done is found in the forthcoming "Results" section.

5.5.5 Results

Overview

To be able to register two overlapping tracking systems semi-automagically, the first task is to perform a sampling of a tracked device in both tracking systems. From this correspondence, the transformation between one tracking system to the other is calculated and optimized. One method for solving this problem has been implemented, and an other method is proposed:

- The implemented procedure interpolates between subsequent samples which are assumed to be close in time, followed by the calculation of the optimal transformation between the points. The functionality is implemented as an OpenTracker node.

- The proposed procedure matches samples that are close together in time and removes outliers in the sampling before calculating the optimal transformation. This method is potentially easier to implement as well as more accurate than the above procedure.

Algorithms

The current implementation of the registration procedure assumes the points from the tracking systems are located fairly close in time, and interpolates between subsequent samples. This method is described in Algorithm 8, and has problems when time between samples is relatively large, or when outliers are present in the data.

Using the method introduced by Schwald et al.: assuming two points are the same if they are sampled during a very short timeperiod, shorter than the update rate of the fastest tracking system, the timing problem can be alleviated. However, the exact position of the point at the reference time is not calculated, and the position is due to be off by a small amount. This method has not been implemented, but is described in Algorithm 9.

Accuracy

Accuracy depends on the quality of sampling. Only sampling static points makes it possible to get a high-quality registration assuming the tracking system delivers good tracking signals. Performing sampling with a moving device introduces problems which could reduce the accuracy.

Methods for handling dynamic devices were introduced in Algorithms 8 and 9. Even with these methods in place, one should be careful when sampling. Moving the

Algorithm 8 : Registering tracking system 1 with tracking system 2

Require: $N > 3$ to be the number of corresponding points to use for registration

$s_1 \leftarrow s_2 \leftarrow s_{1,prev} \leftarrow nil$
$interpolated \leftarrow false$
$i \leftarrow 0$
while $i < N$ **do**
  $sample \leftarrow waitForSample()$
  {Wait for sample from tracking system 1}
  if $s_{1,prev}$ = nil **then**
    if $sample$ comes from tracking system 1 **then**
      $s_{1,prev} \leftarrow sample$
    end if
    $continue$ {Continue while loop from top}
  end if

  if $sample$ comes from tracking system 1 **then**
    $s_1 \leftarrow sample$
    if s_2 is set and not $interpolated$ **then**
      {Linearly interpolate position and use SLERP for rotations}
      $samples[i] \leftarrow interpolate(s_{1,prev}, s_1, s_2.time)$
      $interpolated \leftarrow true$
      $i \leftarrow i + 1$
    end if
  else if $sample$ comes from tracking system 2 **then**
    $s_2 \leftarrow sample$
    $interpolated \leftarrow false$
  end if

  $s_{1,prev} \leftarrow s_1$
end while

{Calculate optimal transform from samples in tracking coordinate system 1 to tracking coordinate system 2}
$T_{1\rightarrow 2} \leftarrow calculateOptimalTransform(samples, i)$ {As done in Algorithm 7}

while $true$ **do**
  $sample \leftarrow waitForSample()$
  if $sample$ comes from tracking system 1 **then**
    $sample \leftarrow transformBy(sample, T_{1\rightarrow 2})$
  end if

  $updateObservers(sample)$ {Notify observers of new tracking coordinate}
end while

Algorithm 9 : Registering tracking system 1 with tracking system 2

Require: $N > 3$ to be the number of corresponding points to use for registration
Require: Δt to be the maximum time difference between corresponding samples
Require: ϵ_{max} to be the maximum distance between transformed and reference points in order to remove outliers

$s_1 \leftarrow s_2 \leftarrow nil$
$i \leftarrow 0$
while $i < N$ **do**
  $sample \leftarrow waitForSample()$
  if $sample$ comes from tracking system 1 **then**
    $s_1 \leftarrow sample$
  else if $sample$ comes from tracking system 2 **then**
    $s_2 \leftarrow sample$
  end if

  if s_1 and s_2 are set **then**
    $t_{offset} \leftarrow |s_1.time - s_2.time|$
    if $t_{offset} \leq \Delta t$ **then**
      $samples[i] \leftarrow (s_1, s_2)$
      $i \leftarrow i + 1$

      $s_1 \leftarrow nil$
      $s_2 \leftarrow nil$
    end if
  end if
end while

{Calculate optimal transform from samples in tracking coordinate system 1 to tracking coordinate system 2}
$(R, t) \leftarrow calculateOptimalTransform(samples, i)$ {As done in Algorithm 7}
for $j = 0$ to $N - 1$ **do**
  $\epsilon \leftarrow |R * samples[j].s_1.position + t - samples[j].s_2.position|$
  if $\epsilon > \epsilon_{max}$ **then**
    $samples \leftarrow samples - samples[j]$
    $i \leftarrow i - 1$
  end if
end for

$T_{1\rightarrow 2} \leftarrow calculateOptimalTransform(samples, i)$ {As done in Algorithm 7}

while $true$ **do**
  $sample \leftarrow waitForSample()$
  if $sample$ comes from tracking system 1 **then**
    $sample \leftarrow transformBy(sample, T_{1\rightarrow 2})$
  end if

  $updateObservers(sample)$ {Notify observers of new tracking coordinate}
end while

tracked device should be done slowly in order to ensure the samples from the different tracking systems are close in space. Having a high-frequency tracking system will also help reduce the error. Additionally, sampling should use a large number of points if noise is known to be present in the data. Typically $100 - 300$ corresponding samples are used.

Predictive tracking using e.g. the Kalman filter (introduced in section 5.2.4) could increase the accuracy of the registration. However, this has to be tested, and it is probable that the sampling of tracking coordinates which are close in time is a simplification which delivers valid results with low errors.

To sum things up, great registration is possible by sampling static points. This is, however, not always a desired feature. By sampling a moving device, accuracy will suffer if not carefully done.

User interface

The current implementation calculates the registration after having sampled a specified amount of coordinates. Because the registration is performed in OpenTracker, there is no Graphical User Interface (GUI). In fact, registration happens automatically as soon as enough samples have been matched from the two tracking systems. All the user has to do is move the tracked device slowly around while sampling occurs. After enough samples have been gathered, the registration transform is calculated and all tracking signals received from this point on from one of the tracking systems is transformed into the coordinate system of the other.

5.5.6 Discussion

Having just implemented one of the mentioned algorithms, testing the other one could show increased precision. The proposed method is also easier to implement since it does not need to interpolate between subsequent samples.

More testing is needed in order to find the quality of registration with the different approaches. If accuracy is not good enough, static sampling should be used. This is because static sampling does not have issues with inaccuracies caused by a moving device. However, sampling statically demands for user input, thus complicating the procedure.

Chapter 6

Visualization and Interaction

6.1 Motivation

Volume data has recently become increasingly important in many fields. The abilities of the newest graphics cards to perform real-time visualization of voxel graphics is probably the main reason for the growing interest. But still there are complexity and speed obstacles to overcome. Being able to visualize complex volumes at interactive rates in stereo is needed. Medicine has been using volume data for a long time, and research is focusing on newer and better ways for surgeons to utilize the new possibilities. Augmented Reality is an interesting field for researchers trying to figure out better and more accurate ways of displaying information, including volume data.[5]

6.2 Virtual cut

6.2.1 Overview

A virtual cut in the context of this thesis is a method giving the impression of a virtual object residing on the inside of a patient, though no surgical opening has been performed. Outside the cut the object is not visible, as illustrated in figure (6.1). This virtual cut technique is in sharp contrast with mesh cutting techniques dealing with how to simulate surgical cuts by cutting through meshes[83][84][85].

6.2.2 Previous work

Augmented Reality Visualization for Laparoscopic Surgery

Fuchs et al.[30] present an Augmented Reality prototype constructed to assist with laparoscopic surgical procedures. The system generates 3D structures from depth maps calculated from images captured by a laparoscope, and the resulting 3D mesh is textured with the same images.

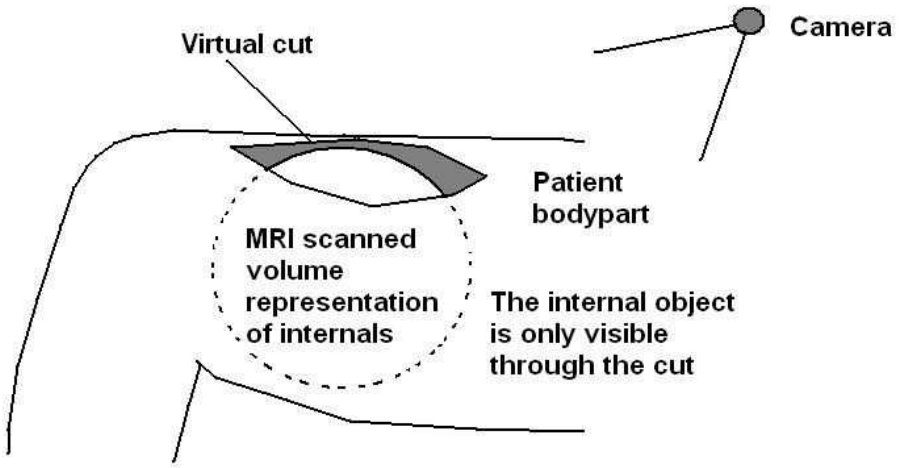

Figure 6.1: *As with a real cut, the internals of a patient are only visible through the cut, and not anywhere else*

The system is described to be intuitive and easy to use due its Augmented Reality nature. A helpful factor is also the way data is visualized; as if it were located inside the patient body when viewed from the outside. Compared to laparoscopic imagery on a video monitor, or even stereo laparoscopic imagery on a stereo display, it is noted that the Augmented Reality interface will be more intuitive and more powerful. An example is the movement of the laparoscope: The surgeon is often interested in specific areas and frequently moves the laparoscope. With the proposed interface, the movement of the instrument does not cause any disruptive changes in viewpoint which could cause confusion.[30].

Future work proposed by Fuchs et al. include the visualization of MR and/or CT data alongside the laparoscopic imaging, thus enabling a better view of the internals, possibly better than what would be experienced during open surgery. Some of the beneficial factors expected from the technology include[30]:

- Reduced average time for procedures
- Reduced training time of physicians
- Increased accuracy due to better understanding of physiology
- Better hand-eye coordination
- Reduced trauma to the patient
- Increased availability of procedures

6.2.3 Problem statement

Being able to see an augmented view of what is going on under the skin of a patient could greatly increase the accuracy of surgery, and also bring other beneficial factors as mentioned in section (6.2.2). However, a problem when rendering virtual geometry on top of a human body, is the perception that objects appear to float above instead of appearing on the inside. A virtual cut could help resolve this issue.

The problem at hand in this part of the thesis is the development of a virtual cut technique.

6.2.4 Procedure

A method for achieving a virtual cut in Studierstube is to render the virtual organs to a texture; then projecting this texture onto a mesh representation of the cut:

- Place the mesh representation where the cut is to be visualized on the patient.
- Render the virtual organs to a texture from the viewpoint of the camera.
- Map the generated texture onto the mesh representation of the cut.
- Render the entire scene with the textured virtual cut mesh.

The virtual organs will only be visible by looking at the mesh representation of the cut. How this could be implemented is described in the following "Results" section.

6.2.5 Results

Overview

The practical approach taken in order to solve the virtual cut problem is to render a virtual object, representing the internal organs, using the standard Coin node **SoSceneTexture2**. A geometric node (being the mesh representation of the cut) has to be created. Both the virtual object and the mesh have to be correctly registered with the body of the patient. This could be performed using the registration method developed in section (5.4). The texture is then mapped onto the mesh representation of the cut in such a way that the virtual object appears to be inside the patient. A possible result of this operation is seen in figure (6.2).

Algorithms

Before being able to see a virtual cut, certain operations need to have been performed:

- Calibrate the $camera$
- Register a virtual cut $mesh$ on the body of the patient. This could be performed as described in Algorithm 7.

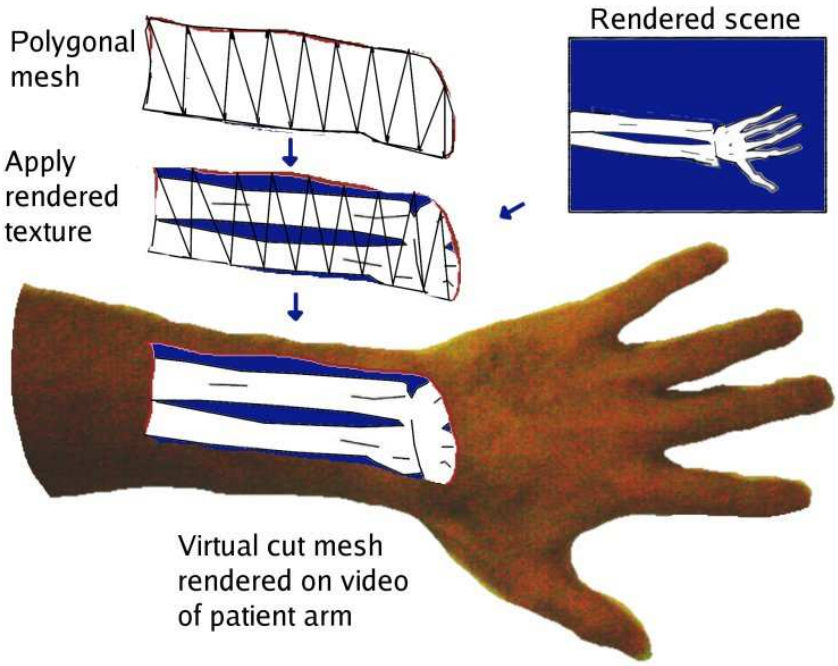

Figure 6.2: *The virtual cut technique is illustrated on an arm. The location of the camera in the tracking system is known, and a scene, representing the bone structure of the arm, is rendered to a texture using the same camera location. Having placed the virtual cut mesh in the scene, texture coordinates have to be applied. These are set to be the normalized screen projection coordinates of the 3D points.*

- Register the virtual $organ$ on the body of the patient. This could also be performed as described in Algorithm 7.

The mapping of the scene texture onto the cut mesh depends on what the camera sees. The texture coordinates for each point in the mesh are set to be the normalized screen projection coordinates of the 3D points. The operation is illustrated in figure (6.2), and described in Algorithm 10.

Algorithm 10 : Virtual cut

Require: A calibrated $camera$ attached to a tracked HMD
Require: The virtual cut $mesh$ has been registered with the patient
Require: The virtual $organ$ has been registered with the patient
Ensure: The virtual organ is rendered inside a virtual cut

for each frame **do**
{Render the virtual organ from the view of the camera and place the result in a texture}
$texture \leftarrow renderToTexture(organ, camera)$

{Finds the normalized screen coordinates of the mesh vertices}
$texcoords \leftarrow calculateNormalizedScreenProjection(mesh, camera)$

{Apply the generated texture to the mesh using the normalized screen coordinates as texture coordinates}
$applyTexture(mesh, texcoords, texture)$

{Render the texturemapped virtual cut mesh. It should now appear as if the virtual organ lies on the inside of a cut on the patient}
$render(mesh, camera)$
end for

Accuracy

The accuracy depends highly on how good the virtual organ has been placed relative to the patient. If the placement is precise, the virtual cut technique should not, theoretically, have many issues with accuracy.

However, the current technique used for mapping the rendered texture onto the mesh representation of the cut causes perspective correction to be applied to the texture, this leads to the texture being stretched and squeezed depending on the depth distance between polygonal points. This is an undesired sideeffect of the method. If it is possible to turn off texture correction in Coin, fixing this problem should not be a difficult task.

Figure 6.3: *This figure illustrates the virtual cut (blue area) as it would work on a real person. The patient has been replaced by a 3D avatar for illustrational purposes*

User interface

An implementation of the virtual cut, in this case placed on a virtual patient, can be seen in figure (6.3). In a real-life setting, the patient would of course be real, but the concept remains the same.

The technique by itself does not need specific user input. When correctly set up, the user should be able to see a virtual object registered inside the patient when looking at the cut mesh from different viewpoints.

6.2.6 Discussion

Having developed a virtual cut technique as described previously, there are a few issues which have not been discussed. It is not just important that the virtual cut is rendered with the correct camera, but lighting is also of importance. How should the organ be lit? No research has been conducted on this topic, and more work is needed in order to find answers.

The virtual cut technique has been visualized on a virtual patient. This is a much easier task than placing it on a real patient. Especially the registration of the virtual organ and the cut mesh with the patient is vital in achieving the aspired cut effect. More work is needed in order to get a satisfactory result in a clinical situation.

6.3 Volume transfer function specification

6.3.1 Overview

[1] Volume data can hold massive amounts of information. In most instances it is practically impossible for a human to comprehend an entire volume by looking at it directly. Hence, there is a need for tools which can manipulate the volume so that relevant parts can be visualized and enhanced with color and opacity. A typical tool of visualizing different parts of a volume and altering color mappings is a transfer function. This part of the thesis will deal with the transfer function specification problem, and builds on the work presented by the author in [5].

6.3.2 Previous work

StudyDesk

StudyDesk is a solution for presenting 3D-graphics interactively. The software foundation on which it is based is Studierstube. For volume visualization, SGI Volumizer is used. By moving a pen and a personal interaction panel (PIP), both of which are transparent and tracked objects, the user interacts with the volume data. Menus and other virtual objects are visualized on the PIP. The duality of pen and PIP makes two-handed interaction possible. [86] notes that two-handed interaction has proven very useful in medical applications. Some of the features included in the StudyDesk interactive volume exploration toolset are[86]:

- **Scaling the volume** is done by moving a sliderbar mapped onto the PIP. By scaling the volume, an increasing number of calculations have to be conducted in order to visualize the volume. This slows rendering down considerably.
- The **lookup table** is a way of controlling the colors and the opacity of the volume data. It allows for the display of data in a specific intensity range. A linear lookup table is used, and the linear curve is altered by adjusting mean and tolerance values by moving sliders on the PIP. The line goes from $min - tolerance/2$ to $mean + tolerance/2$. Additionally, a freehand procedure for drawing the transfer function was introduced.
- **Dragging the volume** is done by grabbing it with the pen, and moving it around and rotating it. This method of interaction is inspired by how we look at objects in the real world.
- **Cutting the volume** is done by flipping the PIP over. The menus will disappear and the transparent interaction panel can be used as a clipping plane. The PIP can then be moved over the volume data, and the volume is cut by the plane,

[1] Text partially taken from [5] - a project thesis conducted by the author autumn semester 2004

making it possible to see deeper into the volume. The two-handed interaction makes it possible to move the volume while also moving the clipping plane. This is a more natural way of interaction than just moving the clipping plane.

- **Freezing cutting planes** is done when the cutting plane is in a preferred position. One just has to click on the pen button, and the plane freezes on the volume.
- **Unfreezing cutting planes** is done by flipping the PIP back over in display mode; then clicking on the pen button.
- **Mirroring** is an interesting way of using the PIP. Flipping it over will have it working like a mirror.
- **Extracting arbitrary slices** is an important feature when used in medical applications.

The two-handed interaction as well as the above mentioned features were experienced to be very helpful. However, it was noted that powerful hardware is needed in order to render volume data in stereo.

Specification of transfer function

Some work has been conducted, using Studierstube, to find new and interactive methods for specifying the transfer function for volume data. The great importance of specifying a good transfer function is noted in [24], but the procedure is often performed on a trial and error basis. A new way of specifying such transfer functions was introduced in the same paper: The user chooses from a set of predefined transfer functions that can be manually adjusted. The transfer functions are modified by clicking on certain intensity values directly on a slice through the volume data. The intensity value chosen is then used as a peak value for an associated and predefined transfer function. Typical predefined transfer functions used are the box-shape, tent-shape, and gaussian shape. One or more transfer functions can be used together to specify a new and composited transfer function. This makes it possible to make certain regions of interest in the volume visible, while insignificant regions are kept transparent.

Previous research on transfer function specification concludes that it does not matter if the technique of choosing the transfer function is automatic, semi-automatic or manual unless it is **fast and simple**. It has been tested that the approach of StudyDesk; to let the user manually draw transfer functions, as with a pencil, turned out to be very difficult to do because of the trial and error type of input needed to find the correct transfer function. A new way of specifying the function is to choose an arbitrary point of interest within a volume, and a distance map is calculated from this point. The map influences the opacity of neighboring voxels. The paper only presents ways to interact with alpha-values, and did not touch color manipulation.[24]

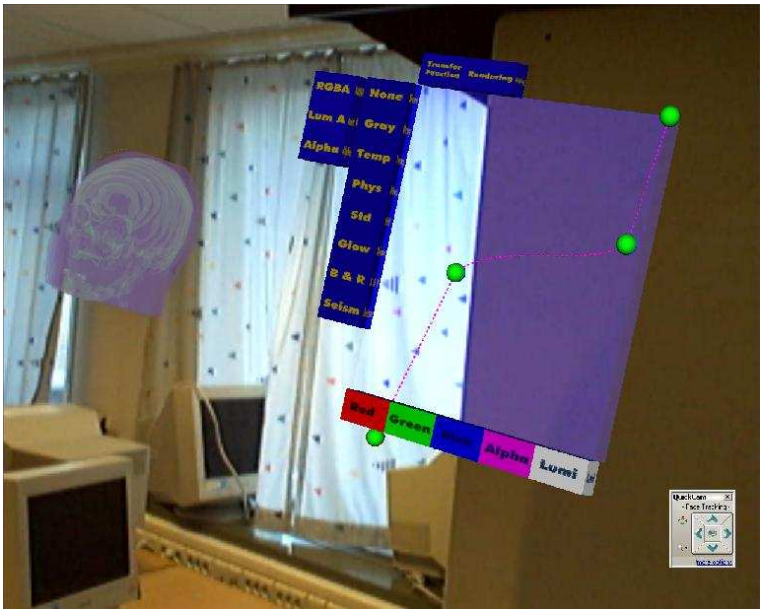

Figure 6.4: *3D GUI for transfer function specification. This screenshot displays the volume transfer function specification application developed autumn semester 2004 at NTNU.*

Work conducted at NTNU autumn 2004

A method for altering the transfer-function when displaying volume data has been developed in a project conducted autumn semester 2004. A screenshot of the resulting application can be seen in figure (6.4).

A special type of Hermite curves, called Catmull-Rom curves, were used to model the transfer function. The Hermite basis matrix is used for the calculation, but the tangent vectors are restricted to the constraints imposed by using Catmull-Rom curves[87]. Points along the curve are calculated using equation (6.1).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{bmatrix} \tag{6.1}$$

P_1 and P_2 are the current neighboring control points. The tangent vectors, R_1 and R_2, are what set Catmull-Rom curves aside from other Hermite curves, and they are defined as in equation (6.2).

$$R_i = \frac{1}{2}(P_{i+1} - P_{i-1}) \tag{6.2}$$

The reason Catmull-Rom curves were chosen, was because of their interpolation property, meaning the curves interpolate the control points, and because their tangent vec-

tors are predefined by the control points.

Undefined curve regions occur between the two first and two last points. The reason for this is that no valid tangent vectors can be found (see equation 6.2 why this is so). To make sure all segments have a defined curve, a linear interpolation between these points was used. The curve loses its continuity at two control points, but gains defined regions.

For the user interface to be easy, the control-points are geometric objects, e.g. spheres or cubes, and they change color and increase in size when the pen enters their bounding boxes. Manipulation is done just by dragging the geometric control points wherever wanted. No restrictions have been made to ensure the points stay within a valid domain. This is because creating a desired curve may demand the freedom to place points in undefined locations. The resulting undefined values are just clamped to be within the valid range.

6.3.3 Problem statement

Previous approaches to transfer function specification have shown that setting up the function has to be fast and easy, which was the background for developing a transfer function specification application autumn semester 2004. The function is specified by movable control points defining a Catmull-Rom curve. A problem present in the previous implementation is that very little is known about the intensity distribution within the volume, and presenting some of this information visually could help the transfer function specification process.

The problem at hand in this part of the thesis is the development of methods for displaying information about a volume helpful for fast and easy transfer function specification.

6.3.4 Procedure

Visualizing the transfer function using a histogram

The transfer function based on the Catmull-Rom curve is discretely sampled into a list of values. These values are used for altering the volume transparency and color. However, when the Catmull-Rom curve is steep, produces illegal values, winds back on itself, etc., the sampled transfer function will not entirely match the one visualized as a Catmull-Rom curve. Under these circumstances, it is very useful to see the transfer function actually being used. This is the motivation behind drawing a histogram of the transfer function together with the Catmull-Rom curve. More on this in the “Results” section.

Visualizing the distribution of volume intensities using a histogram

Altering the transfer function in real-time by moving control points defining a Catmull-Rom curve is a task requiring patience. In the old implementation, the only knowledge of the intensity distribution within the volume data was the visualization of the volume itself. This, however, does not help much in manipulating the transfer function. For this reason, a histogram visualization is desired in order to show the distribution of intensity values within the volume. The histogram will help the user in choosing a good transfer function by displaying important statistical information extracted from the volume data. More on the method can be read in the following "Results" section.

6.3.5 Results

Overview

Both histogram visualization procedures mentioned in previous sections have been implemented, and experiments show they have the potential of easing the process of setting up good transfer functions.

Algorithms

One of the histogram visualization methods implemented displays the sampled transfer function used for changing color and opacity values of the volume. The other iterates through a volume dataset and ends up containing the intensity distribution of the volume. Both methods are described in Algorithms 11 and 12.

Algorithm 11 : Histogram from a Catmull-Rom curve

Require: C to be a Catmull-Rom curve used for transfer-function specification
Require: N to be the number of samples used for transfer-function specification
Ensure: H is a histogram with N values sampled from the Catmull-Rom curve along an axis

$H \leftarrow nil$
for $i = 0$ to $N - 1$ **do**
  {Sample a value along an axis of the transfer function and place it in the histogram at the current sample index}
  $H[i] \leftarrow sampleCatmullRom(C, i)$
end for

User interface

The idea behind the transfer function, as displayed in figure (6.5), is that is should be easy to use and understand. The control points are draggers which can be moved

Algorithm 12 : Histogram from a volume

Require: N to be the number of distinct intensities in the volume
Require: V to be a volume containing scalar values between 0 and $N - 1$
Require: $(width, height, depth)$ to be the dimensions of the volume
Ensure: H is a histogram with N values based on the volume intensities. $H[i]$ is the number of voxels in the volume with i as intensity value.

```
H ← nil
for x = 0 to width do
  for y = 0 to height do
    for z = 0 to depth do
      intensity ← V(x, y, z)
      H[intensity] ← H[intensity] + 1
    end for
  end for
end for
```

around. The draggers increase in diameter when the pen enters their bounding boxes. Then, pressing the pen button inside a bounding box, attaches the dragger to the tip of the pen while the dragger decreases in size. It can now be moved within a restricted area, and it will detach from the tip of the pen when the user releases the pen button. The transfer function is altered in real-time as a dragger is moved, causing the volume data to also change while dragging. It is possible to change both alpha as well as red, green and blue color components and luminance. Predefined transfer functions can be used through clicking in the menu.

The histogram based on the Catmull-Rom curve continuously follows the curve; approximating it as best it can. The volume histogram is rendered in the same location as the Catmull-Rom histogram as seen in figure (6.5).

The new and old transfer function specifications methods can be compared in figure (6.6).

6.3.6 Discussion

The developed transfer function manipulator is quite easy to use, but a view of both the transfer function manipulator and the volume is still needed in order to find a good function. Hence, the user has to watch two different parts of the scene at the same time; potentially making the method difficult to use.

Another problem which is yet to be addressed, is the fact that the current method does not allow for adding or removing control points at runtime. Setting up the number of control points has to be done before the application is started. The author believes there are many circumstances where the adding or removing control points is a desired feature, hence it should be implemented if the method is to be used in a real application.

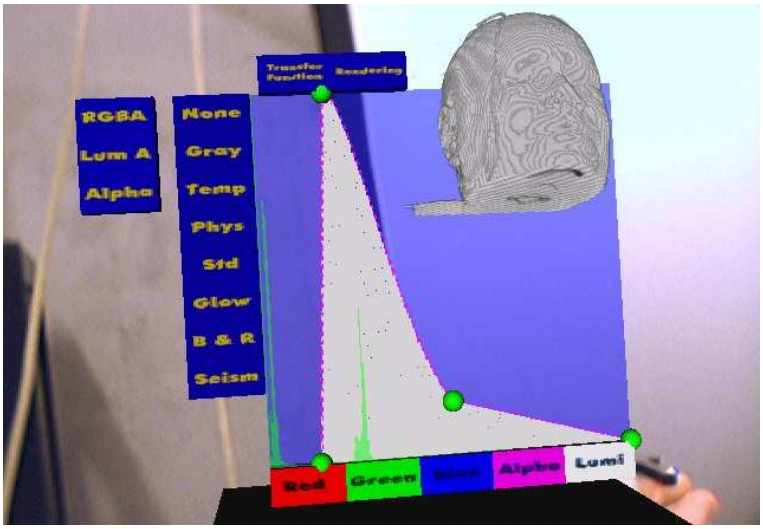

Figure 6.5: *The transfer function specification with the additional histograms displaying the current transfer function (white) and the intensity distribution in the volume (green).*

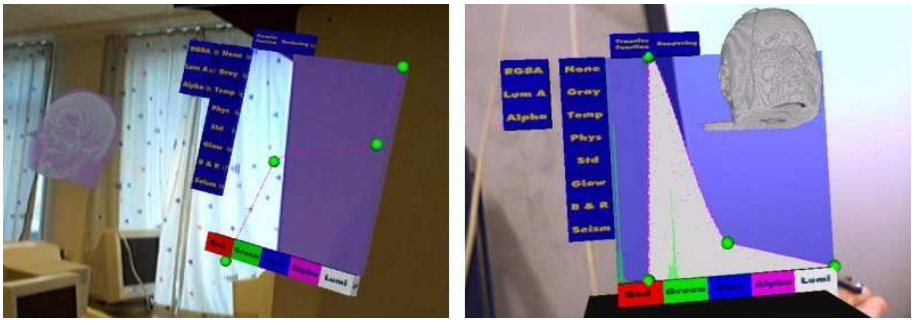

Figure 6.6: *On the left side is an image of the transfer function specification as it looked previous semester. On the right hand side, additional features have been added.*

Chapter 7

Discussion

One of the major difficulties in Augmented Reality, and the major topic of this thesis, is the accuracy problem. Combining the best of two worlds, Augmented Reality has the potential of becoming a useful medical tool in addition to becoming an ubiquitous part of our daily lives. But if the illusion of the real and virtual worlds existing together is compromised due to lack of accuracy, the user might not be satisfied by the system, and mass-acceptance will not occur.

The calibration and registration methods presented earlier in the document all experience accuracy problems. Most of these occur due to issues with the various sampling processes. The reason why sampling is such a problem is related to tracker accuracy, latency in capturing systems, refresh rates, and physics in general. The speed of computers is of major importance: Augmented Reality systems attempt to render a virtual scene on top of images of the real world in real-time. Due to the latency inherent in computer systems, and the dynamics of real world, the rendered images could be invalid by the time the scene is displayed, hence causing accuracy problems. Unless tracking systems, computers, sampling systems and algorithms are immensely improved, accuracy problems will continue to play a major role a long period into the future.

Making Augmented Reality a useful addition to the Image Guided Therapy toolbox is a relatively big task. AR is a major topic covering vast areas, and it is very important to keep the inter-disciplinary nature of the technology in mind when developing applications. Hence, results acquired through the years in other subjects like medicine, product design, psychology, art, user interface design, etc should be adopted.

This thesis has dealt with various problems in the field of Augmented Reality, touching many topics merely on a superficial level. During the work on this paper, many problems had to be solved, and even more problems were left unchallenged due to lack of time. The domain and possibilities of Augmented Reality are practically boundless, making the subject even more interesting. The author's hope for the future is that hardware usable with Augmented Reality will evolve rapidly and progress will be made in important areas, making the technology available and, more importantly, helpful and fun for most people.

Chapter 8

Conclusion

Augmented Reality is a natural way to interact with data, making it the preferred interaction methodology in many settings. AR enables new possibilities within the popular subject of Minimally Invasive Surgery, and is becoming an important tool within image-guided therapy in the near future. This thesis has investigated how Augmented Reality can be applied as a tool in such a setting. Focus was set on the registration and calibration issues, but problems related to the visualization and manipulation of data registered on a patient have also been investigated.

The main result of the work conducted, is a camera calibration procedure for use within Studierstube. The framework calculates the intrinsic and extrinsic camera parameters of video-cameras placed on a Head Mounted Display, and applies these to cameras in the Studierstube scenegraph; thus making the calibration process both fast and easy, but sensitive to noisy measurements. Several other parts were also constructed, e.g. the virtual cut technique; allowing the surgeon to see inside the body of the patient without cutting the body open, enhanced methods for altering the transfer function of volume data, and ways of registering coordinate systems with eachother. Several other tasks were also undertaken in order to achive the results presented in this document. The system developed during the thesis, though practical and easy to use, needs to be further revised, enhanced and added upon in order to become a useful part of the surgeons' daily routine in the MR-lab.

Progress will be made in the years to come which will further enhance the applicability of computer aided planning and intervention in surgery. The ARIS*ER project has just started and will continue several more years. What the final result will be is hard to imagine, but hopefully it will take Minimally Invasive Surgery one step further into the high-tech medical future.

"Reality is merely an illusion, albeit a very persistent one"
Albert Einstein (1879 - 1955)

Figure 8.1: *Yours truely, wearing an HMD*

Chapter 9

Future Work

Future work includes further developments in order to bring the system into a clinical setting. For this to be achieved, the calibration process has to be slightly modified; meaning that sampling of corresponding world and image points must be performed as accurately as possible. This can only be achieved if both head-mounted display and the tracked unit used for sampling are statically mounted in the tracker coordinate system during the sample acquisition process.

Algorithms for registering a virtual object on a patient have been implemented for this thesis, but even though the algorithms look promising, the user interface needs to be rethought. Placing the points on the virtual object in Augmented Reality causes accuracy problems and could result in an erroneous transform being calculated. Future work has to address this issue and implement an other method for sampling these points with high precision.

A method for finding the transform from one tracking system to another was previously presented in this document, but the method implemented should be improved upon due to possible accuracy problems. An algorithm which could resolve some of these issues was presented in the same sections, and future work should include the implementation of this and testing it against the old procedure.

When the static registration problem has been solved to a satisfactory level, dynamic registration should be further investigated. Some interesting methods have been introduced previously in the document, and among these, the Kalman filter, image deflection and video-based approaches look very interesting.

The virtual cut technique looks promising, but inaccuracies due to perspective correction were encountered, and this has to be fixed before the method becomes clinically applicable.

The volume transfer function specification, though relatively easy to use, had problems with the user having to look at both the volume and the transfer function at the same time. An alternative view of the volume while changing the transfer function could be visualized, thus slightly easing the specification process.

When accuracy and registration problems have been solved, the focus should be shifted to Human Computer Interaction, a subject of vital importance if the system is

to be adopted in real-world medical applications.

Bibliography

[1] *ARIS*ERnet: Augmented Reality in Surgery, Research Network for Minimally Invasive Therapy technologies*, 6th Framework Marie Curie Research Training Network, January 2005

[2] *Augmented Reality in Surgery*, `http://www.ariser.info`, May 2005

[3] Samset, E., *MRI-guided interventions: Technological solutions*, PhD Thesis, University of Oslo, `http://www.ivs.no/downloads/samset.pdf`, 2003

[4] Elle, O. J., *Sensor control in robotic surgery*, PhD Thesis, University of Oslo and the Norwegian University of Science and Technology, `http://www.ivs.no/downloads/elle.pdf`, 2004

[5] Karlsen, J. S., *Augmented Reality for MR-guided Surgery*, Project Thesis, Norwegian University of Science and Technology, `http://www.stud.ntnu.no/~jornskaa/prosjekter/thesis.pdf`, 2004

[6] SE NAC National Advisory Committee - *Glossary*, `http://www.semb.co.uk/reference/glossary.htm`, September 2004

[7] Wikipedia, *Minimally Invasive*, `http://en.wikipedia.org/wiki/Minimally_Invasive_Surgery`, June 2005

[8] Wikipedia, *Human computer interaction*, `http://en.wikipedia.org/wiki/Human_computer_interaction`, June 2005

[9] Azuma, R., Bishop, G., *Improving Static and Dynamic Registration in an Optical See-through HMD*, Proceedings of ACM SIGGRAPH, `http://www.cs.unc.edu/~gb/sig94AR.pdf`, 1994

[10] Rainer Splechtna, *Comprehensive Calibration Procedures for Augmented Reality*, Master's Thesis, Vienna University of Technology, 2003

[11] *Mixed and Augmented Reality*, `http://www-users.itlabs.umn.edu/classes/Fall-2004/csci8980-1/slides/augmented_reality.pdf`, October 2004

[12] Bauhaus University, Weimar, *Augmented Reality*, http://www.uni-weimar.de/~bimber/research.php, November 2004

[13] Azuma, R., *A Survey of Augmented Reality*, Hughes Research Laboratories; Teleoperators and Virtual Environments, http://www.cs.unc.edu/~azuma/ARpresence.pdf, 1997

[14] de Landgraaf, W. A., *Interaction between users and Augmented Reality systems: Human-Computer Interaction of the future*,http://www.alextreme.org/docs/paper-hci.pdf , Vrije Universiteit Amsterdam, 2003/2004

[15] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B., *Recent Advances in Augmented Reality*, IEEE Computer Graphics and Applications, http://www.cs.unc.edu/~azuma/cga2001.pdf, 2001

[16] Weidenbach, M., Wick, C., Pieper, S., Redel, D. A., *Augmented Reality in der Echokardiographie*, Zeitschrift fűr Kardiologie, Steinkopff Verlag, Mářz 2000

[17] Reading Surgical Associates, *Laparoscopic surgery*, http://www.rsapc1.com/laparoscopic_surgery/, October 2004

[18] Schnaider, M., Schwald, B., Seibert, H., Weller, T., *Medarpa - A Medical Augmented Reality System for Minimal-Invasive Interventions*, ZGDV, 2002

[19] Brügge, B., MacWilliams, A., Reicher, T., *Study of Software Architectures for Augmented Reality Systems*, Technische Universität München, http://wwwbruegge.in.tum.de/publications/includes/pub/bruegge2002archstudy/bruegge2002archstudy.pdf, 2002

[20] Azuma, R., *Augmented Reality*, http://www.cs.unc.edu/~azuma/azuma_AR.html, May 2005

[21] Fuhrmann, A., Schmalstieg, D., Purgathofer, W. *Fast Calibration for Augmented Reality*, Proceedings of the ACM symposium on Virtual reality software and technology, http://www.cg.tuwien.ac.at/TR/99/TR-186-2-99-16Paper.pdf, 1999

[22] Gonzalez, R. C., Woods, R. E., *Digital Image Processing, second edition*, Prentice Hall, 2002

[23] Kaufman, E. A., *Introduction to Volume Graphics*, http://www.cs.duke.edu/courses/spring03/cps296.8/papers/KaufmanVolumeGraphics.pdf, November 2004

[24] Reitinger, B., Zach, C., Bornik, A., Beichel, R., *User-Centric Transfer Function Specification in Augmented Reality*, Journal of WSCG, http://liverplanner.icg.tu-graz.ac.at/modules/Publications/2004/reitinger_wscg2004.pdf, 2004

[25] Scott, A., *Interventional and Intraoperative Magnetic Resonance Imaging*, Alberta Heritage Foundation for Medical Research, http://www.ahfmr.ab.ca/hta/hta-publications/infopapers/ip17.pdf, 2004

[26] Ottermo, M. V., Stavdal, ., Johansen, T. A., *Palpation Instrument for Augmented Minimally Invasive Surgery* http://www.itk.ntnu.no/ansatte/Johansen_Tor.Arne/PalpInst.pdf, 2003+

[27] Stoyanov, D., ElHelw, M., Lo, B. P., Chung, A., Bello, F., Yang, G. Z., *Photorealistic Visualization for Virtual and Augmented Reality in Minimally Invasive Surgery*, Imperial College, London, http://www.doc.ic.ac.uk/~benlo/danail%20paper.PDF, 2003+

[28] Wayand, W., *The History of Minimally Invasive Surgery*, http://www.bbriefings.com/pdf/952/Wayand.pdf, 2005

[29] *UNC Laparoscopic Visualization Research*, http://www.cs.unc.edu/Research/us/laparo.html, May 2005

[30] Fuchs, H., Livingston, M. A., Raskar, R., Colucci, D., Keller, K., State, A., Crawford, J. R., Rademacher, P., Drake, S. H., Meyer, A., A., Augmented Reality Visualization for Laparoscopic Surgery, http://www.cs.unc.edu/~fuchs/publications/AugRealVis_LaparoSurg98.pdf, 1998

[31] Rogers, E., *Introduction to Human-Computer Interaction (HCI)*, California Polytechnic State University, http://www.cas.kth.se/ras-ifrr-ss04/material/rogers-hci-intro.pdf, 2004

[32] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Care, T., *Human-Computer Interaction*, Addison-Wesley, 1994

[33] U.S. Department of Health and Human Services, Usability Basics, http://usability.gov/basics/index.html, May 2005

[34] *OpenGL - Wikipedia*, http://en.wikipedia.org/wiki/OpenGL, May 2005

[35] *Coin3D developer page*, http://www.coin3d.org/, May 2005

[36] *Introduction to Studierstube concepts*, http://studierstube.org/doc/stb/intro.html, october 2004

[37] Zhang, Z., *A Flexible New Technique for Camera Calibration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, http://research.microsoft.com/~zhang/Papers/TR98-71.pdf, 2000

[38] *Open Source Computer Vision Library Reference Manual*, http://www.cs.unc.edu/Research/stc/FAQs/OpenCV/OpenCVReferenceManual.pdf, May 2005

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

[40] Reitmayr, G., Schmalstieg, D., *An Open Software Architecture for Virtual Reality Interaction*, Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST-01), http://www.ims.tuwien.ac.at/media/documents/publications/opentrackervrst2001final.pdf, 2001

[41] Reitmayr, G., *On Software Design for Augmented Reality*, PhD Thesis, Vienna University of Technology, http://www.ims.tuwien.ac.at/media/documents/publications/reitmayr_diss.pdf, 2004

[42] Clark, T. A., Fryer, J. G., *The Development of Camera Calibration Methods and Models*, Optical Metrology Centre, http://www.optical-metrology-centre.com/Downloads/Papers/Photogrammetric%20Record%201998%20Camera%20calibration.pdf, 1998

[43] Fiala, M., *Automatic Extraction of Radial Distortion Parameters*, http://www.cs.ualberta.ca/~fiala/radfind.htm, October 2004

[44] Sonka, M., Hlavac, V., Boyle, R., *Image Processing, Analysis and Machine Vision, second edition*, Thomson Engineering, 1998, pp441 - 553

[45] Bengtsson, B. D., *Augmented Reality for Safer Coronary Bypass*, Master's Thesis, University of Oslo, http://www.ivs.no/downloads/bengtsson.pdf, 2003

[46] Lenz, R. K., Tsai, R. Y., *Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology*, IEEE Transactions on Pattern Analysis and Machine Intelligence, http://ece.uic.edu/~ssankar1/sriram/papers/20.pdf, 1988

[47] Manthey, D., *General Least-Squares - Direct Solution and Bundle Adjustments*, http://www.orbitals.com/self/least/least.pdf, 2003

[48] Persa, S., Jonker, P., *Real-Time Computer Vision System for Mobile Robot*, Delft University of Technology (Netherlands), 2001

[49] Svoboda, T., *Quick Guide to Multi-Camera Self-Calibration*, Swiss Federal Institute of Technology, Zurich, http://cmp.felk.cvut.cz/~svoboda/SelfCal/Publ/selfcal.pdf, 2003

[50] Li, Q., Zhang, Y., *Camera Calibration with Genetic Algorithms*, IEEE Transactions on Systems, Man and Cybernetics, 2001

[51] Li, F., Zhang, Q., Klette, R., *Accuracy Improvement in Camera Calibration*, The University of Auckland, http://sprg.massey.ac.nz/ivcnz/Proceedings/IVCNZ_61.pdf, 2003+

[52] Heikkila, J., *A Four-step Camera Calibration Procedure with Implicit Image Correction*, University of Oulu, http://www.vision.caltech.edu/bouguetj/calib_doc/papers/heikkila97.pdf, 1997

[53] Memon, Q., Khan, S., *Camera calibration and three-dimensional world reconstruction of stereo-vision using neural networks*, International Journal of Systems Science, http://www.cs.ucf.edu/~khan/p1155_s.pdf, 2001

[54] Roger Tsai's Calibration algorithm download http://www-2.cs.cmu.edu/~rgw/TsaiCode.html

[55] Gonzales, J. I., Sosa, J. D. H., Brito, A. C. D., Cabrera, A. M. N., *Comparative Analysis of Calibration Methods for a Static Camera*, Universidad de Las Palmas de Gran Canaria, http://mozart.dis.ulpgc.es/Gias/Publications/calibration-eurocast2005.pdf, 2005

[56] *StbDoc - Studierstube: SoOffAxisCamera Class Reference*, http://studierstube.org/doc/stb/classSoOffAxisCamera.html, June 2005

[57] *artoolkit2offaxis.m located in the Studierstube framework*, http://www.studierstube.org, May 2005

[58] Elgammal, A., *CS 534: Computer Vision Camera Calibration*, Rutgers University, 2005

[59] Australian Centre for Field Robotics *Bayesian Filtering Classes*, http://www.acfr.usyd.edu.au/technology/bayesianfilter/BayesianFilteringClasses.htm, February, 2005

[60] Kalman, R. E., *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME - Journal of Basic Engineering 82, http://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf, 1960

[61] Bajura, M., Neumann, U., *Dynamic Registration Correction in Video-Based Augmented Reality Systems*, IEEE Computer Graphics and Applications, http://graphics.usc.edu/cgit/pdf/papers/cga.pdf, 1995

[62] Yeung, C. J., Susil, R. C., Atalar, E., *RF Heating Due to Conductive Wires During MRI Depends on the Phase Distribution of the Transmit Field*, Johns Hopkins University of Medicine, http://cigi.rad.jhmi.edu/~eatalar/atalar_pubs/fulltext_ID=101521215&PLACEBO=IE.pdf, 2002

[63] Welch, G., Bishop, G., *An Introduction to the Kalman Filter*, http://vision.ee.pusan.ac.kr/reports/papers/IntroKalmanFilter.PDF, November 2004

[64] Schwald, B., Seibert, H., *Registration Tasks for a Hybrid Tracking System for Medical Augmented Reality*, http://wscg.zcu.cz/wscg2004/Papers_2004_Full/J31.pdf, 2002+

[65] Whitaker, R. T., Grampton, C., Breen, D. E., Tuceryan, M., Rose, E., *Object Calibration for Augmented Reality*, Proceeding of Eurographics'95, http://www.eg.org/EG/CGF/Volume14/Issue3/v14i3pp15-27_abstract.pdf, 1995

[66] Ferrant, M., Nabavi, A., Macq, B., Jolesz, F., A., Kikinis, R., Warfield, S., K., *Registration of 3D Intraoperative MR Images of the Brain Using a Finite Element Biomechanical Model*, http://splweb.bwh.harvard.edu:8000/pages/papers/ferrant/tmi2001.pdf, IEEE Transactions on Medical Imaging, 2001

[67] Risholm, P., *Deformable Registration in an Intra-operative Setting*, http://www.ivs.no/downloads/risholm.pdf, Master's Thesis, Norwegian University of Science and Technology, 2004

[68] Isgr, F., *3D points registration*, http://slipguru.disi.unige.it/teaching/didattica/Vis%202/LUCIDI/icp.pdf

[69] Vaidhyanathan, N., *Image Registration*, http://www.cs.purdue.edu/homes/sun/Teach/590Y_05S/Presentation/imageRegistraiton.pdf

[70] Rusinkiewicz, S., Levoy, M., *Efficient Variants of the ICP Algorithm*, Stanford University, 2001+

[71] Eggert, D. W., *Estimating 3-D rigid body transformations: a comparison of four major algorithms*, Machine Vision and Applications, Sprinter-Verlag, http://www.cs.duke.edu/researchers/artificial_intelligence/temp/eggert_rigid_body_transformations.pdf, 1997

[72] Rauterberg, M., *New directions in User-System Interaction: augmented reality, ubiquitous and mobile computing*, IEEE Proceedings Symposium Human Interfacing, http://www.idemployee.id.tue.nl/g.w.m.rauterberg/publications/IEEE99HI-paper.PDF, 1999

[73] Höllerer, T. H., *User Interfaces for Mobile Augmented Reality Systems*, PhD Thesis, Colubia University, http://www.cs.ucsb.edu/~holl/pubs/hollerer-2004-diss.pdf, 2004

[74] Nakanishi, Y., Sato, Y., Koike, H., *EnhancedDesk and EnhancedWall: Augmented Desk and Wall Interfaces with Real-Time Tracking of User's Motion*, http://interact.media.mit.edu/mas963/enhencedwall.pdf, 2002+

[75] Djajadinigrat, J. P., Overbeeke, C. J., Wensveen, S. A. G., *Augmenting Fun and Beauty: A Pamphlet*, Delft University of Technology, http://studiolab.io.tudelft.nl/static/gems/publications/00DjajDARAugm.pdf, 2000

[76] Reitinger, B., Bornik, A., Beichel, R., Werkgartner, G., Sorantin, E., *Tools for Augmented Reality based Liver Resection Planning*, Graz University of Technology, http://liverplanner.icg.tu-graz.ac.at/modules/Publications/2004/reitinger_spie2004.pdf, 2004

[77] *Liver Surgery Planning System - An Overview*, http://liverplanner.icg.tu-graz.ac.at/modules.php?op=modload&name=LSPSI&file=index, May 2005

[78] *Improving the view of the world; law enforcement and augmented reality technology*, http://www.findarticles.com/p/articles/mi_m2194/is_1_73/ai_113525579, May 2005

[79] Trevisan, D. G., Vanderdonckt, J., Macq, B., *Analyzing Interaction in Augmented Reality Systems*, Catholic University of Louvain, http://www.isys.ucl.ac.be/bchi/publications/2002/Trevisan-ITP2002.pdf, 2002

[80] Zwiggelaar, R., Singh, H., Curtin, J., *Discovering hidden information in medical data*, Imaging, Graphics and Vision, http://www.cmp.uea.ac.uk/research/pdfs/Vision/Medical%203.0.pdf, 2002+

[81] Bergli, N., Johnsen, K., *Computer-Aided Surgery - A literature study*, Department of Information Science (IFI), Bergen, http://www.fjeld.ch/teaching/litt_CAS.pdf, 2002+

[82] van Charante, E. M., Cook, R. I., Woods, D. D., Yue, L., Howie, M. B., *Human-Computer Interaction in Context: Physician Interaction with Automated Intravenous Controllers in the Heart Room*, Cognitive Systems Engineering Laboratory, Department of Industrial and Systems Engineering, The Ohio State University, 1993

[83] Lin, W., Robb, R. A., *Dynamic Volume Texture Mapping and Model Deformation for Visually Realistic Surgical Simulation*, Department of Biophysics, Mayo Foundation/Clinic, Rochester, Minnesota, 1999

[84] Bielser, D., Maiwald, V. A., Gross, M. H., *Interactive Cuts through 3-Dimensional Soft Tissue*, Computer Graphics Forum (Eurographics'99), http://graphics.ethz.ch/Downloads/Publications/Tech_Reports/1998/t_Bie98.pdf, 1998

[85] Zhang, H., Payandeh, S., Dill, J., *Simulation of Progressive Cutting on Surface Mesh Model*, Computer Graphics Laboratories, School of Engineering Science Simon Fraser University Burnaby, Canada, http://www.ensc.sfu.ca/research/erl/force/cutting.pdf, 2002

[86] Wohlfahrter, W., Encarnacao, L. M., Schmalstieg, D., *Interactive Volume Exploration on the StudyDesk*, Proceedings of the 4th International Projection Technology Workshop, Ames, Iowa, USA, http://www.inigraphics.net/press/topics/2000/issue4/4_00a04.pdf, 2000

[87] Foley, van Dam, Feiner, Hughes, *Computer Graphics, Principles and Practice, Second Edition in C*, Addison-Wesley Professional, 1997, pp471 - 530