STEREO AUGMENTED REALITY IN A NAVIGATED SURGICAL MICROSCOPE

Yusuf Özbek B.Eng.

Innsbruck Medical University

Master Thesis for Fulfillment of Requirements for the
Master of Science (M.Sc.)

Supervisor: ao. Univ.-Prof. Dr. Mag. Wolfgang Freysinger

Innsbruck Medical University University of Innsbruck

Univ. ENT Department, 4D Visualization Laboratory
Innsbruck Medical University,
Austria, Innsbruck,
June 2016

Statement of Originality

The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. Herewith I declare that I have not submitted this material, either in whole or in part, for a degree at this or any other university or institution. This is to certify that the printed version is equivalent to the submitted electronic one.

Yusuf Özbek B.Eng.
Author

ao. Univ.-Prof. Dr. Mag. Wolfgang Freysinger
Principal Advisor

ABSTRACT

In a minimal invasive microscopic image-guided surgery (IGS) a precise navigation and stereoscopic augmented reality (AR) is a necessary requirement. The goal of this thesis was explore the field of AR overlay with a purpose-built surgical microscope (Leica M500-N) that tracked intraoperatively with an optical tracker (NDI Optotrak Certus) and developing a medical application concept (AROSCOPE – Stereo Augmented Reality in a Navigated Surgical Microscope), which uses the platform independent open-source libraries. For the microscope embedded surgery, the patient is registered with landmark-based registration technique and the head of microscope is navigated at the same time with the patient together. Thereby segmented anatomic structures from preoperative radiological images and access pathways can be represented accurately as AR overlays in field of view (FOV) of surgeon, without the attention of surgeons is deflected from the surgical field.

The technical concept and core area of AROSCOPE involves two dimensional (2D) visualization and spatial viewing of the patient images, calibration of the stereo cameras that mounted on the head of microscope, patient-image registration, three dimensional (3D) segmentation, intraoperative surgical navigation and overlaying of segmented anatomical structures into the optics of the microscope in real-time.

This thesis presents a solution to these problems and provides the possibility to enhance the surgeon’s ability for a better intraoperative orientation by overlaying 3D information directly in FOV.

This project is funded by the Jubilee Fund of the Austrian National Bank, project 13003.

TABLE OF CONTENTS

CHAPTER 1 - Introduction 1
1.1 Motivation 2
1.2 Background 3
1.3 Scope 3
1.4 Organization 4

CHAPTER 2 - Surgical Navigation 5
2.1 Introduction 6
2.2 Tracking 7
  2.2.1 Tracking systems 7
    2.2.1.1 Mechanical 8
    2.2.1.2 Electromagnetic 9
    2.2.1.3 Optical 12
  2.2.2 NDI Optotrak Certus 14
    2.2.2.1 System components 16
    2.2.2.2 Rigid body design 17
    2.2.2.3 Transformation formats 17
    2.2.2.4 Rigid body configurations 18
  2.2.3 Instrument calibration 19
  2.2.4 Registration 23
  2.2.5 Patient tracking 25
  2.2.6 Fiducials 30
  2.2.7 Microscope tracking 31
  2.2.8 Hand-eye calibration / Calibration pattern tracking 33
2.3 Visualization 37
2.4 Results 38

CHAPTER 3 - Camera Calibration 40
3.1 Introduction 41
3.2 Pinhole camera model 42
  3.2.1 Camera parameters 45
    3.2.1.1 Intrinsic parameters 45
    3.2.1.2 Extrinsic parameters 46

3.3 Distortion 48
3.4 Stereo camera calibration.. 50
3.5 Direct linear transformation.. 54
3.6 Visualization 57
3.7 Results... 59

CHAPTER 4 - Augmented Reality... 60
4.1 Introduction... 61
4.2 Stereo augmented reality 63
4.3 Surgical microscope Leica M500 N 66
4.4 3D volume generation... 70
4.5 Overlaying and rendering 70
4.6 Validation.. 74
4.7 Visualization 76
4.8 Results... 78

CHAPTER 5 - AROSCOPE 79
5.1 Introduction... 80
5.2 System architecture... 81
5.3 Hardware setup 82
5.4 Software architecture... 83
5.5 Installation and configuration.. 84
5.6 Logging... 86
5.7 5 Modules 87
  5.7.1 DICOM module 88
  5.7.2 Navigation module... 93
  5.7.3 3D generation module.. 96
  5.7.4 Calibration module... 98
  5.7.5 Augmented reality module... 100
5.8 Sub-windows and functionalities.. 102
  5.8.1 Open file... 102
  5.8.2 Settings.. 104
5.9 Further development.. 106

CHAPTER 6 - Summary... 107
6.1 Conclusions... 108

6.2 References 109

LIST OF FIGURES

Figure 1: The functional principle of mechanical localizer 8

Figure 2: Patient-image registration with a MTS 9

Figure 3: The functional principle of electromagnetic tracking system 10

Figure 4: Experimental setup of Rhinospider technology with an EMTS 11

Figure 5: Videometric tracking systems. a) Outside-looking-in b) Inside-looking-out 12

Figure 6: NDI Optotrak Certus system setup [11] 14

Figure 7: Optotrak operational working volume [11] 15

Figure 8: Self constructed rigid body design 17

Figure 9: 4-marker tool based configuration 18

Figure 10: Self constructed tool based configuration 19

Figure 11: Static pivot calibration process 20

Figure 12: Dynamic pivot calibration process 21

Figure 13: Point-based registration workflow 24

Figure 14: Model-based registration tool 25

Figure 15: Patient navigation and coordinate system 29

Figure 16: Interaction of computer-assisted surgical navigation components 29

Figure 17: Arrangements of fiducials 30

Figure 18: Microscope frame adaptation from some companies 31

Figure 19: Self constructed rigid bodies adapted on the microscope head 32

Figure 20: Microscope navigation and coordinate system 33

Figure 21: Hand-eye calibration 35

Figure 22: Hand-eye calibration and coordinate system 36

Figure 23: AR transformation chain and coordinate system 37

Figure 24: Visualization of patient navigation 38

Figure 25: 3D, 2D and 1D calibration patterns 42

Figure 26: Pinhole camera model 43

Figure 27: From camera coordinates to image plane coordinates 45

Figure 28: Extrinsic camera parameters 47

Figure 29: Kind of lens distortions 48

Figure 30: Radial distortion 49

Figure 31: Representing of pincushion and barrel distortion on image plane 49

Figure 32: Tangential distortion 50

Figure 33: Epipolar geometry 51
Figure 34: Estimation of DLT for two calibration points 55
Figure 35: Visualization of camera calibration process in calibration module 58
Figure 36: Visualization of validation in calibration module 58
Figure 37: Milgram’s reality-virtuality continuum...61
Figure 38: Components of Leica M500 N 67
Figure 39: Baumer-GAPI complete system and object classes [35].................................69
Figure 40: Generated 3D model with edge visibility...70
Figure 41: Data flow diagram of overlaying workflow 71
Figure 42: Perspective frustum and normalized device coordinates 72
Figure 43: Validation of AR transformation chain using first checkerboard pattern 75
Figure 44: Validation of AR transformation chain using second checkerboard pattern....76
Figure 45: Visualization of an overlaying in a phantom patient.......................................77
Figure 46: AROSCOPE 80
Figure 47: AROSCOPE's hardware setup 82
Figure 48: AROSCOPE's software architecture 84
Figure 49: CMake configuration...86
Figure 50: Overview of DICOM module..88
Figure 51: Overview of navigation module 93
Figure 52: Overview of 3D generation module 96
Figure 53: Overview calibration module 98
Figure 54: Overview augmented reality module 100
Figure 55: Open file window first modus..102
Figure 56: Open file window second modus 103
Figure 57: Main application settings..104
Figure 58: Tracker settings 105

LIST OF TABLES

Table 1: NDI Optotrak Certus characteristics....................15
Table 2: NDI Optotrak Certus system components....................16
Table 3: The content of a RIG file....................20
Table 4: Adding an used rigid body into the tracking list....................26
Table 5: Optotrak tool pose information....................26
Table 6: Converting quaternions to a transformation matrix....................28
Table 7: Probe tool coordinates relative to DRF....................28
Table 8: Transformation chain from pattern to microscope tool....................64
Table 9: Determine 3D object pose to 2D point correspondences....................65
Table 10: Transformation chain from patient image to camera....................65
Table 11: Overview of Leica M500 N features....................66
Table 12: Overview of Baumer TXG06c features....................68
Table 13: Overlaying and rendering with VTK, option 1....................73
Table 14: Overlaying and rendering with VTK, option 2....................74
Table 15: X.Org Server configuration....................82
Table 16: Singleton pattern mechanism....................83
Table 17: CMake file example....................85
Table 18: Functionalities in main window and DICOM module....................92
Table 19: Functionalities in navigation module....................95
Table 20: Functionalities in 3D generation module....................97
Table 21: Functionalities in calibration module....................99
Table 22: Functionalities in augmented reality module....................101
Table 23: Functionalities in open file window first modus....................102
Table 24: Functionalities in open file window second modus....................104
Table 25: Functionalities on settings window....................105

ABBREVIATIONS AND ACRONYMS

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
4D	Four Dimensional
API	Application Programming Interface, Application Programming Interface
AR	Augmented Reality
AROSCOPE	Stereo Augmented Reality in a Navigated Surgical Microscope
ARS	Augmented Reality Systems
CAN-Bus	Controller Area Network
CAS	Computer-Assisted Surgery
CCD	Charge-Coupled Device
CMake	Open-Source Build System
CT	Computed Tomography
CTK	The Common Toolkit
DICOM	The Digital Imaging and Communications in Medicine
DLT	Direct Linear Transformation
DOF	Dimension of Freedom
DRF	Dynamic Reference Frame
DVI	Digital Visual Interface
EMTS	Electromagnetic Tracking System
ENT	Ear, Nose and Throat Medicine
FOV	Field of View / Working Volume
FPS	Frames Per Second
FRE	Fiducial Registration Error
GAPI	Generic Application Programming Interface
GigE	Gigabit Ethernet
GPS	Global Positioning System
GUI	Graphical User Interface
HDM	Head-Mounted Display
ICP	Iterative Closest Point
IDE	Integrated Development Environment
IGS	Image-Guided Surgery

IGSTK	Image-Guided Surgery Toolkit
IR	Infrared
IRED	Infrared Light-Emitting Diodes
ITK	Insight Segmentation and Registration Toolkit
LAN	Local Area Network
LED	Light Emitting Diode
LSB	Least Significant Bit
MPR	Multiplanar View
MR	Mixed Reality
MRI	Magnetic Resonance Imaging
MTS	Mechanical Tracking Systems
NDC	Normalized Device Coordinates
OpenCV	Open-Soruce Computer Vision
OpenGL	Open Graphics Library
OS	Operating System
OTS	Optical Tracking System
POC	Proof of Concept
Qt	GUI Development Framework
RIG	Rigid Body File
RMS	Root Mean Square Error
ROI	Region of Interest
RS	Rhinospider
SCU	System Control Unit
SROM	Serial Read Only Memory
TCP/IP	Transmission Control Protocol / Internet Protocol
US	Ultrasound
USB	Universal Serial Bus
VGA	Video Graphics Array
VTK	The Visualization Toolkit
VTS	Videometric Tracking System
X.Org Server	X Window System

ACKNOWLEDGMENT

Hereby I would thank specially to my advisor Prof. Wolfgang Freysinger for his guidance, knowledge, patience and providing me with an excellent atmosphere to doing this research.

I would like to thank my friend Zoltán Robin Bárdosi in our research group for his help and for the best knowledge support.

Finally, I would like to thank all member of my family individually for their constant support during my thesis and open my way with their being.

CHAPTER 1 - Introduction

1.1 Motivation

Augmented Reality in medicine provides the surgeon the ability to plan surgical interventions and to be assisted in their implementation with additional virtual information, which is generated by computers and combines real and virtual world by source one radiological image in DICOM format e.g. computed tomography (CT) with another e.g. real-time video of patient. In addition, simulations of anatomical processes for presentation and training purposes to be produced using this technology. For most medical applications, it comes to overlay virtual preoperatively obtained data from CT, magnetic resonance imaging (MRI), or ultrasound (US) images to the treated body part of the patient. Thus, complex anatomical structures, even functional relationships are represented.

The positionally accurate superimposition of spatial information in oculars of surgical microscope is a decisive criterion for the use of such a system. In an augmented reality system, real images are supplemented by additional information. Augmented Reality is therefore the link between real and virtual worlds. In clinical practice the surgery providing preoperative data is an indispensable aid for the surgeon. This data is obtained from preoperative images CT, MRI, or US, etc. Segmentation results and 2D or 3D representations, which are obtained from these imaging techniques serves to the surgeon for necessary orientation during the surgery. These spatial data can be made available to the physician as a virtual data in an augmented reality environment.

In minimally invasive procedures in ear, nose and throat medicine (ENT), which are carried out with surgical navigation, the surgeon typically has to direct attention from the surgical field on the monitor of the navigation system. In microscope embedded interventions, the view from the microscope eyepiece must be turned to the preoperative radiological images of the navigation system and vice versa. This is not appropriate for a sensible and ergonomic integration of navigation technology in the surgical workflow, cause due to the constantly changes of the visual field during surgery the intervention is rather more difficult and the hand-eye coordination, adaptation and orientation of the surgeon is limited with the time. All this distracts the surgeon from the real target, to operate exactly and has a negative effect on the acceptance of this technology. This is confirmed by the fact that there are very few reports of microscopic navigated procedures in ENT, but also in neurosurgery.

For navigated operating microscopes are Augmented Reality Systems (ARS) already commercially available solutions, that can represent additional virtual content in the surgical field. This thesis describes the key points and objectives of AROSCOPE.

1.2 Background

AR and position measurement systems are essential and promising components of image-guided surgery [1]. Surgical navigation can thus allow for more precise and more targeted interventions. Especially in critical operations the surgical navigation can be used to overlay physiologically correct 3D information in the visual field of the surgeon.

By means of position accurately augmented reality, enhanced visual perception, predictable reliability, precision of the entire system and all the possible parameters of systems such workflow, handling, visualization strategies, perception physiology etc. can be studied in detail. Thereby the best interface between surgeons and microscopic navigation system can be elicited under circumstances. Ultimately AROSCOPE is used to research the best to be chosen visualization to increase the acceptance of augmented navigation under the microscope at the lateral skull in surgical use.

So AROSCOPE is a comprehensive experimental system in aspects of ergonomics and perception physiology, but also the psychology and navigation with completely known application accuracy can be studied quantitatively.

1.3 Scope

The aim of this thesis was developing a navigation and stereoscopic augmented reality concept (AROSCOPE) for the use in minimal invasive microscopic image-guided surgery in ENT operations.

The concept serves user-friendly interactions and all essential steps from planning until overlaying phase and fulfills important requirements for a successfully surgery in minimal invasive interventions with navigated microscope. To enhance the ability of the surgeon during the surgery and facilitate the whole complex workflow, AROSCOPE offers step by step processing of major steps and clear guideline for the user. Consequently this system enhances the capabilities of the surgeon visual system through the combination of computer generated graphics, computer vision and advanced user interaction technology.

1.4 Organization

There are four parts in this thesis. The first part introduces the background of the tracking technologies such as mechanical, electromagnetic and optical systems based on various methods. Especially the optical tracking system (NDI optotrak certus), and rigid body design described detailed, which is used for patient and microscope navigation in AROSCOPE. The registration topic in this part describes leading algorithms, which is currently used in IGS interventions. The second part focuses on mathematical concept of camera calibration, which represents the important part of augmented reality. The themes: camera models such as pinhole camera model, camera parameters such as intrinsic and extrinsic, stereo camera calibration are processed.

The third part goes into the stereo augmented reality with tracked surgical microscope. The setup and systematical description of surgical microscope for augmented reality, image injection and visualization of overlayed anatomical structures in microscope oculars build main components of this part. Finally the last part describes the system architecture, mainly used components of software system, hardware setup and five different modules which create the whole concept of AROSCOPE.

CHAPTER 2 - Surgical Navigation

2.1 Introduction

In computer assisted- ENT surgery, orthopedic surgery, neurosurgery, or visceral surgery, etc. procedures, the surgical navigation systems [2] are very important and necessary part of the whole complex process. Preoperatively obtained radiological images from the patient are used for the 3D orientation intraoperatively. A navigation system technology allows to the surgeon to determine the access path of the surgical area exactly and guide the surgeon during the surgical intervention to operate after reaching the pathological tissue without any injury. Thus it allows keeping healthy tissue in the minimum of damage with available shortest and safest boundaries of planned access path of the surgical area.

Using the surgical navigation system the surgeon uses special instruments called tool, which are tracked by the position measurement system called tracker. The position of a tracked tool in relation to the patient's anatomy is shown on radiological images of the patient, as the surgeon moves the tool in operation field during the surgery. The surgeon thus uses the navigation system to navigate the position of a tool in 3D Cartesian coordinate system. The information of the tool position that system offers, is particularly useful in situations where the surgeon cannot actually see the tip of the tool such as in above referred surgeries.

Surgical navigation systems work very similarly with the automotive navigation system in background that uses Global Positioning System (GPS). The only difference lies in accuracy. While automotive navigation system uses a car as moving instrument in a map, that all coordinates are known from the GPS, the workstation maps the current coordinate of the car that obtained from the GPS with the map and locates the car during driving. Thereby it provides the application accuracy in the meter range. The surgical navigation systems uses a tool (e.g. lancet, scalpel, grasper, needle, etc.) as moving instrument in radiological images of patient that all coordinates are known from the tracker, the workstation maps the current coordinate of the tool with the images and locate the tool during the surgery. Thereby it offers the application accuracy in millimeter range.

2.2 Tracking

The tracking is a process to localize and determine the 2D or 3D movements of objects or people in space. The tracking systems called as localizer too, ensure 6 dimension of freedom (6-DOF) to track a rigid body or rather tool (3-DOF for positions *x*, *y*, *z* coordinates and 3-DOF for orientation *yaw*, *pitch* and *roll*) called as pose. It is used in military, entertainment, sports, video games, medical applications, and for validation of computer vision and robotics.

The tracking should be regarded as an interface between user and workstation and therefore constitute a decisive factor for the realism of the surgical navigation. To visualize the computer-generated scene and calculate potential interactions between instrument and patient images correctly the workstation should be able to know where the instrument is located at any time that moved from the user in the FOV of the tracker.

There exist several tracking systems based on different measurement techniques to fulfill medical navigation requirements. In the field of the surgical navigation the tracking system occupies mainly with the positioning of the patient, devices such as surgical microscope, robotic systems, or endoscope, etc. tools as medical instruments or their replicas. In the case of this thesis the tracking system is used to navigate the movements of patient and microscope head intraoperatively.

In this chapter the state of the art and gold standard tracking technologies are described in detail. Especially, the optical 3D tracking system NDI optotrak certus is explained, which is used in this project.

2.2.1 Tracking systems

Mainly the tracking systems differ in three main categories:

1. Mechanical (early systems)
2. Electromagnetic (standard for flexible instruments)
3. Optical (gold standard of navigation)

2.2.1.1 Mechanical

Mechanical tracking systems (MTS) [3] consist of articulated passive arms called booms with mounted rotary joints and used as localizer in the past until it is replaced with optical tracking systems. It could allow high accuracy, resolution, reliability and stable positioning while position and orientation information of the arm endeffector is resolved using forward kinematics. The use of mechanical digitizers facilitated frameless stereotaxy by localizing either an operating microscope's focus, or a surgical tool inside the patient's cranium in low lag. For all that MTS were very cumbersome to use and can interfere with surgical area, besides it doesn't allow to track multiple devices and has very small measurement volume (FOV) depending on the size of the arm.

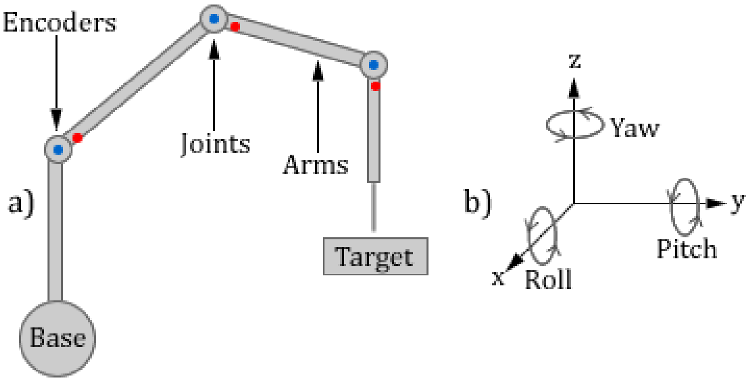


Figure 1: The functional principle of mechanical localizer

The working principle of a MTS (Figure 1a) is based on a physical connection between rigidly fixed target (patient) and fixed reference point (base) in 3D coordinate system. In each distal joint it is placed an encoder to compute the angle and measure the movements of target in position and orientation (6-DOF, Figure1b) with respect to the reference point. During the measurements it must be considered that encoders (blue points) and the arm (red points) should not point the same plane by rotating of the arms. Otherwise the selected target points *x*, *y* and *z* lies in the same plane and that affects the accuracy of tracking.

In Figure 2 it is visible an image from a workshop demonstration, while we working with a MTS in year 2009. It was our first meet with a navigation system in our research group and the goal of the workshop was to master a MTS and learn the manual landmark-based registration process with image- and anatomical fiducials in IGS, while the person, who works front of the workstation selects the image fiducials at the patient images, the other person try to points same fiducial with the arm of localizer at the fixed phantom patient.

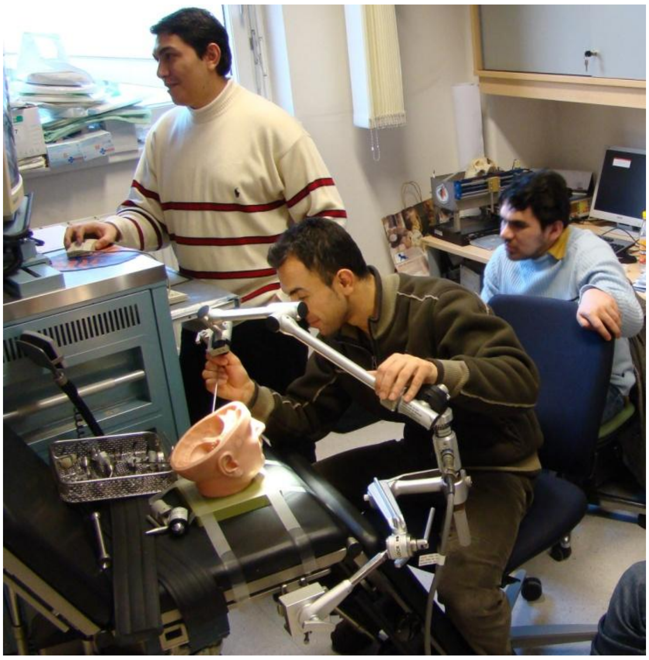


Figure 2: Patient-image registration with a MTS

2.2.1.2 Electromagnetic

Electromagnetic tracking systems (EMTS) [4] generally consist of three components:

1. Field generator (Tracker)
2. Signal generator (SCU)
3. Central control unit (workstation)

The field generator consists of several coils to generate a position varying magnetic field that is used to establish the Cartesian coordinate space from the tracked object. The sensors, which are connected to the signal generator and attached to the tracked patient or rather in the near of surgical area, contains small wire coils called sensors, in which current is induced via the magnetic field that generated by field generator (Figure 3).

By tracking the behavior of each connected sensor, the position and orientation of the tracked object can be determined and localized by setting one of the sensors as a dynamic reference frame (DRF) and measuring the positions all other sensors relative to the DRF in real-time. With this technique, the positions of each sensor are detected when moving within the coordinate space. The workstation serves to control the field generator and capture data from the signal generator. EMTS can serve three positions in *x, y, z* and two or three orientation angles, and therefore are referred as 5-DOF or 6-DOF.

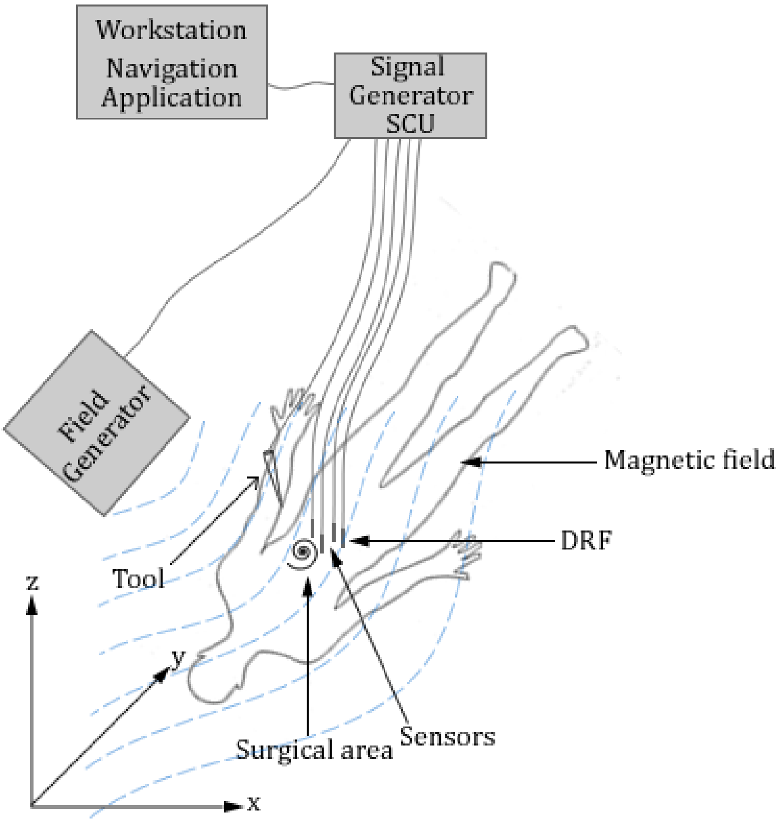


Figure 3: The functional principle of electromagnetic tracking system

The important advantage of EMTS is that electromagnetic fields do not depend on line-of-sight for surgery, can operate in useable areas e.g. inside the patient body and track flexible tools such as endoscope. Other advantages of the EMTS are: They have very small and light weight components and can produce high resolutions under controlled conditions.

Since EMTS depends on the measurement of magnetic fields produced by the filed generator, the tracking units may be disturbed by the presence of some electronic device that produces electromagnetic interference or include susceptibility to distortion from nearby metal sources and thus limits accuracy by tracking. Other disadvantages of the EMTS are cabled and very sensitive sensors, expensive instrumentation and limited FOV.

A medical application example for the EMTS is Rhinospider (RS) technology [5] developed in our research group that uses an EMTS (NDI Aurora) and offers submillimetric application accuracy in ENT surgeries. Three or four titanium spheres, which contain wired sensors inside, are inserted percutaneously in the near of the surgical area followed by a CT scan and automatically detected titanium sphere positions in patient images are registered with the sensor positions, which lie in the same location and then tracked intraoperatively. Figure 4a) illustrates the main setup of this technology. The wired RS sensors are inserted in the surgical area, which are connected to the marker strober with the tool together and marker strober to the SCU device. The field generator determines and localizes the positions of the sensors in the FOV of the tracker and allows surgical navigation. b) shows titanium spheres with RS sensors, which are inserted in it and attached in surgical field. The figures c, d and e) demonstrate the relationship between rigid RS structure and wired sensors.

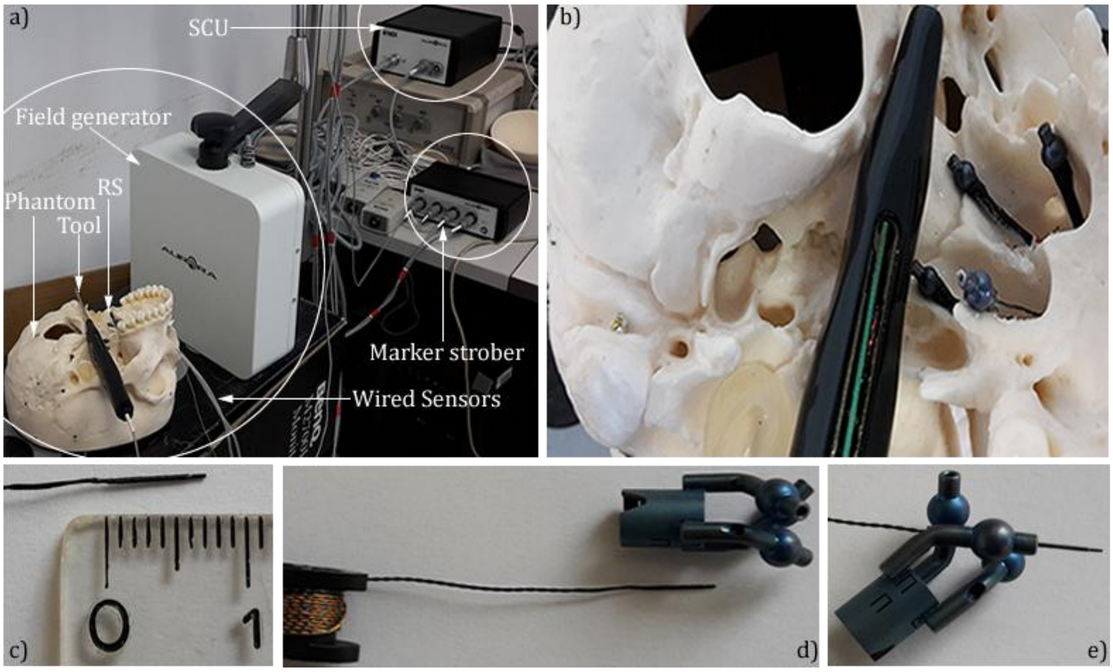


Figure 4: Experimental setup of Rhinospider technology with an EMTS

2.2.1.3 Optical

In general, optical tracking systems (OTS) [6] can differ in two main categories: Videometric tracking systems [7] [8] and infrared-based (IR) tracking systems [9]. While videometric tracking system detect and identifies the marker patterns called as landmarks on video image sequences and uses usually one or more calibrated video cameras, the IR-based tracking systems use a band-pass filter to eliminate all ambient light of other wavelengths in environment and makes the identification of optical markers a comparatively simple and reliable task. The IR-based systems are usually used in medicine for IGS interventions.

The videometric tracking systems (VTS) come basically in two sub-variants: Outside-looking-in and inside-looking-out approaches and often used for head tracking with head-mounted display (HMD) that equipped with passive markers on it. By outside-looking-in systems the cameras are fixed in the environment and markers are mounted on the moving object that should be tracked by those cameras, while by inside-looking-out the cameras are fixed on the tracked object and markers in the environment. Based on the locations of the detected markers, the position and orientation of the tracked objects are calculated.

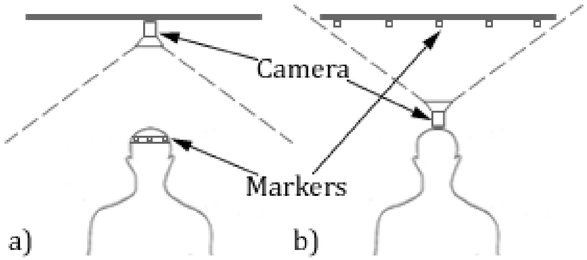


Figure 5: Videometric tracking systems. a) Outside-looking-in b) Inside-looking-out

There are some disadvantages to the inside-looking-out system. For small or medium-sized working volumes, mounting the sensors on the object is more challenging than mounting them in the environment. Because markers can be relatively simple, small, and cheap, they can often be located in numerous places on the tracked object, and communication from the object to the rest of the system can be relatively simple or even unnecessary. This is particularly attractive for full-body motion In particular, a significant problem with an outside-looking-in configuration is that only position estimates can be made directly, and so orientation must be inferred from position estimates of multiple fixed landmarks. In other hand these wireless systems serves large FOV with low lag and high availability and they have no magnetic interference problems.

IR-based tracking systems can be categorized under two other approaches: Active and passive optical trackers.

Active or passive OTS [10] consists of three main components:

1. Position sensor (tracker)
2. System control unit (SCU)
3. Central control unit (workstation)

and use active wired infrared light-emitting diodes (IRED) as marker that are activated by an electrical current and operates in the near of IR spectrum that fired from the tracker or wireless passive spherical retroreflective markers, that reflect infrared light emitted by the illuminators on the tracker. The trackers consist of either two planar (usually used for passive tracking systems) or three linear (usually used for active tracking systems) charge-coupled device stereoscopic cameras (CCD). The IREDs that should be tracked on a target are connected to the SCU and fired sequentially to be detected by each CCD cameras on the tracker. To determine the pose of the tracked objects the SCU uses a process of triangulation based on the known geometric configuration and the known fixed distance between the CCD components. For achieving to 6-DOF pose information from the tracked object, minimally three non-collinear IREDs should be available. Since the IREDs must be powered, traditionally active systems were also wired systems.

Passive OTS work in the near IR range like active systems and they use spherical retroreflective markers in different sizes instead of active markers that gives the ability of the sphere to reflect a beam of light back in the opposite direction, over a large range of incoming directions. The spheres are detected and localized by the tracker cameras. These systems are equipped always with 2D CCD cameras, because the pattern of the reflective markers, which has to be unique for each tracking tool so that unambiguous assignment of each probe is practicable, is identified on a 2D image.

One important advantage of this system is that no wires are needed between the tracking system and the tracked tools.

Generally provide the OTS very high accurate, high resolution and update rate. The FOV can be very large but the main disadvantages of these systems are they suffer line-of-side problem, so there should be always a direct intervisibility between the tracker and tracked tools. One of the other important disadvantage is that these systems cannot operate in the invisible areas e.g. inside the body and cannot track the flexible tools such as needle used in percutaneous needle-guided interventions or endoscope.

2.2.2 NDI Optotrak Certus

The optotrak certus system from NDI Europe GmbH is an optical measurement device (Figure 6) used to track the kinetic or dynamic position and orientation in 6-DOF of IREDs in real-time within a specific area (Figure 7) [11]. With an accuracy of up to 0.1 mm and resolution of 0.01 mm, the optotrak certus tracking system continues to deliver the degree of precision that has become an optotrak hallmark. A maximum marker frequency of 4600 Hz. enables to capture data at even higher speeds. In addition, the optotrak can track up to 512 markers and is pre-calibrated for fast set-up and ease of use.

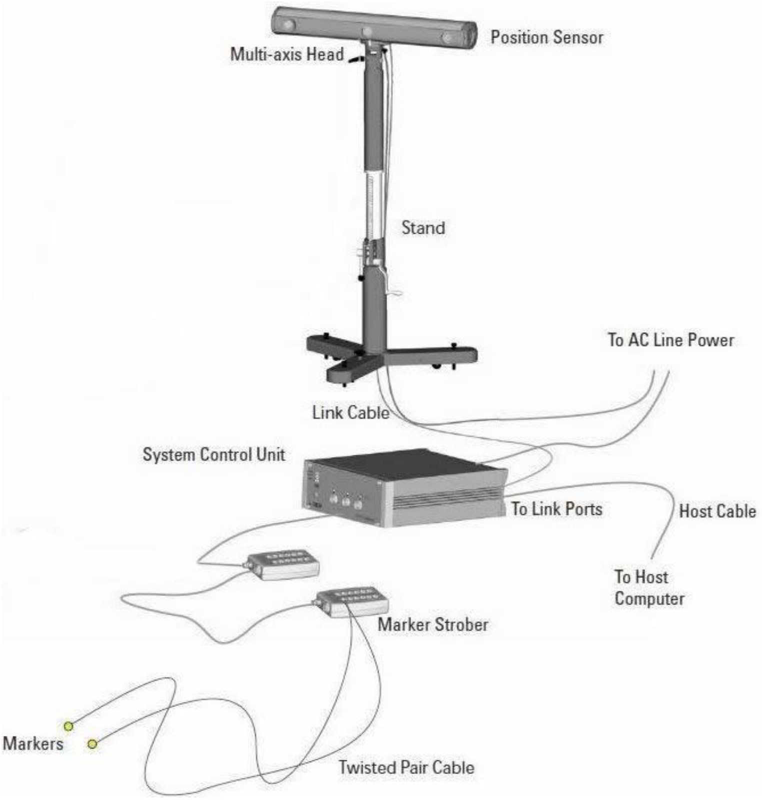


Figure 6: NDI Optotrak Certus system setup [11]

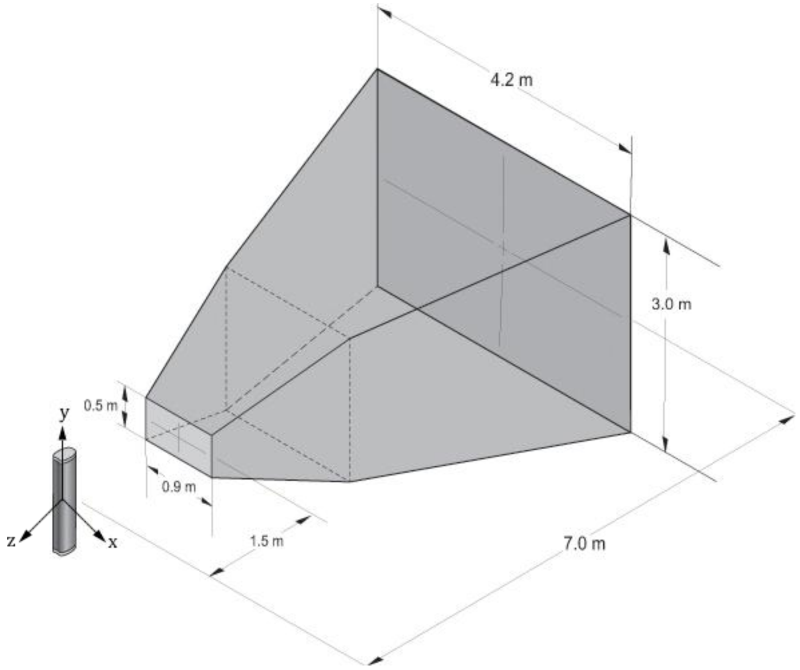


Figure 7: Optotrak operational working volume [11]

During the tracking the pose information of markers are reported relative to global coordinate system and the global coordinate system for each marker is placed default in origin (0, 0, 0) at the middle sensor of the tracker. Table 1 below lists the main characteristics of the optotrak in detail.

Feature	**Value**
Max. number of markers to track	512
Max. sampling rate (marker frequency)	From 1000 to 4600 Hz
Max. frame rate	4600 / N+2 Hz , N= number of markers
Number of strober ports	3
Strober types	Marker, wireless and tool
Max. number of rigid bodies to track	170 (3 markers rigid body)
Max. number of strobers	10
Host computer connection options	Ethernet 10/100 Mbps, PCI and USB
Marker voltage	From 6.0 to 12.0
Duty cycle (%)	From 10 to 85

Table 1: NDI Optotrak Certus characteristics

2.2.2.1 System components

Table 2 describes the basic hardware components of optotrak system. [11]

Component	**Description**
Position sensor	An optical device that detects infrared light emissions within a specific range.
System control unit (SCU)	A processing device that controls the operation of the position sensor and attached strobers. It processes the information collected by the position sensor and sends it to the host computer via USB.
Markers	Infrared light emitting diodes IREDs that are tracked by the position sensor when they activated by the strober. A structure with three or more markers whose relative positions are fixed is called a rigid body
Strobers	A device controlled by and connected to the SCU, responsible to activating and deactivating markers. There are four strober types: 1. Tool strober 2. Marker strober 3. 3020 strober adapter 4. Wireless strober
Cables	The system is supplied with following cables: 1. The host cable connect host computer to the SCU 2. Link cables connect the position sensor to the SCU, an USB interface to the SCU, or a position sensor to another position sensor. 3. Strober extension cables extend the distance between the SCU and a strober. 4. Power cables.
Host computer	A user-supplied computer used to communicate with the optotrak system. The host computer sends system operation introductions to the SCU trough software applications or drivers. The host computer then receives and interprets the data that collected by the SCU.

Table 2: NDI Optotrak Certus system components

2.2.2.2 Rigid body design

Optotrak offers the users to build their own rigid body structure and use it for the purpose. In Figure 8 four IRED diodes are attached on a plastic tray and connected as twisted pairs with the port (1, 2) of marker strober, which contains 24 ports and plugged directly into the port of SCU device.

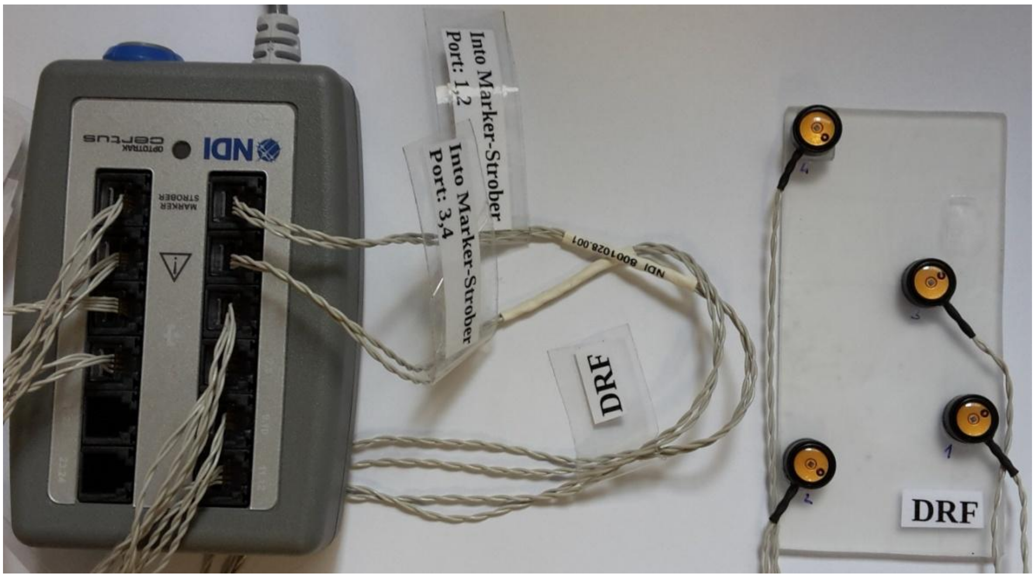


Figure 8: Self constructed rigid body design

For the patient and microscope navigation in AROSCOPE, it is constructed five different rigid bodies regarding their placement in 3D Cartesian coordinate system on the plastic tray. One of them is used as DRF, three are attached to the microscope head and last one is used to track the checkerboard pattern. The markers called encased markers consist of IRED diodes and sized 11 mm in diameter. Each self constructed rigid body consists of four markers. The structural property of designed rigid body allows the optotrak the rotational movements of the rigid body and as well as each marker's translational movements.

2.2.2.3 Transformation formats

The combination of translation and rotation information is called a transformation. The translation of a rigid body's origin is reported using T_x ,T_y and T_z coordinates. The rotation of a rigid body can be reported in mathematical quaternion, Euler, or rotational matrix format. By default, the optotrak system reports rotations in quaternion format. A quaternion rotation uses four dimensional (4D) objects that are represented as ordered quadruples q_0 , q_x , q_y , q_z. The components represent a rotation about an axis (defined by the vector q_x , q_y , q_z) by an angle (determined by the scalar q_0). For normalized or unit quaternions, the rotation of a vector v by the quaternion q is given by $v' = qvq^{-1}$.

2.2.2.4 Rigid body configurations

To track the microscope head, patient and other components at the same time, the optotrak should be configured properly before beginning of any measurement process with AROSCOPE. For this purpose it can be used a combined hardware configuration with included 4- marker tool- and self constructed rigid bodies. Figure 9 describes the 4-marker tool based configuration of AROSCOPE's navigation module. To navigate the patient, it needed a probe and DRF tool, which plugged into ports 1 and 3 on tool strober and to navigate microscope, it is needed three 4-marker tools, which are plugged into ports 2 and 4 on one tool strober and into port 1 on other tool strober. To track the checkerboard it is needed a self constructed 4-marker rigid body, which is connected to ports 1, 2 and 1, 3 as twisted pairs on marker strober. In this configuration design, all ports of 4-marker tools can be switched on tool strobers as well on SCU except the ports on marker strober.

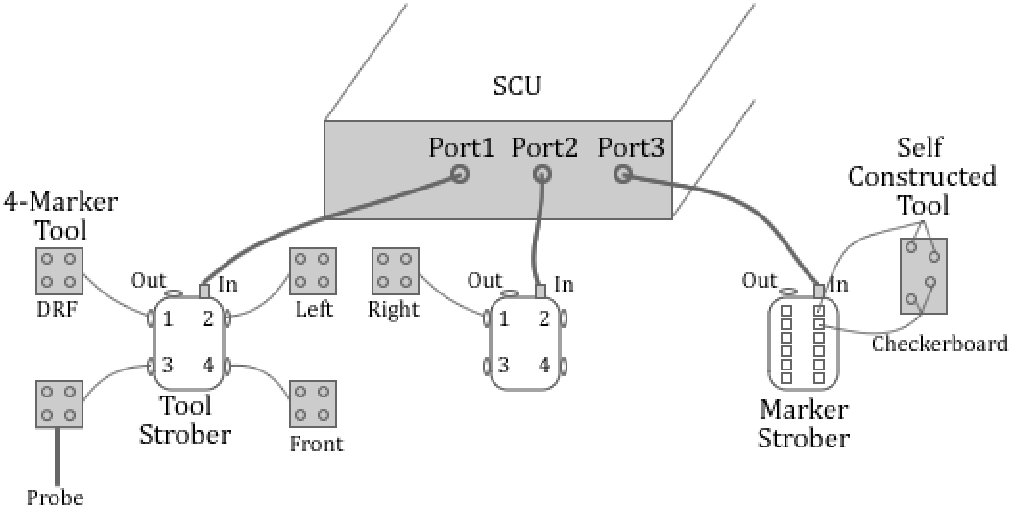


Figure 9: 4-marker tool based configuration

Self constructed tool based configuration (Figure 10) accepts only self build rigid bodies expect probe tool. In this configuration the probe tool is plugged into port 1 on tool strober and all others rigid bodies into ports (DRF- 1, 2 and 3, 4), (Left- 5, 6 and

7, 8), (Front- 9, 10 and 11, 12), (Checkerboard- 13, 14 and 15, 16) and (Right- 17, 18 and 19, 20) as pairs on marker strober respectively. The ports of plugged tools on marker strober can be switched only as twisted pairs.

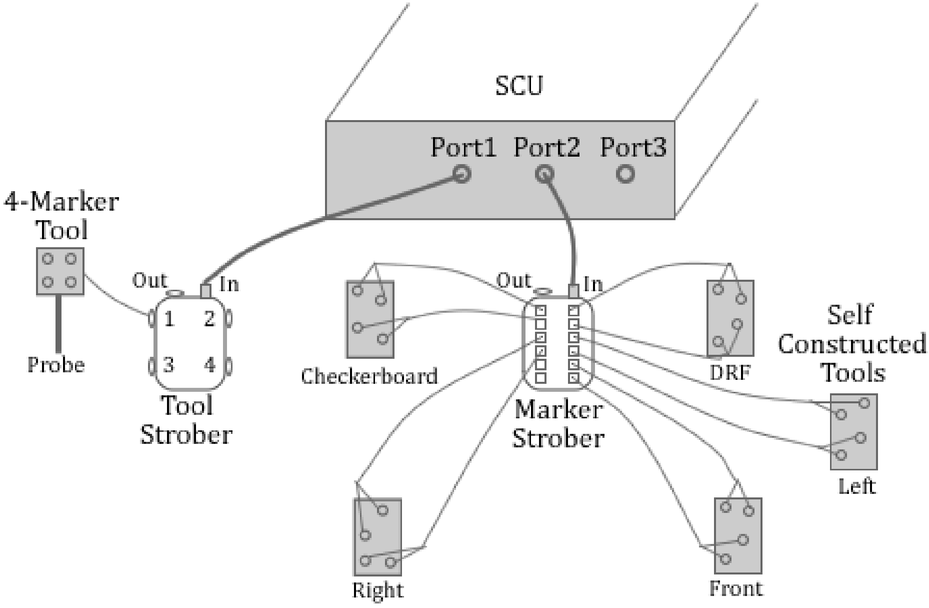


Figure 10: Self constructed tool based configuration

The main difference between a tool and a rigid body is that a tool incorporates a serial read only memory (SROM) device, which is wired into the tool connector. The SROM device stores information such as the number of markers and relative locations of the markers of the tool. The system automatically reads this information from the tool once it is connected to the system. The tools must be connected to a tool strober. Rigid bodies can be built from markers, which are connected to a marker strober.

2.2.3 Instrument calibration

The calibration or characterization of tracked tools is very important part for the tracking and for the accuracy during the surgical navigation. The calibration process describes the future of the calibrated tool or rigid body and provides marker description file called rigid body file (RIG) with information such as positions of the markers on the tool relative to the tracker, marker dimensions, marker angles, RMS error of the performed calibration, etc. Table 3 shows the content of a RIG file.

In AROSCOPE each rigid body is calibrated with included NDI 6D Architect [12] software individually and created a RIG file. From the outcomes shown in Table 3, it can be estimated, that the calibrated tool has a square form, where the markers are exactly located on it, the individual deviations of the sensors with overall calculated calibration error, maximum available rotation angle in which the tool will be visible by the tracker, etc. The listed 3D coordinates give the possibility to calculate the centroid of the tool (described below) and let handle it as a rigid body during the navigation.

```
Marker Description File
4       ;Number of markers
Marker    X          Y             Z     Views
1      23.4046    -26.4016       2.0465    1
2      25.5751     23.2754       7.3074    1
3     -23.4745     26.3664      -2.0273    1
4     -25.5051    -23.2402      -7.3265    1
MaxSensorError 0.20, Max3dError 0.50,     MarkerAngle 60,
3dRmsError 0.50,     SensorRmsError 0.10, MinimumMarkers 3,
MinSpread1 0.00,     MinSpread2 0.00,     MinSpread3 0.00
```

Table 3: The content of a RIG file

In background of the calibration process stands the pivot calibration [13] [14] technique that performed either as static or dynamic. The static calibration is undertaken, if all markers of the tool are always visible by the tracker that means the markers lie on a plane and the tool will be not used as a probe. And dynamic pivot calibration is performed, if all markers of the tool are visible and they must be moved during the data sampling by the tracker to detect all markers, or the tool has a tip and it will be used as probe during the navigation. The static tool calibration looks like as follows.

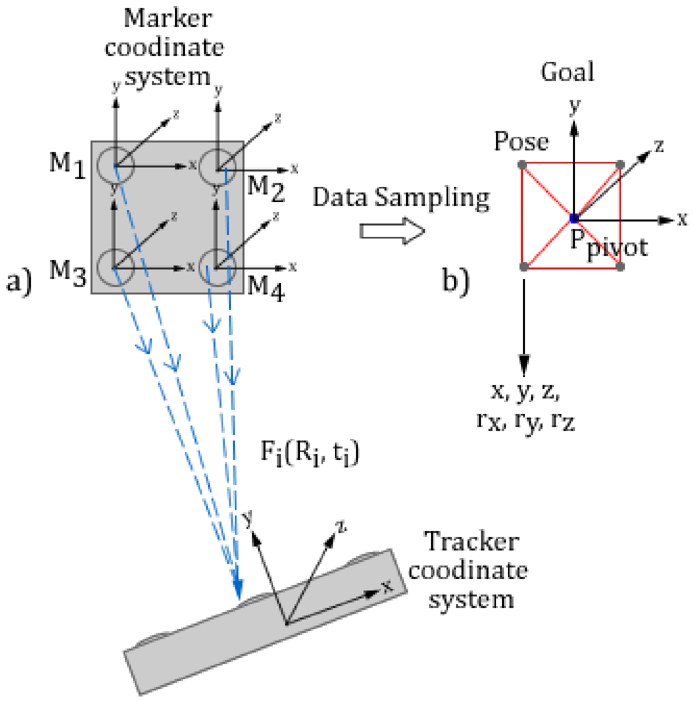


Figure 11: Static pivot calibration process

Each marker has own coordinate system at the marker body. The goal of the static tool calibration is to determine the unknown point P_{pivot} (centroid of the marker rigid body) most likely at the middle of tool rigid body. For this, tracked tool is fixed on a point statically and at the same time the pose of each marker M_1, M_2, M_3 and M_4 is sampled by the tracker. P_{pivot} vector is constant in the marker coordinate system (Figure 11a). After pose sampling it occurs four different point clouds from each marker that contains coordinates x, y, z and rotations r_x, r_y, r_z. To calculate pivot point P_{pivot} on the tracker frame, the navigation system triangulates the exact location of the tool arrays in space and takes average of each coordinates and rotations of four markers (Figure 11b).

The dynamic or tip pivot calibration is more complex as static calibration. Figure 12 shows typical steps of a tool tip calibration process.

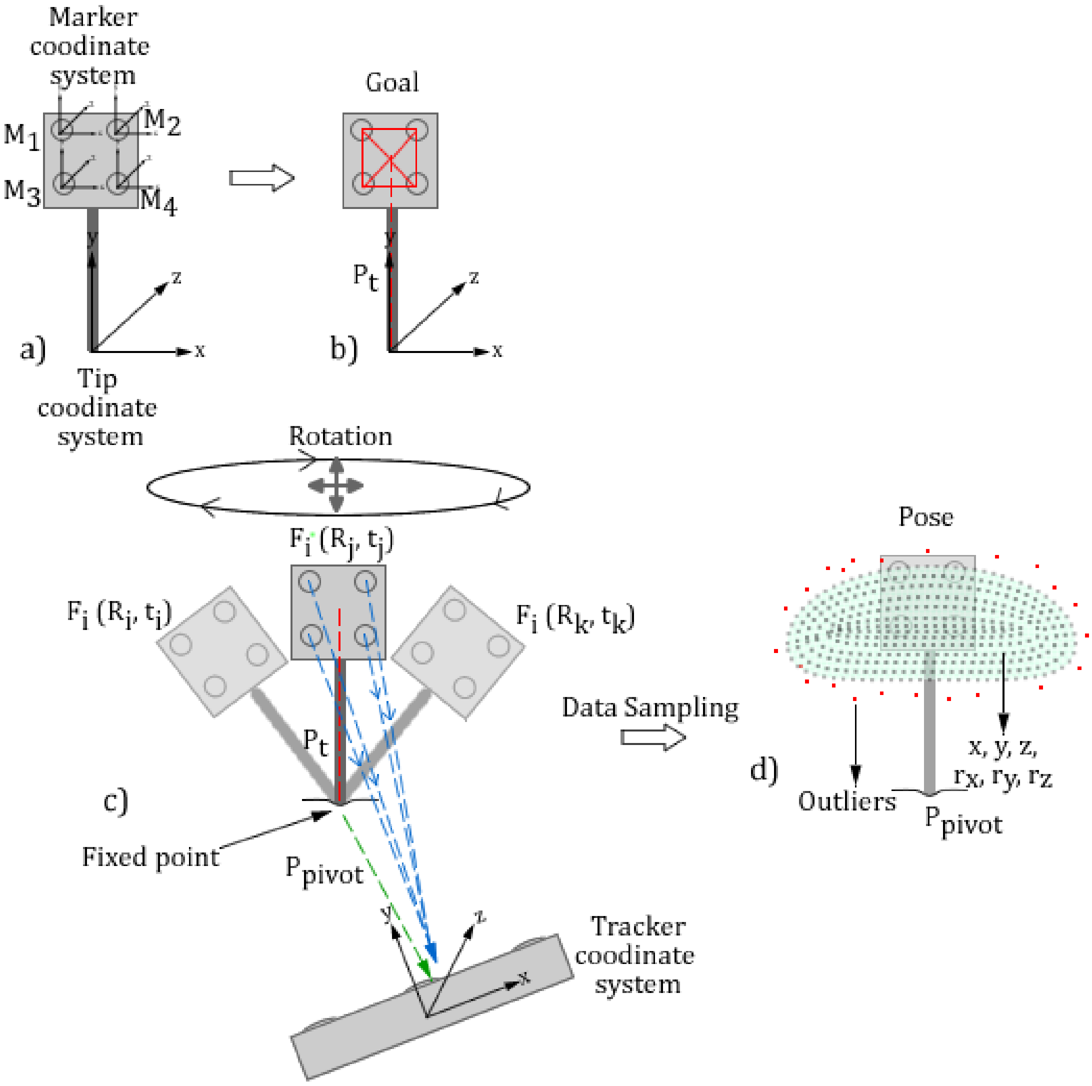


Figure 12: Dynamic pivot calibration process

Each marker has own coordinate system at the marker body and tip a). The goal of the tip calibration is to determine the unknown point P_t or rather P_{pivot} at the tip of tool rigid body b). For this, the tracked tool is rotated on a fixed point spherically and at the same time the pose of each marker $M_1,\ M_2,\ M_3$ and M_4 is sampled by the tracker. P_t and P_{pivot} vectors are constant in the marker coordinate system c). After pose sampling it is generated a half spherical point cloud from each marker that contains coordinates $x,\ y,\ z$ and rotations $r_x,\ r_y,\ r_z$ with some outlier poses. The four cloud spheres are concentric d).

To calculate unknown pivot point P_{pivot} on the tracker frame for i'th pose e.g. for $F_i\,(R_i\,,\ t_i\,)$ it is undertaken following calculation process [15]:

1. Determine the marker coordinate system (orthonormal base from 4 markers) in both poses.
2. Calculate the $F_i\,(R_i\,,\ t_i\,)$ frame transformation between marker and tracker frames, for both poses.
3. $F_i\,(R_i\,,\ t_i\,)$ takes the P_t vector to the pivot point P_{pivot} .
4. $F_i\ * P_t\ =\ P_{pivot}$.
5. First rotation by R_i , then translation by t_i .
6. $R_i\ * P_t\ +\ t_i\ = P_{pivot}$.
7. The unknowns are P_t and P_{pivot} .
8. Use many poses to compute P_t

The solution is obtained in a least squares manner by collecting multiple transformation samples e.g. $F_0\,(R_0\,,\ t_0\,),\ F_1\,(R_1\,,\ t_1\,)$ ... $F_n\,(R_n\,,\ t_n\,)$ while rotating the probe tool around its tip on a fixed point resulting in the following overdetermined equation system:

For i'th pose $R_i\ * P_t\ +\ t_i\ = P_{pivot}$: $(R_i\ - I)\begin{pmatrix} P_t \\ P_{pivot} \end{pmatrix} = -t_i$,

and for all performed poses: $\begin{pmatrix} R_0 & -I \\ R_1 & -I \\ \vdots & \\ R_n & -I \end{pmatrix}\begin{pmatrix} P_t \\ P_{pivot} \end{pmatrix} = \begin{pmatrix} -t_0 \\ -t_1 \\ \vdots \\ -t_n \end{pmatrix}$

This equation system is solved using the pseudo inverse [16]:

2.2.4 Registration

In order to determine the same location of at the patient's anatomy approached point with the tracked surgical instrument e.g. probe (anatomical landmark) in the patient images (fiducial landmark) exactly, the patient and the patient images must be registered with each other. This registration process is known as image-to-patient registration. AROSCOPE uses for this purpose the implementation of *igstkLandmark3DRegistration()* landmark-based registration method in IGSTK [15] that produces a closed-form solution to the least-squares problem of computing rigid body transformation parameters between two coordinate systems [17].

Point-based registration consists of computing the rigid-body transformation ($T_{(x)} = A_x + t$), where A is the rotation matrix and t the displacement or translation vector that best matches the preoperative image fiducials locations to the intraoperative patient fiducials. The inputs of point-based registration method are the image fiducials that obtained from patient images, which are identified usually in preoperative phase and patient fiducials that obtained from patient, which is determined in intraoperative phase. The output of the processing is the rigid transformation that brings the both fiducial datasets as close as possible according to a similarity measure.

Assume there are two corresponding fiducial sets I (I_1, I_2 ... I_n) and P (P_1, P_2 ... P_n) (Figure 13). The first fiducial set I refers to the image a) and second P to the patient fiducial set b) in their respective coordinate system. Superimposed are the points I_i the corresponding points P_i, so it can be measured the error rate. It means the deviation between the point I_i and point P_i. The lines between the corresponding points (blue and yellow spheres) give the root mean square (RMS) errors, so the distances between the transformed point I_i and the landmark point P_i: $RMS = {}^{image}I_{fiducial} - {}^{image}R_{patient} \cdot {}^{patient}P_{fiducial} - {}^{image}t_{patient}$ c).

In point-based registration it is required to find the 3D translation t and the rotation R that aligns one fiducial set of N points I_i with corresponding fiducial set P_i, $i = 1, 2, 3 \ldots N$. such that the distance d_i between corresponding fiducial sets is minimized in the RMS range. The RMS distance to be minimized is commonly termed the fiducial registration error (FRE), thus solves the main problem of registration to minimize the FRE during translation and rotation were: $FRE^2 \equiv \frac{1}{N}\sum_{i=1}^{N} |RI_i + t - P_i|$^2

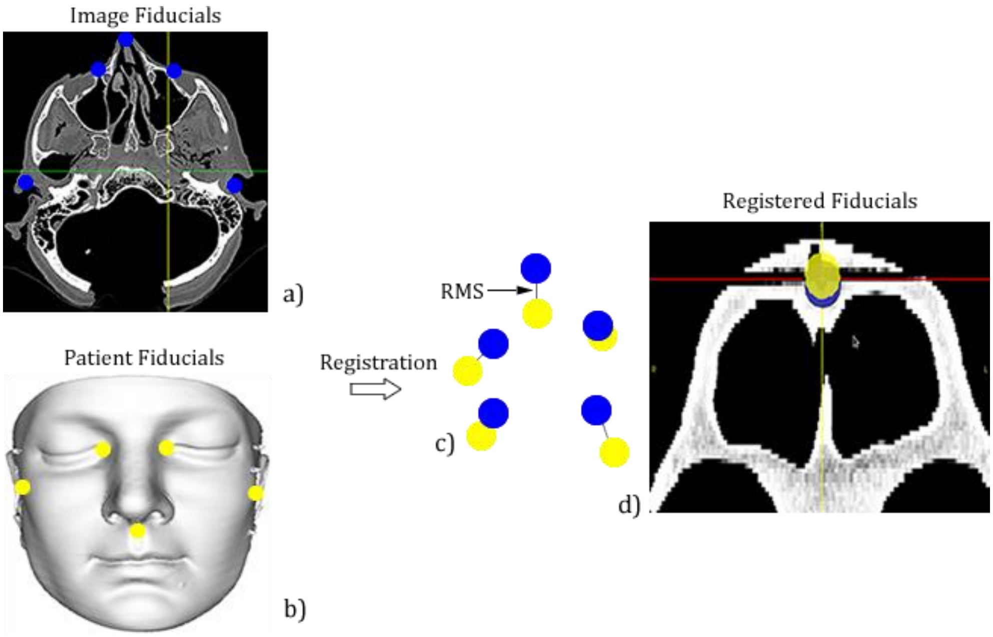
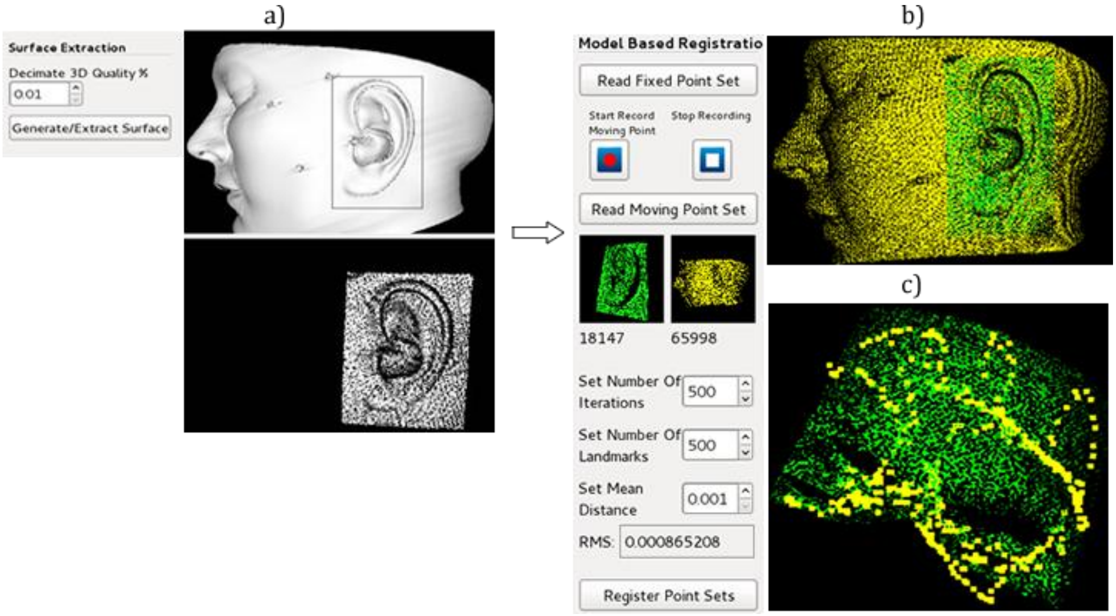


Figure 13: Point-based registration workflow

Along point-based registration there are other methods to register patient to image e.g. model-based called surface-based registration [18]. A medical application example for this technique is StoS (Surface to Surface) medical open-source tool [19] that developed in our research group. The StoS realizes a model bases registration with iterative closest point (ICP) algorithm [20]. The goal of this project was to get in the first step a point cloud in Cartesian coordinate system from segmented CT data set by using the surface extraction technique in The Visualization Toolkit (VTK), which is used as reference points for ICP (Figure 14a). The second point cloud is recorded by the patient with a tracker by moving the probe over the surface of the skin using the Image-Guided Surgery Toolkit (IGSTK), which is used as target points for the ICP algorithm b). In the second step the both point clouds are registered with the ICP and visualized accordingly. Finally after registration obtained transformation matrix is used for the patient navigation in IGSTK. In Figure 14c) it is visible a result of ICP registration. The green point set represents the extracted surface from the patient 3D image and yellow point set from tracked probe tool obtained points.



Decimate 3D Quality % 0.01

Set Number Of Iterations 500

Set Number Of Landmarks 500

Set Mean Distance 0.001

RMS: 0.000865208

Figure 14: Model-based registration tool

2.2.5 Patient tracking

Mathematically, the navigation is a representing a chain of matrix multiplication, which is included in following equations. The ${}^{x}T_{y}$ defines the homogeneous 4×4 transformation matrix that determines the position and orientation of a coordinate system y relative to a coordinate system x. R describes an orthogonal 3×3 rotation matrix, which x indicates the rotation of the coordinate system within the coordinate system y, t describes the 3x1 translation vector between the two coordinate systems and the elements, $s_1 - s_3$ elements specify scaling and aberrations of camera. Both matrices are multiplied in the homogeneous transformation matrix e.g. ${}^{x}T_{y}$:

${}^{DRF}T_{tool} = ({}^{tracker}T_{DRF})^{-1} . \ {}^{tracker}T_{tool}$,which computes transformation from tracked tool relative to DRF coordinate system.

$$
{}^{x}T_{y} = \begin{pmatrix} & R & t & \\ s_1 & s_2 & s_3 & 1 \end{pmatrix} = \begin{pmatrix} r_{xx} & r_{yx} & r_{zx} & t_x \\ r_{xy} & r_{yy} & r_{zy} & t_y \\ r_{xz} & r_{yz} & r_{zz} & t_z \\ s_1 & s_2 & s_3 & 1 \end{pmatrix}
$$

To navigate the patient intraoperatively, AROSCOPE’s navigation module represents and uses standard medical navigation procedures with patient images e.g. CT, a tracker, DRF tool (attached in near of surgical area) and probe tool (indicates any surgical instruments) as navigation system components.

In order to build a coordinate system between with above listed components, first the used rigid bodies should be added into the tracking list of optotrak regarding the same order of their plugged ports in strobers and outcomes of tracked tools must be read and estimated properly. For this purpose it is implemented a driver class in AROSCOPE, which manages tool connections. A tracking list for DRF and probe tool looks like:

```
// Add DRF tool in tracking list
if(RigidBodyAddFromFile(
drfID,             // ID associated with this rigid body.
1,                 // First marker in the rigid body.
drfRigFile,        // RIG file containing rigid body coordinates.
OPTOTRAK_QUATERN_RIGID_FLAG |
OPTOTRAK_RETURN_QUATERN_FLAG))  // Flags.
{ goto ERROR_EXIT; }
// Add probe tool in tracking list
if(RigidBodyAddFromFile(
probeID,              // ID associated with this rigid body.
1 + numberOfMarker,   // First marker in the rigid body.
probeRigFile,         // RIG file containing rigid body
                         coordinates.
OPTOTRAK_QUATERN_RIGID_FLAG |
OPTOTRAK_RETURN_QUATERN_FLAG))  // Flags.
{ goto ERROR_EXIT; }
```

Table 4: Adding an used rigid body into the tracking list

Optotrak serves the transformation format information such as quaternions, rotations and angles, etc. of tracked tools for DRF and probe as follows:

```
// Obtain pose information from DRF tool
drfCX= pRigid[drfID].transformation.quaternion.translation.x;
drfCY= pRigid[drfID].transformation.quaternion.translation.y;
drfCZ= pRigid[drfID].transformation.quaternion.translation.z;

drfRX= pRigid[drfID].transformation.quaternion.rotation.qx;
drfRY= pRigid[drfID].transformation.quaternion.rotation.qy;
drfRZ= pRigid[drfID].transformation.quaternion.rotation.qz;
drfRW= pRigid[drfID].transformation.quaternion.rotation.qw;

// Obtain pose information from probe tool
probeCX= pRigid[probeID].transformation.quaternion.translation.x;
probeCY= pRigid[probeID].transformation.quaternion.translation.y;
probeCZ= pRigid[probeID].transformation.quaternion.translation.z;

probeRX= pRigid[probeID].transformation.quaternion.rotation.qx;
probeRY= pRigid[probeID].transformation.quaternion.rotation.qy;
probeRZ= pRigid[probeID].transformation.quaternion.rotation.qz;
probeRW= pRigid[probeID].transformation.quaternion.rotation.qw;
```

Table 5: Optotrak tool pose information

The quaternions [21] that obtained from a tracked tool is represented formally by $q_w + i\, q_x + j\, q_y + k\, q_z$, then the equivalent 3x3 matrix to represent the rotation R and 3x1 translation vector t, is:

$$\begin{pmatrix} 1-2*qy^2-2*qz^2 & 2*qx*qy-2*qz*qw & 2*qx*qz+2*qy*qw & t_x \\ 2*qx*qy+2*qz*qw & 1-2*qx^2-2*qz^2 & 2*qy*qz-2*qx*qw & t_y \\ 2*qx*qz-2*qy*qw & 2*qy*qz+2*qx*qw & 1-2*qx^2-2*qy^2 & t_z \\ s1 & s2 & s3 & 1 \end{pmatrix}$$

The first three rows and column of the matrix indicate the operations with rotations R and last column the translation t.

To allow the surgeon a DRF-based patient and microscope navigation and in order to get the possibility to move the patient, or tracker during the surgery, first the DRF pose should be written in a 4x4 transformation matrix and then the inverse matrix from it should be calculated. Because all poses or measurements of other tracked tools should be relative to the DRF tool to protect the coordinate based relationships between tracker, patient, and tools. For this purpose it is used above formula to convert quaternions to a transformation matrix for tools. The matrix operations occur in the Visualization Toolkit (VTK) and implemented as follows:

```
// DRF tool operations
vtkMatrix4x4 drfM = vtkMatrix4x4::New();
vtkMatrix4x4 drfInverseM = vtkMatrix4x4::New();

// Fill the matrix entries
drfM.SetElement(0, 0, 1 – 2 * (drfRY * drfRY)- 2 *(drfRZ * drfRZ));
drfM.SetElement(0, 1, 2 * (drfRX * drfRY) – 2 * (drfRZ * drfRW));
drfM.SetElement(0, 2, 2 * (drfRX * drfRZ) + 2 * (drfRY * drfRW));
drfM.SetElement(0, 3, drfCoorX);

drfM.SetElement(1, 0, 2 * (drfRX * drfRY) + 2 * (drfRZ * drfRW));
drfM.SetElement(1, 1, 1 – 2 * (drfRX * drfRX)- 2 *(drfRZ * drfRZ));
drfM.SetElement(1, 2, 2 * (drfRY * drfRZ) – 2 * (drfRX * drfRW));
drfM.SetElement(1, 3, drfCoorY);

drfM.SetElement(2, 0, 2 * (drfRX * drfRZ) – 2 * (drfRY * drfRW));
drfM.SetElement(2, 1, 2 * (drfRY * drfRZ) + 2 * (drfRX * drfRW));
drfM.SetElement(2, 2, 1 – 2 * (drfRX * drfRX)– 2 *(drfRY * drfRY));
drfM.SetElement(2, 3, drfCoorZ);
drfM.SetElement(3, 0, 0);
drfM.SetElement(3, 1, 0);
drfM.SetElement(3, 2, 0);
drfM.SetElement(3, 3, 1);

// Inverse the matrix
vtkMatrix4x4::Invert(drfM, drfInverseM);

// Probe tool operations
vtkMatrix4x4 prM = vtkMatrix4x4::New();
vtkMatrix4x4 probeCoorM = vtkMatrix4x4::New();

// Fill the matrix entries
prM.SetElement(0, 0, 1 - 2 * (prRY * prRY) - 2 * (prRZ * prRZ));
prM.SetElement(0, 1, 2 * (prRX * prRY) - 2 * (prRZ * prRW));
prM.SetElement(0, 2, 2 * (prRX * prRZ) + 2 * (prRY * prRW));
prM.SetElement(0, 3, prCoorX);

prM.SetElement(1, 0, 2 * (prRX * prRY) + 2 * (prRZ * prRW));
prM.SetElement(1, 1, 1 - 2 * (prRX * prRX) - 2 * (prRZ * prRZ));
prM.SetElement(1, 2, 2 * (prRY * prRZ) - 2 * (prRX * prRW));
prM.SetElement(1, 3, prCoorY);

prM.SetElement(2, 0, 2 * (prRX * prRZ) - 2 * (prRY * prRW));
prM.SetElement(2, 1, 2 * (prRY * prRZ) + 2 * (prRX * prRW));
prM.SetElement(2, 2, 1 - 2 * (prRX * prRX) - 2 * (prRY * prRY));
prM.SetElement(2, 3, prCoorZ);

prM.SetElement(3, 0, 0);
prM.SetElement(3, 1, 0);
prM.SetElement(3, 2, 0);
prM.SetElement(3, 3, 1);

// Multiple inverse transformation matrix of DRF and transformation
// matrix of probe tool save result in probeCoorM matrix
vtkMatrix4x4::Multiply4x4(drfInverseM, prM, probeCoorM);
```

Table 6: Converting quaternions to a transformation matrix

Now the following rows of matrix *probeCoorM* give the new coordinates relative to the DRF for probe tool.

```
// Get new probe tool coordinates
float probeCoorX= probeCoorM->GetElement(0, 3);
float probeCoorY= probeCoorM->GetElement(1, 3);
float probeCoorZ= probeCoorM->GetElement(2, 3);
```

Table 7: Probe tool coordinates relative to DRF

Figure 15 describes the main coordinate systems and the sub-components of a surgical patient navigation. a) indicates DICOM formatted radiological image data as a voxel volume in the image coordinate system by navigation software (*image*). b) real patient or phantom with attached DRF on it (*DRF*). c) tracker coordinate system (*tracker)* and d) probe tool in optical tracker coordinate system (*probe*). The transformations between the coordinate systems of the individual components are given by:

$$ {}^{probeTip}T_{image} = (C)\ {}^{image}T_{DRF}^{\ -1} .(B)\ {}^{DRF}T_{tracker}^{\ -1}\ .\ (A)\ {}^{probeTip}T_{tracker} $$

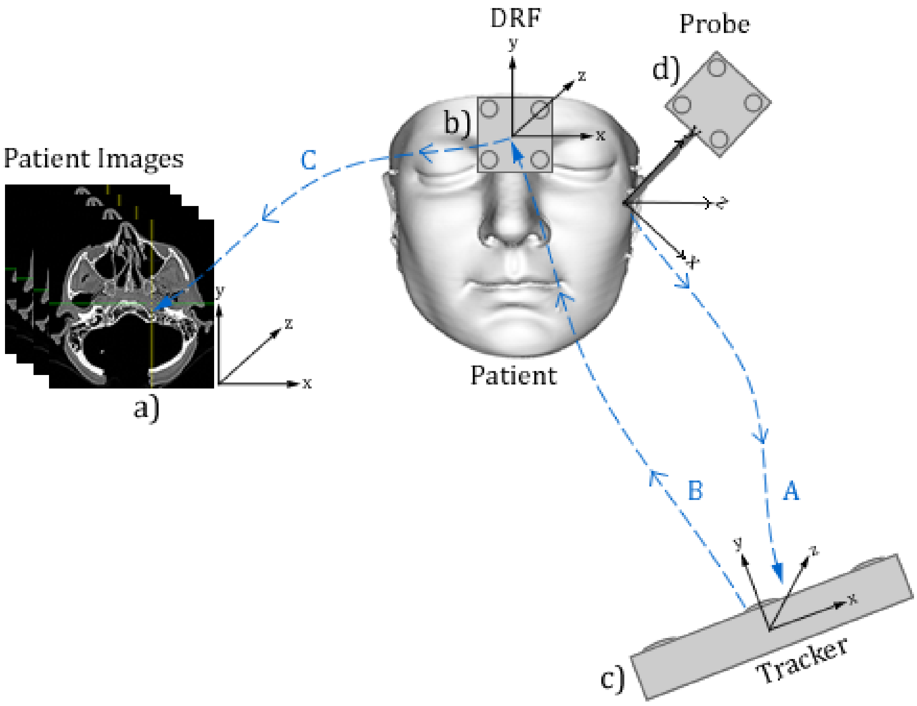


Figure 15: Patient navigation and coordinate system

The interaction and workflow of individual components of a navigation system in computer-assisted surgery (CAS) looks like schematically:

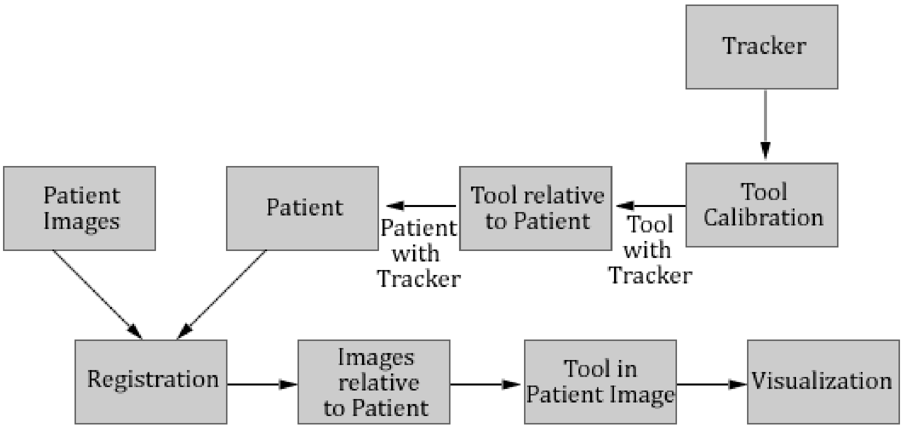


Figure 16: Interaction of computer-assisted surgical navigation components

The patient images and real patient, which is linked to the tracker with the attached and tracked tool, are registered using registration of fiducials with each other. After registration of both fiducial sets the position of tracked tool is calculated first relative to the patient and then to patient images, in which happens a suitable visualization.

2.2.6 Fiducials

In order to achieve the highest possible precision in the surgical field independent of which registration strategy is used, the fiducials that are defined during the patient to image registration, must be selected as close as possible to the surgical area and this enclose ideally [22]. Otherwise, the error is increased when fiducials are defined far from the operation area and the application accuracy of navigation system is affected negatively. At the same time the defined fiducials must be placed at prominent anatomical sites that can be clearly identified and easily accessed from the outside. For a successful navigation, optimally it must at least three fiducials should be defined. Additionally one important criterion for selecting registration fiducials is the exclusion of symmetrical configurations. To prevent that the planned fiducials are set symmetrically, thereby producing a wrong transformation matrix, symmetrical landmark constellations should be detected already automatically and visualized during the fiducial design. In Figure 17 are the examples of registration point arrangements represented.

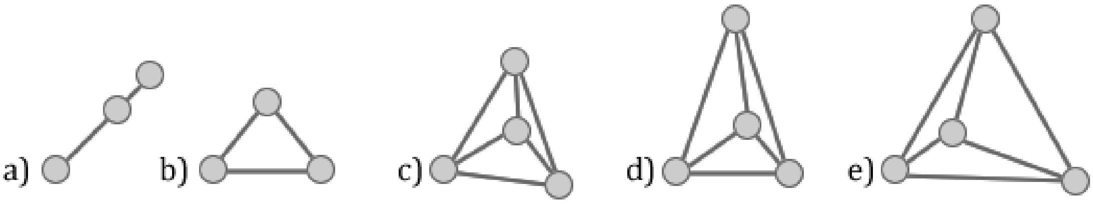


Figure 17: Arrangements of fiducials

The registration cannot be clearly calculated, if the registration fiducials collinear with a 3-point registration a) in an equilateral triangle b), or placed in an isosceles triangle. When using four registration points, the arrangement in a regular tetrahedron c) represents the most unfavorable constellation. d) and e) illustrate other examples of placement of four registration points in space with different edge lengths.

These criteria to be agreed in the ENT surgery is not always possible, as for example, during operations in the sphenoid sinus, mastoids, or petrous bones, since the target area is located inside the patient and defining the registration fiducials from outside of the patient effects large deviations in regard navigation accuracy. For this reason it is developed the RS technology [5] (described in 2.2.1.2) that allows positioning fiducials in the inside of patient anatomy and it guarantees submillimetric application accuracy.

2.2.7 Microscope tracking

Microscope tracking in AROSCOPE follows the same procedures similarly as in patient tracking. Three rigid bodies are attached to the back side of the microscope head in mutually orthogonal planes each with four active markers. This clearly ensures that each side of the microscope is currently seen and tracked by optotrak intraoperatively. So through internal quality criteria of optotrak are always an adapted optical target clearly selected and tracked. In the case that two rigid bodies are visible at the same time by the tracker, navigation module selects best visible rigid body regarding calculated smallest orientation angle of view parameters *pitch, roll* and *yaw* and tracks it. The measurements of these rigid bodies occur relative to the tracker.

The microscope head of the neurosurgical operation microscope can be moved over the patient in 6-DOF to obtain the currently needed perspective of the surgical area during surgery. To model this in the navigation and augmented reality modules, the transformation between the microscope head and patient must be known. This is composed of a rotation and a translation together and mathematically tracking of microscope position relative to the camera coordinate system is represented:

$$ {}^{camera}T_{tracker} = \left({}^{camera}T_{microscope}{}^{-1} \right) . \left({}^{microscope}T_{tracker}{}^{-1} \right) $$

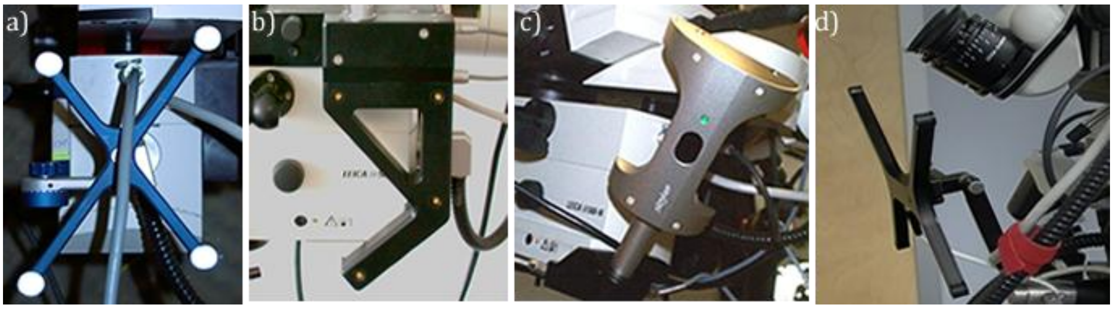


Figure 18: Microscope frame adaptation from some companies

The above figure lists already exiting microscope rigid body setups from some medical technology companies a) Brainlab, b) Medtronic, c) Stryker and, d) Radonics with active or passive markers and the Figure 19 represents the rigid body adaptation setup of microscope tracking in AROSCOPE.

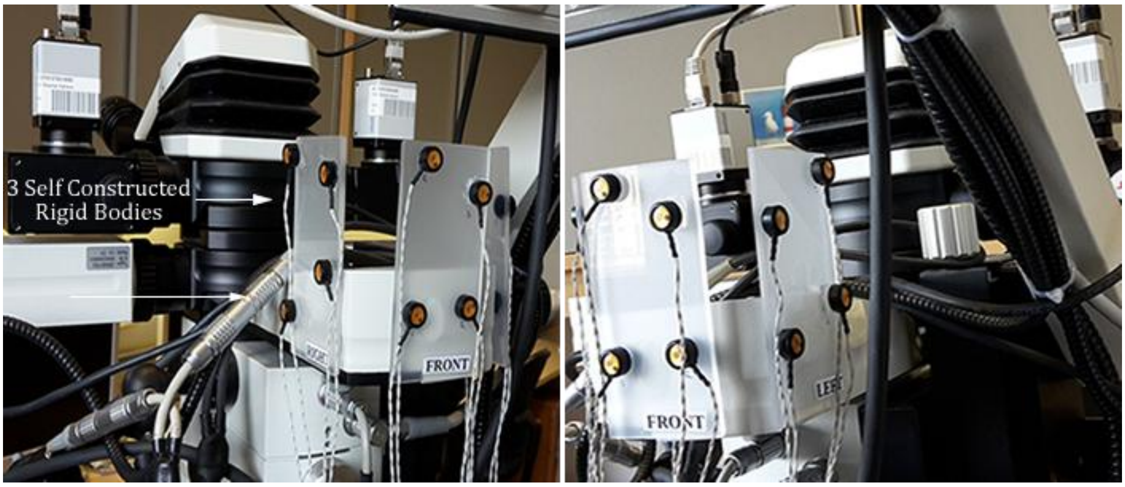


Figure 19: Self constructed rigid bodies adapted on the microscope head

Figure 20 describes the main coordinate systems and the sub-components of microscope navigation. The images a) b) and c) are the same described in Figure 15. The image d) represents the microscope head in the tracker coordinate system (*microscope*).

The transformations between the coordinate systems of the individual components are given by:

$${}^{image}T_{microscope} = (C)\ {}^{microscope}T_{tracker}{}^{-1} \ .\ (B)\ {}^{DRF}T_{tracker} \ .\ (A)\ {}^{image}T_{DRF}$$

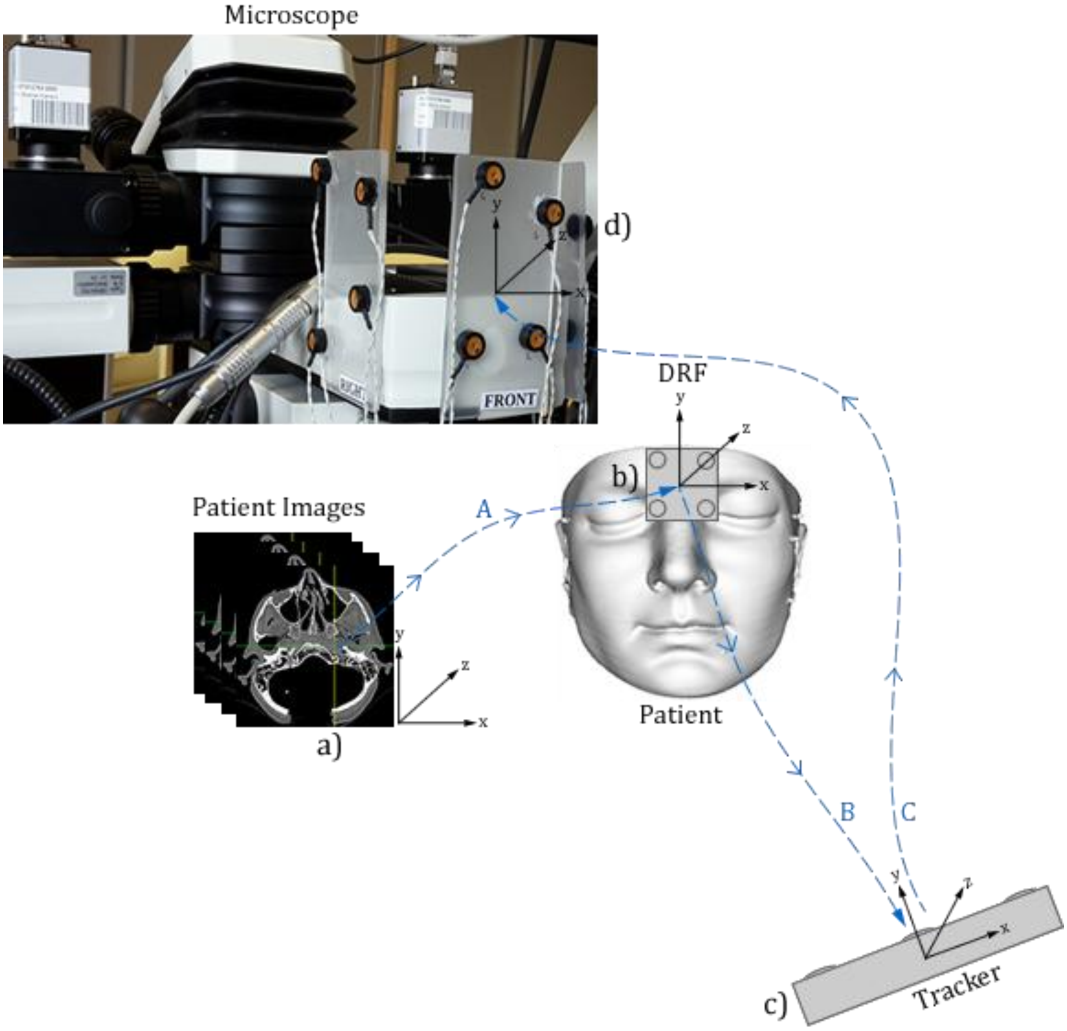


Figure 20: Microscope navigation and coordinate system

2.2.8 Hand-eye calibration / Calibration pattern tracking

For an accurate alignment of the injected 3D images inside the microscope view, the unknown transformation from the calibration pattern to the tracker coordinate system as well as the transformation from the camera to the calibration pattern coordinate system needs to be estimated simultaneously [23], in order to know the position of cameras relative to tracker coordinate system (Figure 23d). This last component should be inserted as a part into the navigation’s coordinate system. In order to navigate the calibration pattern, a self constructed rigid body is rigidly mounted to the pattern and navigated intraoperatively (Figure 21a).

The relationship between calibration pattern, tracker and microscope is built mathematically as follows:

$^{calibrationPattern}T_{microscope}$ = (C) $^{microscope}T_{tracker}^{-1}$. (B) $^{calibrationPatternTool}T_{tracker}$.
(A) $^{calibrationPattern}T_{calibrationPatternTool}$

To incorporate the 2D calibration pattern coordinate system to 3D tracker coordinate system and calculate the 3D transformation matrix from it, the tracked probe tool is used as an interface between tracker and rigid body of calibration pattern (Figure 21a). For this purpose is performed a hand-eye calibration technique. The underlying idea behind this approach is to determine the tracker - calibration pattern marker transformation based on consecutive measurements of tracker and marker transformation matrices at different poses. For this: first, the tip of probe tool is placed at origin point (*0*, *0*). That means on cross point of pattern's first square, where the black squares touch each other at the top left corner, and the translation pose x, y, z of point $P0$ is saved. Secondly the probe tool is moved until the last cross point in x direction starting from the origin point $P0$ and the pose $P1$ is obtained again. Finally the probe tool is moved until the last cross point in y direction starting from the origin point $P0$ and the last pose $P2$ is obtained. Therefore the coordinates in $P1$ and $P2$ directions are given as two 3x1 translation vectors. To find the coordinates of $P3$ from given other vectors, the cross product is applied and the observed result with other elements is written in a 4x4 matrix T:

$T = \begin{pmatrix} \vec{x} & \vec{y} & \vec{z} & \vec{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}$, where

$\vec{x} = \left(\frac{P1 - P0}{\|P1 - P0\|}\right), \vec{y} = \left(\frac{P2 - P0}{\|P2 - P0\|}\right), \ \vec{z} = \vec{x} \ x \ \vec{y},$

$\vec{t} = P0,$

$x =$ cross product,

$\|.\| = \sqrt{x_1^2 + \ ... + x_n^2}$ Euclidean norm and

$(\vec{x}, \vec{y}, \vec{z} \ , \vec{t}) \in IR^3$.

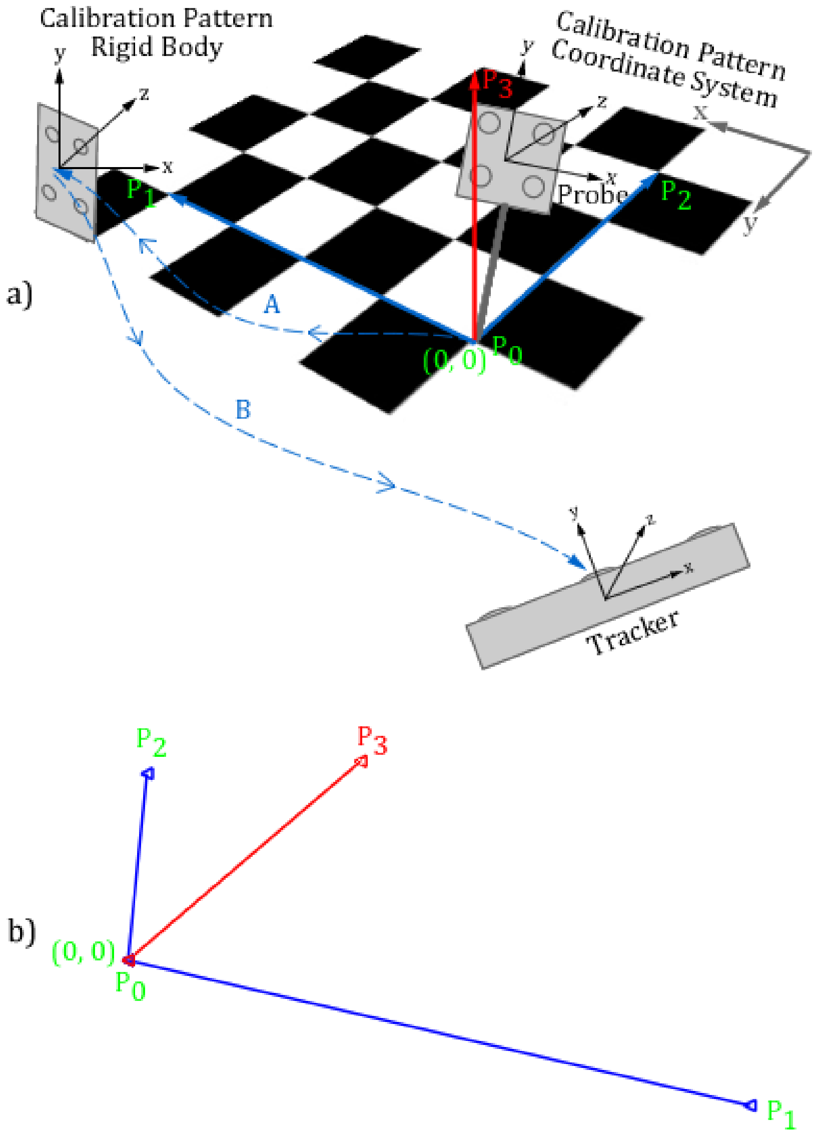


Figure 21: Hand-eye calibration

Figure 21b represents the graphical view after determining the cross product $P3$ from given other two vectors. The solution of the crross product computation repsesented in a 4x4 matrix, which defines transformation from calibration pattern coordinate system to the calibration pattern rigid body coordinate system:

$$\begin{pmatrix} 0.9994 & -0.0273 & -0.0103 & 54.8569 \\ 0.0087 & -0.0485 & 0.9987 & 49.8332 \\ -0.0334 & -0.9984 & -0.0483 & -60.9967 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 22 describes the main coordinate systems and the sub-components of hand-eye calibration/checkerboard navigation. a) represents the rigid transformation from calibration pattern coordinate system (*calibrationPattern*) to the calibration pattern tool coordinate system (*calibrationPatternTool*). b) describes calibration pattern coordinate system in tracker coordinate system (*tracker*) and finally c) describes calibration pattern coordinate system in microscope coordinate system (*microscope*).

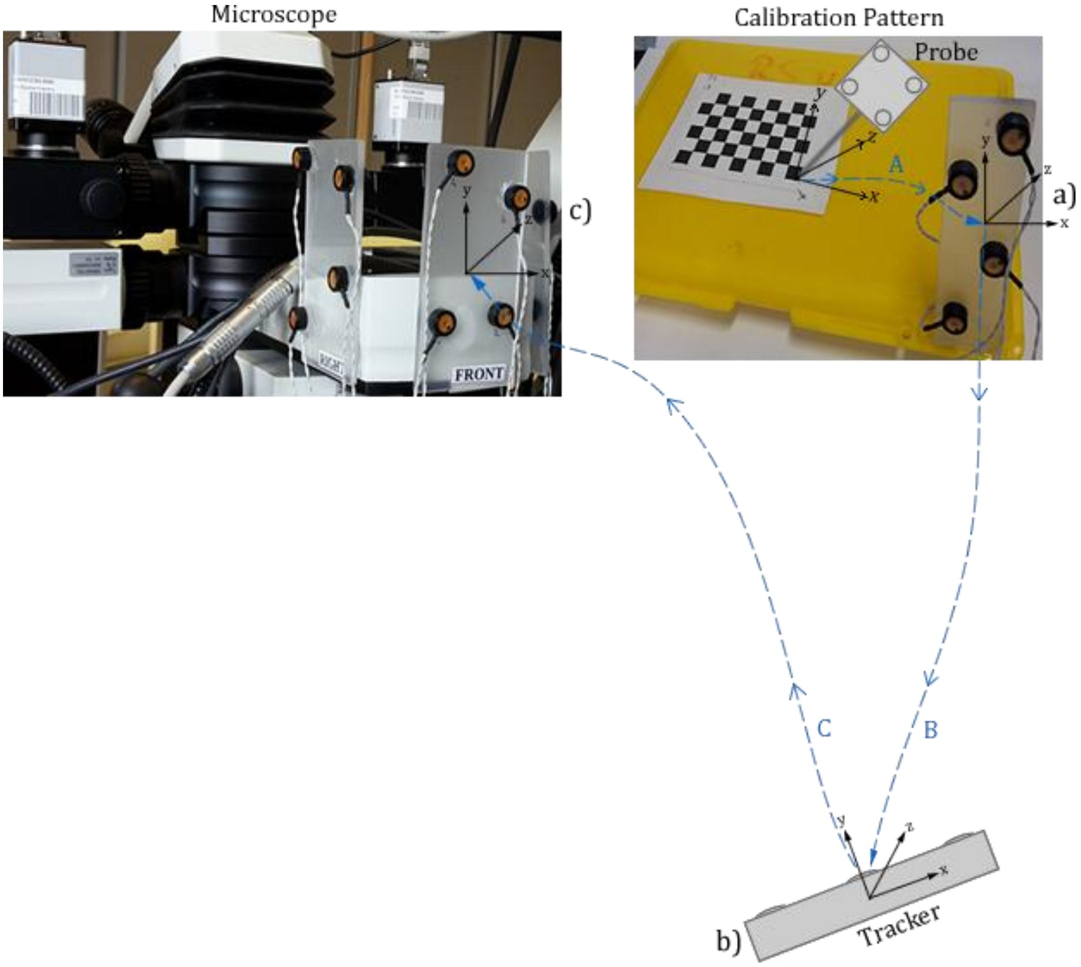


Figure 22: Hand-eye calibration and coordinate system

Now the complete transformation chain operations for patient, microscope and calibration pattern tracking yield:

$$ {}^{image}T_{camera} = (D)\ {}^{camera}T_{microscope}^{\ -1} \cdot (C)\ {}^{microscope}T_{tracker}^{\ -1} \cdot (B)\ {}^{DRF}T_{tracker} \cdot (A)\ {}^{image}T_{DRF} $$

Figure 23d represents the relationship of rigid body transformation from microscope coordinate system (*microscope*) to the camera coordinate system (*camera*), which is calculated by hand-eye calibration using Direct Linear Transformation (DLT) algorithm (described in 3.5 and 4.2).

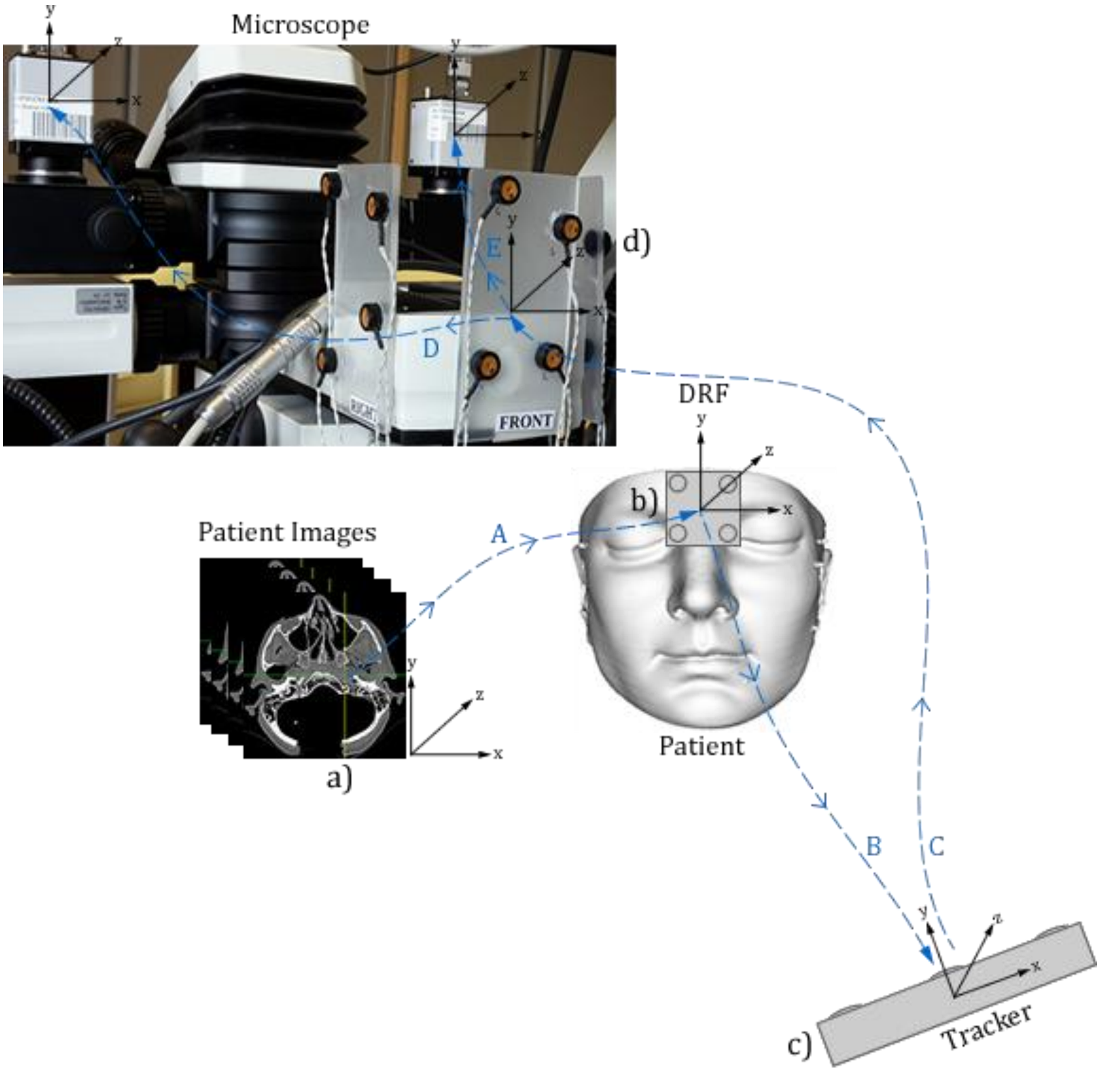


Figure 23: AR transformation chain and coordinate system

2.3 Visualization

Displaying the tracking and patient images processes in AROSCOPE takes place in navigation module, which visualizes patient images in orthogonal views such axial, sagittal, coronal and multiplanar (MPR) using IGSTK’s visual representation classes. The patient registration occurs in two steps. First the image fiducials are set using mouse pointer in patient images and visualized as a blue colored ellipsoid in all views. In second step the patient fiducials are defined using tracked probe tool on patient skin and visualized as a yellow colored ellipsoid. After a successfully registration with

IGSTK's landmark-based 3D registration class and starting navigation with calculated RMS error a crosshair on patient images represents the current position of the probe relative to the images in real-time and the navigation module gives feedbacks about such as visibility of patient, probe or microscope rigid bodies and current coordinates of all tracked tools. Figure 24 illustrates a screenshot of orthogonal views during the patient navigation after a patient-image registration (described in 2.2.4) process in navigation module (described in 5.7.2). The crosshairs represents the current position of the probe tool on the patient. The RMS error or rather deviation between registered image fiducial and patient fiducial (both in diameter 4 mm) is visible to the eye.

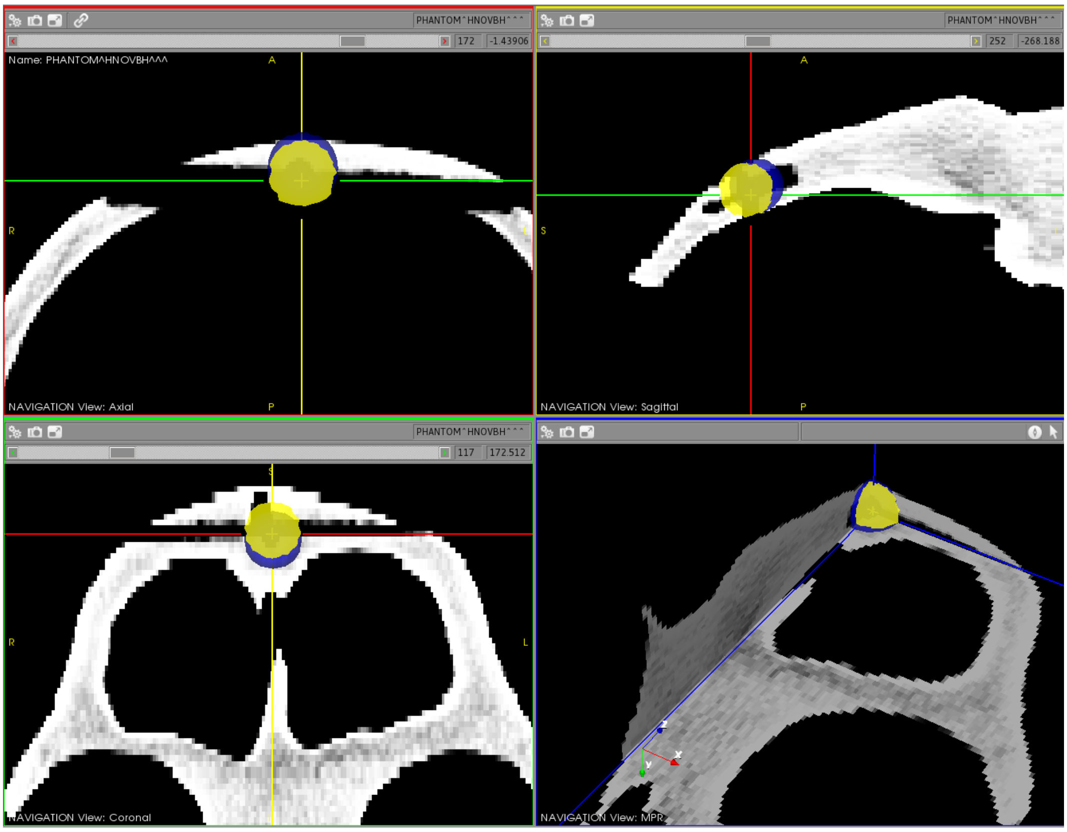


Figure 24: Visualization of patient navigation

2.4 Results

In this section the surgical instruments and other components such as patient, microscope, calibration pattern, are integrated as an input source in the software module using optical tracking methods. Thereby 4-marker tools or self constructed rigid bodies can be used for the navigation of patient and microscope. The communication of optotrak system with the workstation occurs via USB interface. The workstation can obtain output information from tracker and handle movements of patient, tracked tools, microscope, or calibration pattern in real-time.

Although there is used a manual point-based registration *igstkLandmark3DRegistration()* method with using 4 or 5 anatomical landmarks, it was possible to achieve submillimetric navigation accuracy (RMS between 0.3 - 0.8 mm) with respect to output of the implemented registration algorithm [17] in IGSTK and repetitive performed registrations while developing and testing in multiple times, which is undertaken by selecting one of the patient fiducial as target and pointing the tip of the probe on this fiducial [24] (Figure 24). During visualization of whole process it was possible keeping all processed tasks under view and navigation module gave clearly and understandable feedbacks to the user.

CHAPTER 3 - Camera Calibration

3.1 Introduction

Camera calibration is an elementary component and necessary step to project (transform) 3D scene on a 2D plane to aim the establishing a mathematical relationship between world and image coordinates in order to ensure the mutual conversion of points.

A central area of computer vision is the 3D interpretation of images that captured by a camera. Based on the image data objects or their positions can be determined in the space. For this purpose, it is very important that the intrinsic such as focal distance, radial lens parameters, image origin, etc. and extrinsic parameters such as rotation and translation parameters of the camera model are determined in any form. This determination of camera parameters is called camera calibration and is an important issue in computer vision, because the accuracy and robustness of the reconstruction or rater projection (transformation from word coordinate into image coordinate) depends on the accuracy of the calibration process and it is greatly influenced by the quality of this step.

There are various camera calibration techniques [25] [26] [27] that use images of a specific calibration pattern such as 3D reference object-based calibration, 2D plane-based calibration, or 1D line-based calibration, which uses several shots of a known scene and auto-calibration, and which uses several shots of an unknown scene. For the auto-calibration technique it is required only image point correspondences to determine internal camera parameters and achieve a precise camera calibration. It is used multiple uncalibrated images of unstructured scenes in contrast to classic camera calibration; auto-calibration does not require any special calibration objects in the scene. Hence in this technique a large number of parameters need to be estimated, resulting in a much harder mathematical problem.

Camera calibration technique, which uses a 3D calibration pattern, usually consists of two or three printed calibration patterns orthogonal to each other (Figure 25 a) whose geometry in 3D space is known and one shot of pattern is enough to calibrate the camera. The 2D calibration pattern consists of one planar calibration pattern b) shown at few different arbitrary positions and orientations to the camera during calibration process. The 1D line-based calibration method c) is a relatively new technique and it requires to use a calibration object with two or more collinear points e.g. a string of balls hanging from the ceiling, that distances are known. In this method a camera can be calibrated by observing a moving line around a fixed point.

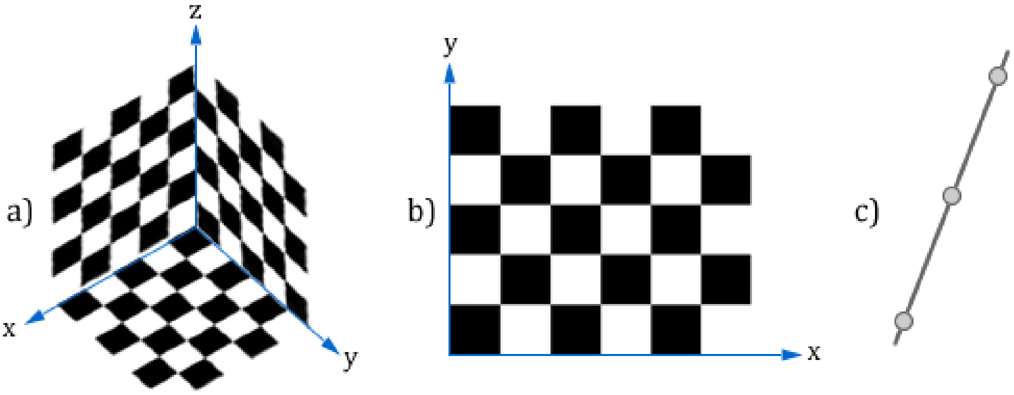


Figure 25: 3D, 2D and 1D calibration patterns

In this thesis a 2D plane-based (calibration pattern with known geometry) camera calibration technique according Zhang's [26] camera calibration method using open-source OpenCV library is undertaken, in order to calibrate industrial stereo cameras (Baumer TXG06c) mounted on the microscope head and offer live video streams from microscope to the workstation in resolution 776×578 px. The camera calibration process is always performed with a fixes zoom and focus values of microscope lenses.

In this section the mathematical background of stereo camera calibration and camera models is discussed and explained in detail, which is implemented as a part of AROSCOPE's calibration module.

3.2 Pinhole camera model

The pinhole camera model [28] describes the mathematical relationship between the 3D coordinates of a scene recorded by the camera and its corresponding projection on the 2D image plane called perspective projection and it is used to model basic geometry of projected 3D point onto a 2D surface. If an image is captured with a camera, it projects the 3D scene to 2D image plane. It means every point in the 3D world coordinate gets mapped to the 2D plane coordinate on the taken image.

Figure 26 describes a pinhole camera model. The camera is placed at the optical center *C* and optical axis parallel to Z_c . The three axes of the coordinate system are referred to as X_c, Y_c, Z_c . The axis Z_c points in the viewing direction of the camera and is referred to as the principal point. The 3D plane is the front side of the camera or image plane, which intersects with axes X_c and Y_c. The point *P* indicates a 3D point in the real world. The aim is: trying to capture that onto a 2D image plane P_c . The image plane represents the 2D plane that gets after taking an image and actually contains the image, which is represented after capturing a picture. In this case, the 3D point *P(X, Y, Z)* gets mapped to camera's image plane at image point $P_c\ (u, v)$. The distance between the focal plane and this image plane is called the focal length *f* of the camera and this parameter describes the focus value of the camera.

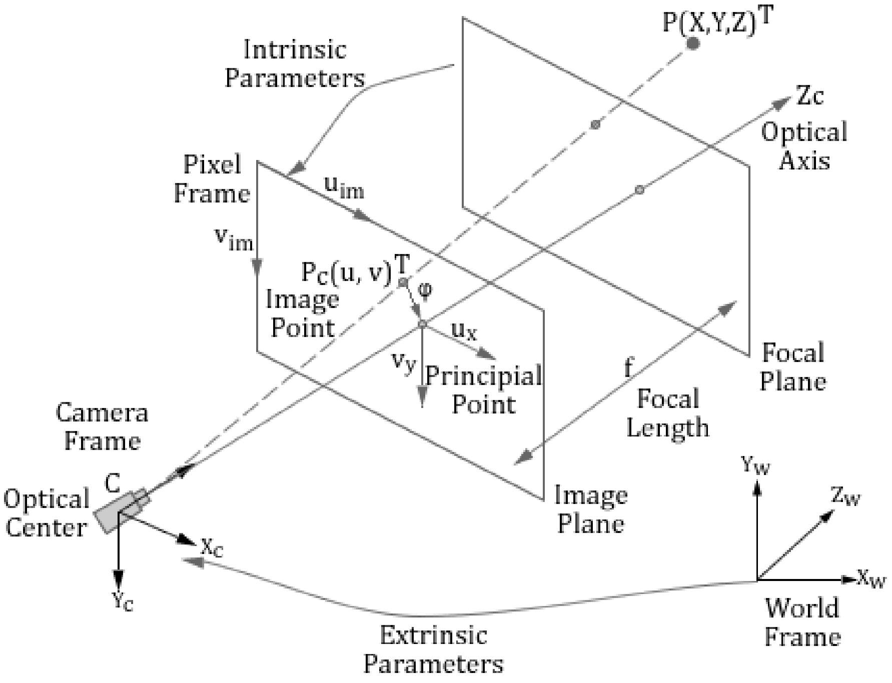


Figure 26: Pinhole camera model

To calibrate the camera, first the camera calibration matrix should be found (perspective projection matrix), which maps 3D *P* to 2D P_c . The P_c is calculated using similar triangle as:

$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y}$ this gives: $u = \frac{fX}{Z}, v = \frac{fY}{Z}$

This relation can be expressed in simple matrix notation using homogeneous coordinates [29] (projects a 3D scene onto a 2D image plane) for P_c by:

$$\varphi \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$ where $\varphi = Z$ is the homogeneous scaling factor.

It can be here verified that this indeed generates the point $P_c = (u, v, w) = \left(\frac{fX}{Z}, \frac{fY}{Z}, 1\right)$. P is still not in homogeneous coordinates.

In the next step, if the origin of the 2D image coordinate system does not coincide with where the Z axis intersects the image plane, that means the origin of the pixel coordinate system is not at the top-left pixel of the image defined but, the origin of the pixel coordinate system corresponds to the principal point $(u, v)^{\mathrm{T}}$ and located at the center of the image, it is needed to translate P_c to the desired origin. This translation by perspective projection equation defined by:

$$u = \frac{fX}{Z} + t_u, \qquad v = \frac{fY}{Z} + t_v$$

This relation can be expressed in a similar form as:

$$\varphi = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f_x & \gamma & t_u \\ 0 & f_y & t_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$ where γ is the parameter describing the skew of the two image axes.

In last step the perspective transformation P can be written in homogenous coordinate system:

$$\varphi \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f_x & \gamma & t_u \\ 0 & f_y & t_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{xx} & r_{yx} & r_{zx} & t_x \\ r_{xy} & r_{yy} & r_{zy} & t_y \\ r_{xz} & r_{yz} & r_{zz} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ w \end{pmatrix}$$

The last (red) matrix in perspective transformation defines the world coordinate system, the last and last but one (red and green) together defines the camera coordinates system and all three matrices (red, green and blue) define the image coordinate system. Above all the blue matrix defines the intrinsic K and green matrix $(R,\ t)$ the extrinsic parameters of camera. Thus the 3D point $P(X, Y, Z)$ is transformed into 2D point $P_c\ (u, v)$ and written as $P = K[R\ t]$.

The solution of camera calibration or projection from 3D world to 2D image generally described mathematically: $x' = KR[I_3| - t]X$, where x' is 3x1 2D image point vector, K is 3x3 5DOF calibration matrix (intrinsic), R is 3x3 rotation matrix, I_3 identity matrix, t is 3x1 translation vector of the pinhole (extrinsic) and X is 4x1object point vector, which is mapped to the x'.

If the equation combined in a matrix then: PX describes the calibration process, where P is a 3x4 projection matrix. In Figure 27, s indicates word coordinate system (red matrix), c camera coordinate system (red and green matrices) and φ defines image coordinate system (red, green and blue matrices).

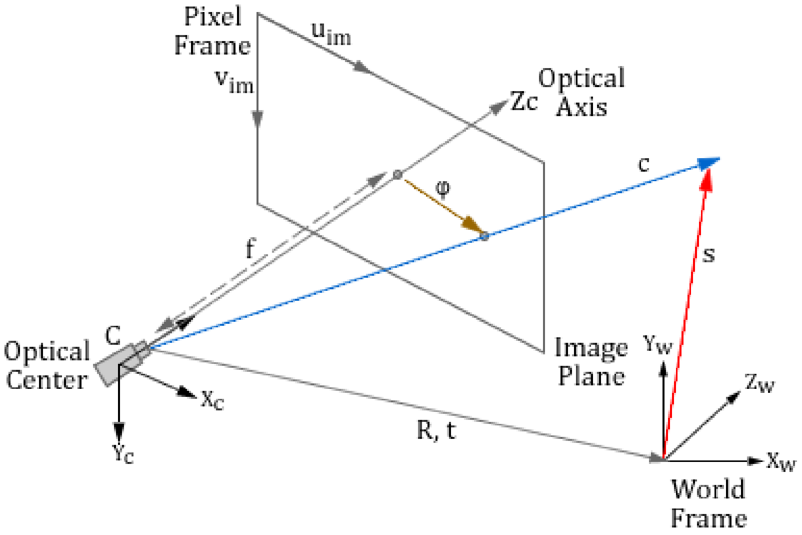


Figure 27: From camera coordinates to image plane coordinates

3.2.1 Camera parameters

A camera has 11 unknown parameters and in general, the world and image coordinate systems are related by a set of physical parameters such as intrinsic and extrinsic.

3.2.1.1 Intrinsic parameters

The intrinsic parameters $(f_x, f_y, \gamma, t_u, t_v)$ describe the optical, geometric and digital characteristics of the camera and allow a mapping between camera coordinates and pixel coordinates in the image coordinate system. The intrinsic parameters consist of:

1. Focal length $(f_x,\ f_y)$
2. *x*-coordinate of the image center point (O_x)
3. y-coordinate of the image center point (O_y)
4. Pixel scale in the *x* direction (s_x)
5. Pixel scale in the *y* direction (s_y)

From Figure 26 it can be seen the perspective projection, transformation between image plane and pixel coordinates, and the geometric distortion introduced by the optics.

In order to reach from camera coordinates to image plane coordinates, it is applied the perspective projection:

$$u = f\frac{X_c}{Z_c} = f\frac{R_1^T(P_w - t)}{R_3^T(P_w - t)}\ , \qquad v = f\frac{Y_c}{Z_c} = f\frac{R_2^T(P_w - t)}{R_3^T(P_w - t)}$$

And from image plane coordinates (u, v) to pixel coordinates$(u_{im},\ v_{im})$:

$$u = -(u_{im} - u_x).s_x\ , \qquad v = -\left(v_{im} - v_y\right).s_y$$

where $(u_x,\ v_y)$ are the coordinates of the principal point and s_x , s_y correspond to the effective size of the pixels in the horizontal and vertical directions. This equation can be written in a matrix notation:

$$\begin{pmatrix} u_{im} \\ v_{im} \\ w \end{pmatrix} = \begin{pmatrix} \frac{-1}{s_x} & \gamma & u_x \\ 0 & \frac{-1}{s_y} & v_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Now the pixel coordinates should be related to world coordinates, using following equation:

$$u_{im} - f s_x \frac{R_1^T(P_w - t)}{R_3^T(P_w - t)} + u_x\ , \qquad v_{im} - f s_y \frac{R_2^T(P_w - t)}{R_3^T(P_w - t)} + v_y$$

3.2.1.2 Extrinsic parameters

The extrinsic parameters (R, t) of the camera identify uniquely the transformation (rotation and translation with respect to the world coordinate system) between the unknown camera coordinate and the known world coordinate system. Determining these parameters means determining the translation vector between the relative positions of the origins of the two reference coordinate systems and the rotation matrix that brings the corresponding axes of the two coordinates into alignment. Basically the extrinsic parameters consist of:

1. Translation in the *x* direction (T_x)
2. Translation in the y direction (T_y)
3. Translation in the *z* direction (T_z)
4. Rotation about the *x*-axis (R_x)
5. Rotation around the y-axis (R_y)
6. Rotation around the *z*-axis (R_z)

That means, they describe the movement of the camera to the origin of the world coordinate system and also rotating around the three Euler angles. According to a perspective transformation it applies the following relationship between the world and image coordinate system:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t$$

Here the matrix (*u*, *v*, *w*) indicates the image coordinates with scaling factor and it is defined by a 3x3 rotation matrix *R*, which is multiplied with the 3D point (*X*, *Y*, *Z*) and shifted with a 3D translation vector *t*. Hence:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} r_{xx} & r_{yx} & r_{zx} & t_x \\ r_{xy} & r_{yy} & r_{zy} & t_y \\ r_{xz} & r_{yz} & r_{zz} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \\ 0 \end{pmatrix}$$

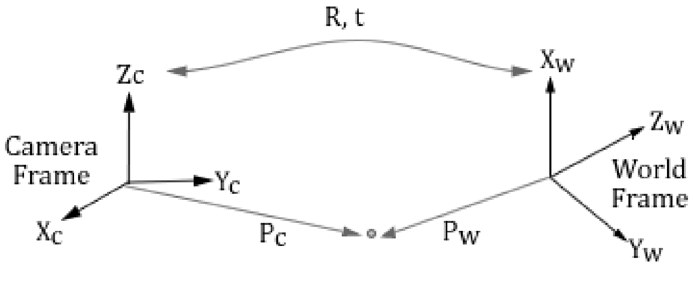


Figure 28: Extrinsic camera parameters

To combine the intrinsic and extrinsic camera parameters using homogenous coordinates:

$$M_{intr} = \begin{pmatrix} \frac{-f}{s_x} & \gamma & u_x \\ 0 & \frac{-f}{s_y} & v_y \\ 0 & 0 & 1 \end{pmatrix} \quad , \; M_{extr} = \begin{pmatrix} r_{xx} & r_{yx} & r_{zx} & -R_1^T T \\ r_{xy} & r_{yy} & r_{zy} & -R_2^T T \\ r_{xz} & r_{yz} & r_{zz} & -R_3^T T \end{pmatrix}$$

$$\begin{pmatrix} u_h \\ v_h \\ w \end{pmatrix} = M_{intr}\, M_{extr} \begin{pmatrix} X \\ Y \\ Z \\ w \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \\ w \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ w \end{pmatrix},$$

where *M* = 3x4 projection matrix.

3.3 Distortion

If it is taken an image with a camera, that uses either plus or minus lenses, the images have often distortions. This means that the projection of a world point on the image plane does not exactly correspond to what is expected. Already with the naked eye distortion of the image formed are often recognizable. Especially at the edges, a stronger curvature is perceived. A small distortion can already lead to a major distortion in the reconstruction and therefore it can cause large errors by the camera calibration process, if it is not handled correctly. A distinction is made between two different distortions; radial or tangential distortion. Figure 29 illustrates the kind of distortions for radial distortions a) no distortion (orthoscopy), b) barrel distortion, which occurs if minus lenses are used and c) pincushion, which occurs if plus lenses are used [30].

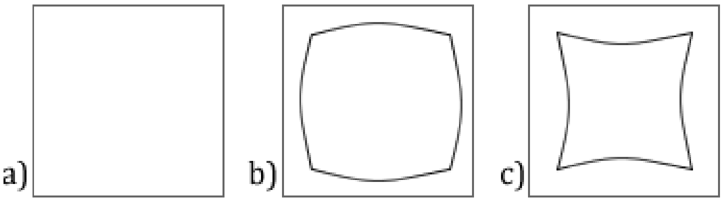


Figure 29: Kind of lens distortions

The radial distortion scales the distance of a pixel to focus. The various points (circular) distorted the focus away or toward focus radially. But the directional vector arising from the focus is preserved. If a pixel using the distance to the focus and the respective direction vector describes a distorted pixel P_v can be transferred to an undistorted pixel P_u by the distance of the undistorted point is determined to focus.

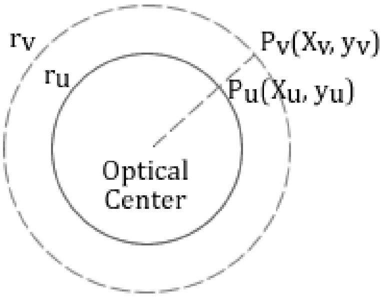


Figure 30: Radial distortion

If the image is more distorted then image center, it means of a pincushion distortion. That is, the distance of the image point to the focus is smaller than in the case of an ideal projection; The whole picture is more to the center of the picture is distorted (Figure 31a). If the image is distorted but the distortion is away from the center, it means of a barrel distortion (Figure 31b). Here again the distance of the image point is on focus greater than in an ideal projection. In Figure 31 are given the radial and tangential distortions. a) the ideal point (red) is shifted more to the center (distorted point: blue): Pincushion distortion. b) the ideal point (red) is shifted away from the center (distorted point: blue): Barrel distortion.

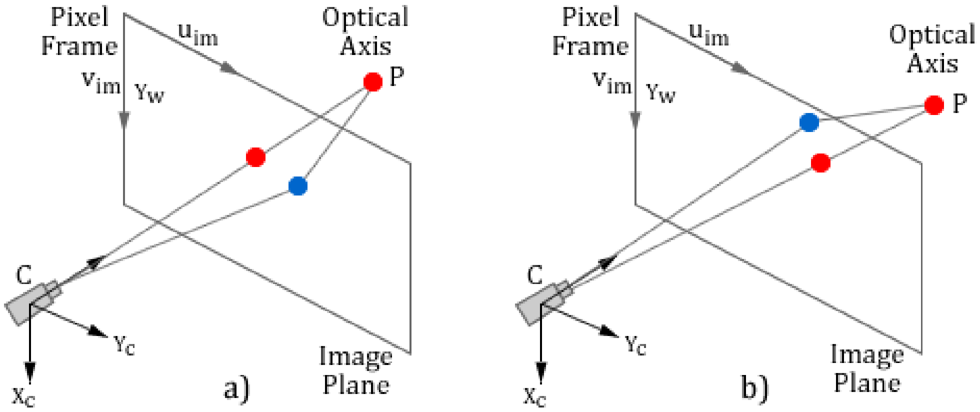


Figure 31: Representing of pincushion and barrel distortion on image plane

In the tangential distortion pixels are distorted using a tangential vector. The distance of the pixel to the focus here is not radially, but scaled along a tangent. Here, the direction vector is not necessarily obtained. It can be seen that P_v is the distorted image point and P_u of undistorted. Compared to the radial distortion, the tangential distortion carries a less significant role and is calculated only at high determination accuracy. They often occur in electro-optical processes, that uses electrical and optical effects.

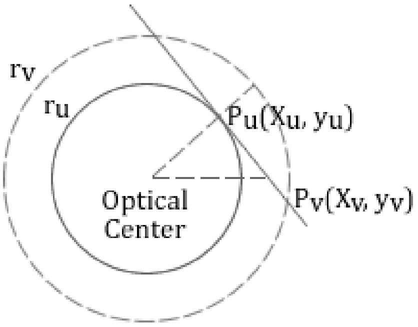


Figure 32: Tangential distortion

The distortions can be described:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t$$

$$x` = \frac{x}{z}, \qquad y` = \frac{y}{z}$$

$$x`` = x` \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x` y` + p_2\left(r^2 + 2x'^2\right)$$

$$y`` = y` \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1\left(r^2 + 2y'^2\right) + 2p_2 x` y`$$

where $r^2 = x'^2 + {y'}^2$

$$u = f_x * x'' + u_x\,, \qquad v = f_y * y'' + v_y$$

The $k_1,\ k_2,\ k_3,\ k_4,\ k_5,\ k_6$ are radial distortion coefficients and $p_1,\ p_2$ are tangential distortion coefficients.

3.4 Stereo camera calibration

To understand the stereo camera calibration and work with the stereo vision concept, it should be understood first the epipolar geometry. Since in stereo camera calibration it is used two cameras and it is an extension of pinhole camera model with an additional camera. The epipolar geometry [25] describes the fundamental geometrical relationship of two cameras that view a 3D scene from two distinct positions and illustrates relations between the 3D points and their projections onto the 2D perspective images that lead to constraints between image points (Figure 33).

Assume a point X in 3D world coordinate system and is imaged in two views by two pinhole cameras, at p_L in the first, and p_R in the second that are projections of point X onto the image planes. To find the correspondence point in p_R if a point in p_L is given, relative positions of camera and to find the 3D geometry of the scene, the epipolar geometry presents useful solution.

The points O_L and O_R indicate the center of projection for the two cameras. The perspective projection from 3D scene into the 2D image plane is found as described in pinhole camera model. The epipoles e_L (left image of O_L) and e_R (right image of O_R) are the points of intersection of the line joining the optical centres (baseline) with the image plane. The epipole is the image in one camera of the optical centers of the other camera. The epipolar plane π is the plane defined by a 3D point and the optical centres or equivalently, by an image point and the optical centres. This plane is described by three points $O_L,\ O_R,\ p$. The epipolar line is the straight line of intersection of the epipolar plane with the image plane. It is the image in one camera of a ray through the optical centre and image point in the other camera. All epipolar lines intersect at the epipole. A point in one image generates a line in the other on which its corresponding point must lie. The search for correspondences is thus reduced from a region to a line. This epipolar constraint arises because, for image points corresponding to the same 3D point, the image points, 3D point and optical centres are coplanar.

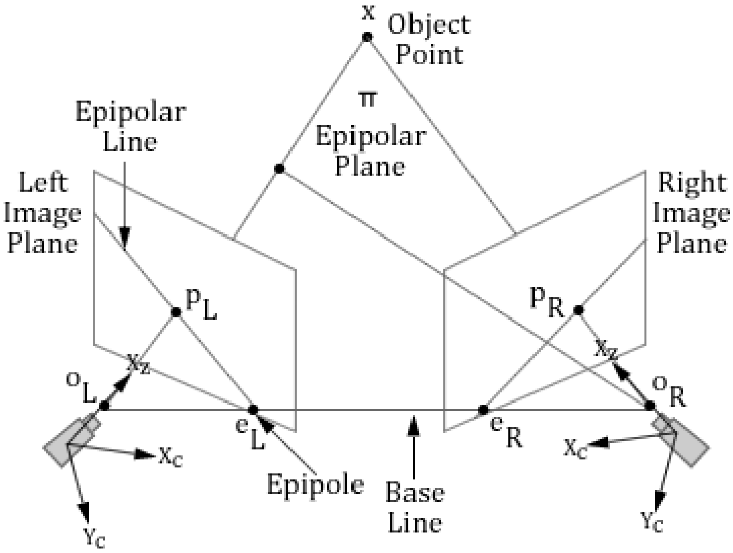


Figure 33: Epipolar geometry

Stereo camera calibration is very similar to single camera calibration and the whole complex processes of camera calibration are taken by the open-source OpenCV library [31]. To calibrate microscope cameras in AROSCOPE's calibration module, it is used following procedure stepwise.

1. Attach printed 2D calibration pattern in a flat surface and place it under the microscope's FOV.
2. Set the focus and zoom values of microscope optic properly to identify and visualize the squares on calibration pattern with good resolution.
3. Capture images from both cameras (optimally at least 6) in different arbitrary positions and orientations.
4. Be sure whether the whole pattern is obtained in the captured images.
5. Calibrate images for left and right camera.
6. Validate the calibration by capturing one more images of same pattern and calculate the difference between world and image coordinates by drawing a line between them according obtained camera calibration parameters.

The stereo camera calibration processed in step 5 is performed using OpenCV library, firstly by finding the corner parameters of the 2D calibration pattern. The function *cv::findChessboardCorners()* detects and locates the internal checkerboard corners relating to Harris corner detector algorithm [32].

The input parameters for this function are:

1. Captured images (Source calibration pattern view. It must be an 8-bit grayscale or color image).
2. Number of cross points along *x* (horizontal) and *y* (vertical) directions. *cv::Size (columns, rows).*
3. Flags. Various operation flags that can be zero or a combination of the some values:

The output of the function is found corners, namely object points and image points that used later in the calibration step. With using the function *cv::drawChessboardCorners()*, the determined corners can be visualized as a circle on the images to ensure whether the cross points are correct detected or not and the circles are connected with lines if the cross points was found correctly. This function takes as input captured image, number of cross points along x and y directions and output of the *cv::findChessboardCorners()* function, so corners.

In the second step the calibration process takes place. With the function *cv::calibrateCamera()* the cameras can be calibrated according the captured images separately regarding to Zhang's camera calibration algorithm [26].

The inputs of this function e.g. in order to calibrate the left camera, are:

1. Object points: Physical position (world coordinates) of the found calibration pattern corners in 3D space. 3xN or Nx3 matrix, where N is the total number of points in all views.
2. Image points: The location or rather projections of the calibration pattern corners on the image coordinate system in 2D space (corresponding image points) that observed by the left camera. 2xN or Nx2.
3. Checkerboard size: The resolution of the calibration image.

And the outputs of the function are:

1. Camera matrix: The input/output 3x3 camera matrix of the left camera
   $\begin{pmatrix} f_x^j & 0 & c_x^j \\ 0 & f_y^j & c_y^j \\ 0 & 0 & 1 \end{pmatrix}$, where $j= 0$ or 1
2. Distortion coefficient: The output vectors of distortion coefficients for left camera, 4x1, 1x4, 5x1 or 1x5 matrices.
3. Rotation matrix: The 3x3 rotation matrix for the left camera coordinate systems.
4. Translation matrix: The 3x1 translation vector of left cameras coordinate systems.

After calibration, the images for both cameras should be undistorted, if there is any distortion to correct. The *cv::undistortPoints()* function computes the ideal point coordinates from the observed point coordinates. The input and output parameters of this function for a camera are:

1. Image points: Observed point coordinates, 1xN or Nx1 2-channel for the left camera.

2. New image points: Output ideal point coordinates after undistortion and reverse perspective transformation for the left camera. If matrix P is identity or omitted, newImagePoints1 will contain normalized point coordinates.
3. Camera matrix: The input 3x3 camera matrix for the left camera.
4. Distortion coefficient: The input vectors of distortion coefficients for left camera, 4x1, 1x4, 5x1 or 1x5 matrices.
5. New camera matrix: The output 3x3 camera matrix for the first camera that contains new camera parameters.

To validate the performed calibration and represent the calculated projection error in that process, both cameras are validated respectively (Figure 36). For this purpose, one more images for both cameras from the same calibration pattern are taken and regarding calculated camera matrix horizontal virtual lines are drawn/projected between the detected first two cross points of squares. Thus the error of measurements by camera calibration is calculated and it is visible to the eye.

3.5 Direct linear transformation

In Figures 26 and 33 it is shown, how the relationship and associated coordinate system between world- (object) and camera-frame (image) in pinhole camera model works. Through stereo camera calibration it is achieved to the intrinsic parameters and it is known the related 2D location from a 3D point. By using the DLT it is estimated the current location of a 3D point $(X,\ Y,\ Z)$, which is located in object space based on related location in image space $L(u,\ v)$ and $R(u,\ v)$ for both cameras.

The Direct Linear Transformation [25] [33] is an algorithm to estimate the 2D location of the object or points on an object and to determine perspective transformation matrix using obtained different views of the object image.

The correspondence image points $[u_L,\ v_L]$ for the left, $[u_R,\ v_R]$ for the right camera and the unknown object points $[X,\ Y,\ Z]$ can be related through a series of constants:

$$u_L = \frac{L_1X + L_2Y + L_3Z + L_4}{L_9X + L_{10}Y + L_{11}Z + 1}, \qquad v_L = \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1},$$

$$u_R = \frac{R_1X + R_2Y + R_3Z + R_4}{R_9X + R_{10}Y + R_{11}Z + 1}, \qquad v_L = \frac{R_5X + R_6Y + R_7Z + R_8}{R_9X + R_{10}Y + R_{11}Z + 1},$$

where $L_1 \dots L_{11}$ coefficients for DLT and this equation can be written (described in 3.2.1.2) as:

$$\begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} = \begin{pmatrix} L_1 & L_2 & L_3 & L_4 \\ L_5 & L_6 & L_7 & L_8 \\ L_9 & L_{10} & L_{11} & 1 \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

For one calculated calibration point, it is given 7 known $(u_L,\ v_L,\ u_R,\ v_R,\ X,\ Y,\ Z)$, 22 unknowns $(L_1 \dots L_{11},\ R_1 \dots R_{11})$ and 4 equations.

For determining these 22 unknown parameters, it is needed at least 22 equations. This is done by choosing more than one calibration point in object space. For each additionally selected calibration point $x_i\ [X,\ Y,\ Z\ ]$, it is introduced 4 new equations, while the constants L_i and R_i remain the same parameters.

Six calibration points yield 24 equations; therefore it is necessary to acquire at least six calibration points $x_1\ [X,\ Y,\ Z\ ] \dots x_6\ [X,\ Y,\ Z\ ]$ to determine the unknown $(L_1 \dots L_{11})$ and $(R_1 \dots R_{11})$, if the cameras are not calibrated, otherwise only three.

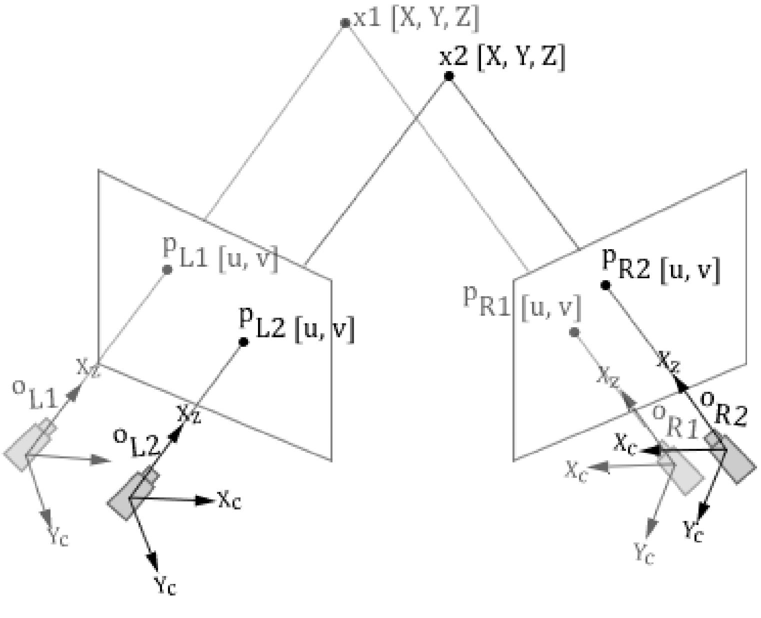


Figure 34: Estimation of DLT for two calibration points

If it is determined the components 7 known for at least six points, the equations can be written in a matrix form. For the selected two point in object space the equations for the left and right image point yields:

$$u_{L1} = \frac{L_1X_1 + L_2Y_1 + L_3Z_1 + L_4}{L_9X_1 + L_{10}Y_1 + L_{11}Z_1 + 1}, \qquad v_{L1} = \frac{L_5X_1 + L_6Y_1 + L_7Z_1 + L_8}{L_9X_1 + L_{10}Y_1 + L_{11}Z_1 + 1},$$

$$u_{L2} = \frac{L_1X_2 + L_2Y_2 + L_3Z_2 + L_4}{L_9X_2 + L_{10}Y_2 + L_{11}Z_2 + 1}, \qquad v_{L2} = \frac{L_5X_2 + L_6Y_2 + L_7Z_2 + L_8}{L_9X_2 + L_{10}Y_2 + L_{11}Z_2 + 1},$$

$$u_{R1} = \frac{R_1X_1 + R_2Y_1 + R_3Z_1 + R_4}{R_9X_1 + R_{10}Y_1 + R_{11}Z_1 + 1}, \qquad v_{R1} = \frac{R_5X_1 + R_6Y_1 + R_7Z_1 + R_8}{R_9X_1 + R_{10}Y_1 + R_{11}Z_1 + 1},$$

$$u_{R2} = \frac{R_1X_2 + R_2Y_2 + R_3Z_2 + R_4}{R_9X_2 + R_{10}Y_2 + R_{11}Z_2 + 1}, \qquad v_{R2} = \frac{R_5X_2 + R_6Y_2 + R_7Z_2 + R_8}{R_9X_2 + R_{10}Y_2 + R_{11}Z_2 + 1}$$

The above equations can be restructured e.g. for u_{L1} and v_{R2} as:

$u_{L1}(L_9X_1 + L_{10}Y_1 + L_{11}Z_1 + 1) = L_1X_1 + L_2Y_1 + L_3Z_1 + L_4$,

$u_{L1} = L_1X_1 + L_2Y_1 + L_3Z_1 + L_4 - u_{L1}L_9X_1 - u_{L1}L_{10}Y_1 - u_{L1}L_{11}Z_1$

$v_{R2}(R_9X_2 + R_{10}Y_2 + R_{11}Z_2 + 1) = R_5X_2 + R_6Y_2 + R_7Z_2 + R_8$,

$v_{R2} = R_5X_2 + R_6Y_2 + R_7Z_2 + R_8 - v_{R2}R_9X_2 - v_{R2}R_{10}Y_2 - v_{R2}R_{11}Z_2 \dots$

$u_{L1} \dots u_{LN}$, $v_{L1} \dots v_{LN}$ and $u_{R1} \dots u_{RN}$, $v_{R1} \dots v_{RN}$ for each calibration points.

Restructured equations can be written in matrix form and it is then achieved to a linear equation for each calibrated point:

$$\begin{array}{c} L\colon point\ 1 \\ \dots \\ R\colon point\ 2 \\ \dots \\ L\colon point\ N \\ R\colon point\ N \end{array} \underbrace{\begin{bmatrix} X_1\ Y_1\ Z_1\ 1\ 0\ 0\ 0\ 0 - u_{L1}X_1 - u_{L1}Y_1 - u_{L1}Z_1 \\ \dots \\ 0\ 0\ 0\ 0\ X_2\ Y_2\ Z_2\ 1 - v_{R2}X_2 - v_{R2}Y_2 - v_{R2}Z_2 \\ \dots \\ X_N\ Y_N\ Z_N\ 1\ 0\ 0\ 0\ 0 - u_{LN}X_N - u_{LN}Y_N - u_{LN}Z_N \\ 0\ 0\ 0\ 0\ X_N\ Y_N\ Z_N\ 1 - v_{RN}X_N - v_{RN}Y_N - v_{RN}Z_N \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ \dots \\ L_{11} \end{bmatrix}}_{T^*} = \underbrace{\begin{bmatrix} u_{L1} \\ \dots \\ v_{R2} \\ \dots \\ u_{LN} \\ v_{RN} \end{bmatrix}}_{D}$$

where, $AT^* = D$.

In this equation system only the parameters $(L_1 \dots L_{11})$ and $(R_1 \dots R_{11})$ are unknowns. After camera calibration solved calibration matrices for the left L and right R images are calculated and the parameters $(u_L,\ v_L,\ u_R,\ v_R,\ L_1 \dots L_{11},\ R_1 \dots R_{11})$ are known. Using the method of Last Square [34] to find the location of x_i $[X,\ Y,\ Z\ ]$, the solution defined as: $T^* = (A^tA)^{-1}A^tD$

3.6 Visualization

The realization of camera calibration in AROSCOPE takes place under the calibration module. This interacts with the stereo cameras that connected to the workstation via TCP/IP protocol and process live video stream with using Baumer-GAPI API [35]. The camera calibration, validation, and visual representation procedures are implemented using functionalities in OpenCV library. This module offers the possibility to use 2D square calibration pattern with known geometry in any size regarding square dimension and number of horizontal and vertical squares on calibration pattern as long as the FOV of the microscope optic allows.

Additionally, the captured images and calibration results are saved for both cameras separately to import and use them later in next sessions. Figure 35 represents the calibration phase in camera calibration module for left and right camera parallelly. In this chase the calibration pattern consist of 5x8 squares and 10 mm edge length. Colored spheres at the cross points on each squares describe here found corners of pattern. Below the main views the captured images from left and right camera are visible.

On the Figure 36 it is represented the validation after performed calibration. Here are drawn horizontal virtual lines between two squares, which real distances are known (20 mm), according calculated camera calibration for each camera. The red colored numbers give the error of the measurements. The content of calibration module window is described under section 5.7.4 in more details.

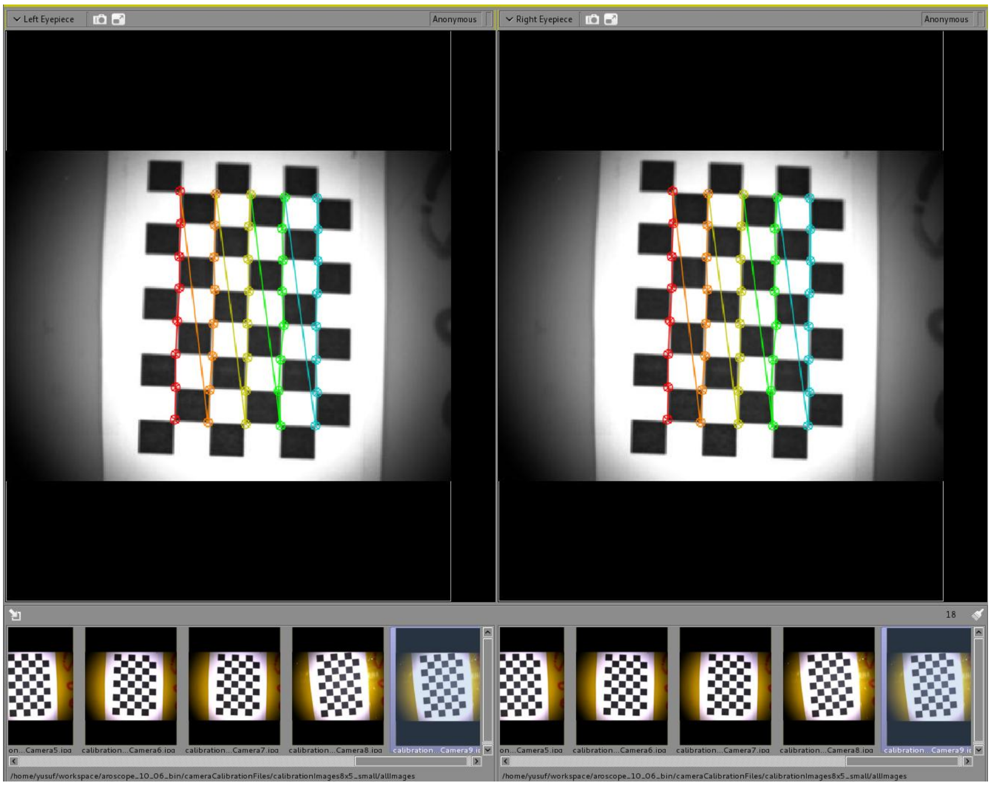
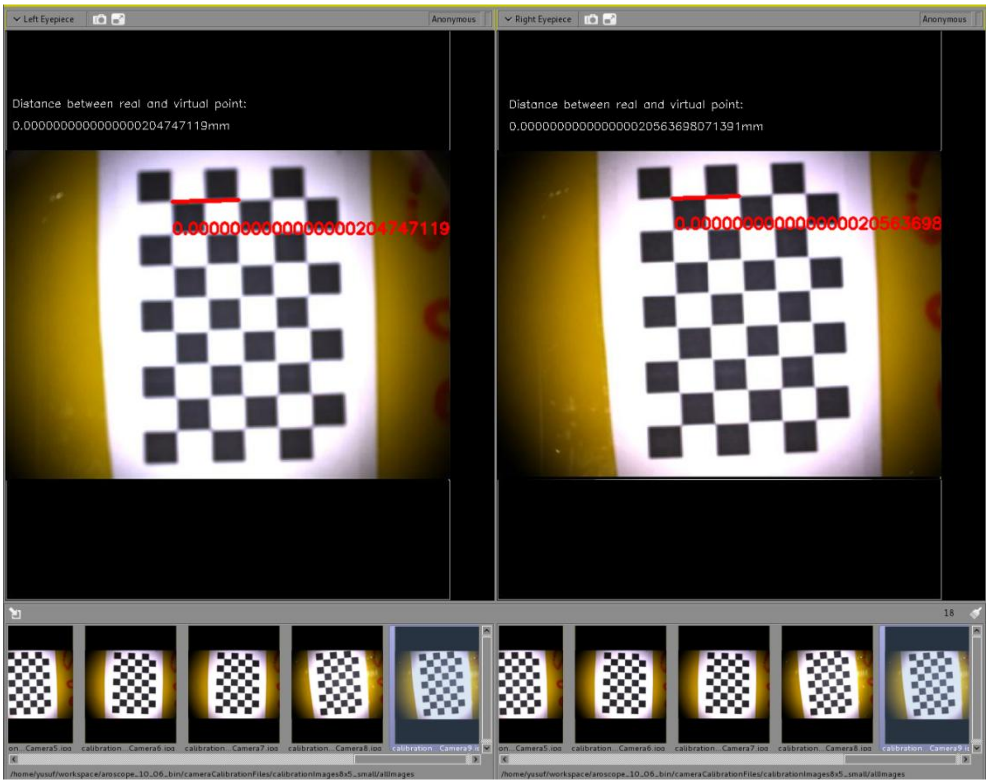

Left Eyepiece
Anonymous
Right Eyepiece
Anonymous
18
on...Camera5.jpg
calibration...Camera6.jpg
calibration...Camera7.jpg
calibration...Camera8.jpg
calibration...Camera9.j
/home/yusuf/workspace/aroscope_10_06_bin/cameraCalibrationFiles/calibrationImages8x5_small/allImages
/home/yusuf/workspace/aroscope_10_06_bin/cameraCalibrationFiles/calibrationImages8x5_small/allImages

Figure 35: Visualization of camera calibration process in calibration module



```
Distance between real and virtual point:
0.00000000000000000204747119mm
```

```
Distance between real and virtual point:
0.0000000000000000020563698071391mm
```

Left Eyepiece
Anonymous
Right Eyepiece
Anonymous
0.00000000000000000204747119
0.0000000000000000020563698
18
on...Camera5.jpg
calibration...Camera6.jpg
calibration...Camera7.jpg
calibration...Camera8.jpg
calibration...Camera9.j
/home/yusuf/workspace/aroscope_10_06_bin/cameraCalibrationFiles/calibrationImages8x5_small/allImages
/home/yusuf/workspace/aroscope_10_06_bin/cameraCalibrationFiles/calibrationImages8x5_small/allImages

Figure 36: Visualization of validation in calibration module

3.7 Results

In this section both industrial stereo cameras are integrated as an input source in the calibration module. For the evaluation of camera calibration process, which is undertaken with fixed zoom and focus parameters of microscope using Zhang's camera calibration technique, it is used two different 2D calibration patterns such as 5x8 squares with 10.0 mm edge length and 6x9 with 5.0 mm using camera resolution 776×578 px.

As result it was possible to achieve submillimetric re-projection error between 0.4 and 0.6 mm RMS after all performed calibration experiments, regarding validation of camera calibration (explained in 3.4 and 4.6) and returned RMS value of the function *cv::calibrateCamera()*, which is good enough to facilitate an accurate alignment of the injected 2D images or segmented 3D structures inside the microscope ocular.

CHAPTER 4 - Augmented Reality

4.1 Introduction

Augmented reality [36] [37] supplements real images with additional graphical information in real-time and thus creates is a link between the real and virtual world. In medicine or microscope embedded surgical interventions allows the ARS [38] the surgeon to see virtual features of 2D images (monocular) or segmented 3D anatomical structures (binocular), which are obtained from patient's radiological images in preoperative phase and overlay those extended information accurately in eyepieces of surgical microscope in intraoperative phase.

Figure 37 [39] describes the continuous scale ranging between the virtuality and reality, which AR appears as a mixed reality (MR) environment, with one part being real and the other one virtual. Therefore it comprises all possible variations and compositions of real and virtual objects.

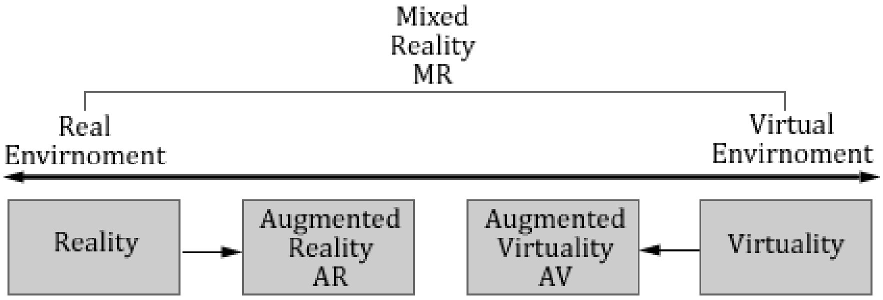


Figure 37: Milgram's reality-virtuality continuum

Since in minimally invasive procedures such as neurosurgery or ENT surgery the surgical microscopes are regularly used, so the correct virtual environment should be injected in the oculars as monocular or binocular to allow the surgeon eliminate the troubles like losing hand-eye coordination or attention, that comes into existence, when the surgeon have to change the viewing direction between microscope eyepiece and workstation constantly. Depending on this negative effect the adaptation and orientation problems can occur during the surgery, which distracts the surgeon from the real target to operate exactly. Thus, the main goals of augmented reality in minimal invasive surgeries are: Increasing the precision and ergonomics during surgery, reducing errors in the diagnosis trough 3D representation and intuitive navigation.

In the surgical microscopes with augmented reality extension, it differs with optical see-through and video see-through technology [40]. In a video see-through system, the surgeon sees the real image not directly, unlike optical see-through system [41], but receives an overlayed video image from workstation.

By using this technology the advantages lies by providing a significant assistance for surgeons and physicians and by reducing the risk to the patient in intraoperative phase, because by the representing of additional information directly in viewing direction of surgeon that provided by an ARS, potential errors can be detected better and prevented. This improvement is result that the surgeon must not turn away from the surgical area to see additional information on workstation and therefore can direct his entire attention to the surgery. The two main disadvantages such systems arise; the risk of the faulty overlay and the risk of a fail the overlay, where an incorrect injection causes more negative consequences then the failing of system. Since the surgeon, which relies on the faulty overlaying could inflict serious injury to the patient.

To design an AR setup in navigated microscopic image-guided surgery the following components are bound with each other and several application specific problems should be solved to avoid the above mentioned disadvantages:

1. Registration of patient (described in 2.2.4) with radiological images to know the exact position of the patient during the intervention. Science the overlaying of computer generated information is always relative to the patient’s anatomy.
2. Tracking of patient and other surgical instruments (described in 2.2.5) is an important part for surgeries which integrates a navigated surgical microscope. Science the exact location and orientation of patient and used medical instruments should be known in real-time, to get an accurate overlaying in the oculars.
3. Calibration of used cameras (described in 3.4), which are attached at the microscope, to receive real-time video stream from the surgery field is also a necessarily part of an AR setup to project a 3D scene on a 2D plane exactly and to know the characteristic features of camera, which are used by overlaying, in order to achieve a realistic fusion of virtual and the real world.

4. Tracking of microscope (described in 2.2.7) during the intervention is another challenge for the AR system. Since the head of microscope is moved constantly over the patient, the exact position information is a major knowledge to handle movements by overlayed image or structure and maintain the accuracy of AR view.
5. Segmentation of anatomical 3D structures (described in 4.4) from intervened surgical field which are overlayed in both oculars, produces a virtual environment and also a leading issue for a successfully surgery.

In this chapter the hard- and software components and technical implementation of an ARS are explored. Image injection technique in both eyepieces of the purpose-built binocular microscope (Leica M500 N) [42] is described and overlaying procedures of segmented 3D objects are processed.

4.2 Stereo augmented reality

After the camera calibration, the intrinsic and extrinsic camera paramters are known. Now to find the object pose from 3D word coordinate system estimated corresponding 2D image projection correspondences, which contains rotation and translation to transform object from initial 3D coordinates to 2D camera coordinate system. it is used the function *cv::solvePnP()* in OpenCV library [43] for both cameras, which accepts following inputs:

1. Object points (transformed object points after hand-eye calibration)
2. Image points (from both cameras)
3. Camera matrix (intrinsic) and
4. Distortion coefficient (from both cameras)

The outcomes of this function are:

1. Rotation vector: 3x1 rotation vector for each calibration image.
2. Translation vector: 3x1 transltion vector for each calibration image.

Where brings points from the world coordinate system to the camera coordinate system.

To convert 3x1 rotation vector to a 3x3 rotation matrix it is used the function *cv::rodrigues()*. After this operation the 3x1 translation vector is applied in the last row of rotation matrix. Therefore the 3x4 projection matrix (described in 3.2.1.2) is generated.

If it is known the intrinsic and extrinsic parameters of a camera, then a 3D scene can be transformed or projected into a corresponding 2D image plane. Since it is used two cameras in this case, then a stereo vision can be achieved from projected 2D points. To process this computation and build a transformation chain from patient image until microscope cameras, following procedures in AROSCOPE are performed.

1. Register patient and radiological images together using probe to get a 4x4 registration matrix (*imageToDrf*).
2. Perform camera calibration to calculate the 3x3 camera intrinsic (*cameraM*).
3. Perform hand-eye calibration with the probe on checkerboard pattern (*patternToPatternToolM*) to transform object points to the microscope tool coordinates, which is a 4x4 transformation matrix (*objPToMicroscopeM*).

```
// Chain from pattern to microscope (hand-eye calibration)
vtkMatrix4x4::Multiply4x4(m_PatternToolM,
                          m_PatternM,
                          m_patternToTrackerM);

vtkMatrix4x4::Multiply4x4(m_MicroscopeToolInverseM,
                          m_PatternToPatternToolM,
                          m_PatternToMicroscopeM);

// For each calibration image calculate m_objPToMicroscopeM
for(int i= 0; i< m_NumberOfImages; ++i)
{
  for(int j= 0; j< m_ObjectPoints.size(); ++j)
  {
    float v_in[4];
    v_in[0]= m_ObjectPoints[j].x;
    v_in[1]= m_ObjectPoints[j].y;
    v_in[2]= m_ObjectPoints[j].z;
    v_in[3]= 1.0;

    float *v_out= m_PatternToMicroscopeM[i]
                  ->MultiplyFloatPoint(v_in);

    Point3f p;
    p.x= v_out[0] / v_out[3];
    p.y= v_out[1] / v_out[3];
    p.z= v_out[2] / v_out[3];

    m_objPToMicroscopeM[i].push_back(p);
  }
}
```

Table 8: Transformation chain from pattern to microscope tool

4. Estimate 4x4 camera extrinsic parameters according calculated hand-eye calibration matrix and camera intrinsic (*microscopeCameraM*).

```
// Find object pose from 3D to 2D correspondences for each image
for(int i= 0; i< m_NumberOfImages; ++i)
{
  solvePnP(m_objPToMicroscopeM[i],
           m_ImagePoints[i],
           m_CameraM,
           m_DistortionCoefficient,
           m_RotationVector,
           m_TranslationVector);

  Rodrigues(m_RotationVector, m_RotationMatrix);
}

// Find average extrinsic matrix for each solved DLT image

// Create camera matrix for left camera (imageToCamera)
m_LeftCamM->SetElement(0, 0, m_RotationMatrix.at<double>(0, 0));
m_LeftCamM->SetElement(0, 1, m_RotationMatrix.at<double>(0, 1));
m_LeftCamM->SetElement(0, 2, m_RotationMatrix.at<double>(0, 2));
m_LeftCamM->SetElement(0, 3, m_TranslationVector.at<double>(0, 0));

m_LeftCamM->SetElement(1, 0, m_RotationMatrix.at<double>(1, 0));
m_LeftCamM->SetElement(1, 1, m_RotationMatrix.at<double>(1, 1));
m_LeftCamM->SetElement(1, 2, m_RotationMatrix.at<double>(1, 2));
m_LeftCamM->SetElement(1, 3, m_TranslationVector.at<double>(0, 1));

m_LeftCamM->SetElement(2, 0, m_RotationMatrix.at<double>(2, 0));
m_LeftCamM->SetElement(2, 1, m_RotationMatrix.at<double>(2, 1));
m_LeftCamM->SetElement(2, 2, m_RotationMatrix.at<double>(2, 2));
m_LeftCamM->SetElement(2, 3, m_TranslationVector.at<double>(0, 2));

m_LeftCamM->SetElement(3, 0, 0);
m_LeftCamM->SetElement(3, 1, 0);
m_LeftCamM->SetElement(3, 2, 0);
m_LeftCamM->SetElement(3, 3, 1);
```

Table 9: Determine 3D object pose to 2D point correspondences

5. Build transformation chain with following operations (*imageToCamera*):

```
// Chain from image to the left camera
vtkMatrix4x4::Multiply4x4(m_DrfToolM,
                          m_ImageToProbeM,
                          m_ImageToDrfM);

vtkMatrix4x4::Multiply4x4(m_MicroscopeToolInverseM,
                          m_ImageToDrfM,
                          m_ImageToMicroscopeM);

 vtkMatrix4x4::Multiply4x4(m_LeftCamInverseM,
                           m_ImageToMicroscopeM,
                           m_ImageToCameraM);
vtkMatrix4x4::Invert(m_ImageToCameraM,m_ImageToCameraInverseM);
```

Table 10: Transformation chain from patient image to camera

4.3 Surgical microscope Leica M500 N

A stereoscopic binocular surgical microscope is designed to perform microscopic surgeries, in which for both eyes, a separate optical path is provided. Both eyes see the tracked surgical area therefore from a slightly different angle, so that a stereo effect impression occurs.

The important main characteristic and features of operation microscope are [42]:

M500 N	**Features**
Magnification	Electric zoom 1:6
Focus mechanism	Electric multi-focus adjustment Manual focus (OHS-1 stand)
Working distance	Electric multifocal adjustment: 207 mm to 407 mm
Total magnification	13.5 × 1.4 × (10 × eyepiece used)
Field of view diameter	16 mm to 150 mm (10 × eyepiece used)
Main light source	300W xenon lamp light source spare: 21V 150W halogen lamp
Stand	6-axis electromagnetic brake with floor stand
Main body	OHS-1
Optics head	M500 N
Video	Video port (real color) with digital camera head
Projector	Two Leica monochrome projectors (beam splitters)
Trigger control	Dual trigger controls for motorized zoom, focus,
Tilt	Head tilt release and boom support release
Control interface	LCD display to control the electrical functions
CAN-Bus	Interface for CAN-Bus connection between components of microscope e.g. optics carrier (zoom/focus), hand grips, etc.

Table 11: Overview of Leica M500 N features

The neurosurgery binocular purpose-built surgical microscope Leica M500 N from Leica Microsystems GmbH consists of following hardware components, which are labeled in Figure 38:

1. 1, 2: Baumer TXG06c industrial color cameras.
2. 3, 4: Video adapter for the first and second camera.
3. 5, 6: Left and right eyepieces.
4. 7: Lens system for zoom, focus, etc.

5. 8, 9: Monochrome video beam splitters for left and right eyepiece to inject/overlay computer generated graphics into the microscope view.
6. 10: Microscope optics (main lenses).
7. 11, 12: Left and right hand grips to handle zoom and focus operations.
8. 13: Control unit that consists of a RS-232 serial communication interface, On/Off connection for external device, foot-/hand switches and two CAN-Bus interfaces.

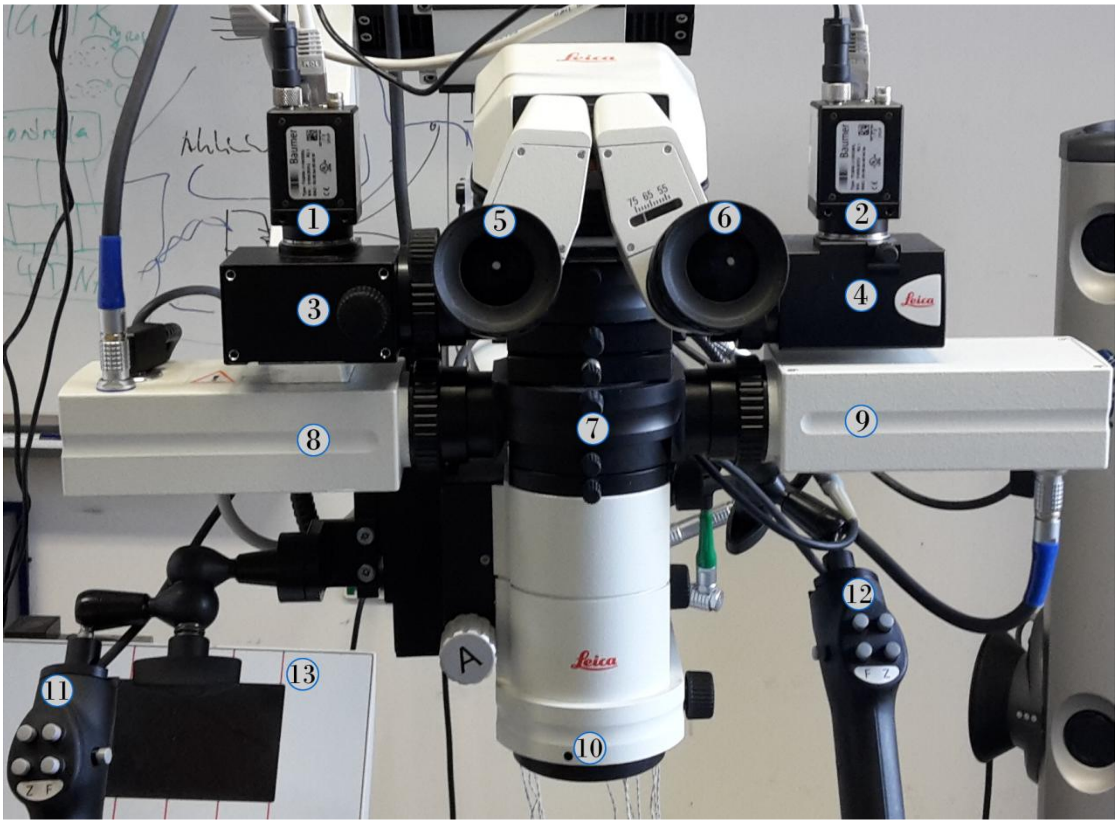


Figure 38: Components of Leica M500 N

Two Baumer TXG06c progressive scan CCD Gigabit Ethernet (GigE) stereo cameras from Baumer AG, which provides real-time video stream from microscope optic to AROSCOPE’s calibration and augmented reality modules via TCP/IP protocol; consist of following operation properties [44]:

TXG06c	**Features**
Transmitting system	GigE
Software	Baumer-GAPI: Generic software interface, Windows/Linux
Sensor	1/2“ interline progressive scan CCD

Shutter / readout mode	Global shutter / progressive scan readout
Number of pixel	776 × 578 px
Frames	Up to 64 full FPS
Scan area	6.44 mm × 4.80 mm
Pixel size	8.3 µm × 8.3 µm
Color filter	RGB Bayer mosaic
Trigger mode	Yes, overlapped operation
Free running mode	Yes, overlapped operation
Signal processing	Real-time software programmable
Pixel clock	40 MHz fast scan / 20 MHz high quality (HQ) scan
A/D converter	12 bit
Exposure control	Total: 4 µsec... 60 sec, step 1 µsec
Gain control	0.. 20 dB
Offset (black level)	0.. 255 LSB (12 bit)
Image data buffer	Max. 15 images
Optical interface/filter	C-Mount/Hoya E-CM500S

Table 12: Overview of Baumer TXG06c features

The video adapters of both cameras capture live video stream from microscope’s optic and direct the image frames to the cameras according listed functionalities in above table. Hence the workstation, which equipped with two Ethernet card, receives live video frames from surgical area and uses this video stream for purposes e.g. camera calibration and in addition for augmented reality. Two video beam splitters with 1024×768 resolutions are connected to the workstation via Digital Visual Interface (DVI) interface, which contains two Nvidia graphic cards (GeForce GTX 570 and GeForce 9600 GT), to serve live image injection into the microscope oculars. The computer generated 2D patient images or 3D models for overlaying are sent to the beam splitters and overlayed with the real patient’s surgical area according obtained projection parameters that got by camera calibration and estimation algorithm processes.

To get live video stream from both cameras, the functions in Generic Application Programming Interface (Baumer-GAPI) library [35] are used. With this API Baumer provides an interface for optimal integration, simultaneous operations and control of

Baumer GigE cameras, which offers interfaces to software implementation with e.g. C/C++ programming languages.

The *class BGAPI::System()* represents the interface by means of which it can be connected to several cameras and provides access point to other classes. Objects of *BGAPI::System()* can be enumerated, created, opened and released. The class *BGAPI::Camera()* represents the camera hardware and objects must work within an object of *BGAPI::System()* analogous to the hardware which must be connected to an interface and can be enumerated, created, opened and released. A *BGAPI::Camera()* object can add content to *BGAPI::Image()* objects, which are empty containers for content that are provided by the entry point function *BGAPI::createImage()*. These containers can be filled with image information such as width, height and pixel format, and image data provided by a *BGAPI::Camera()* object. *BGAPI::Image()* objects can be created and released and it supports image transformation. Figure 39 represents the entire system functionality of the API:

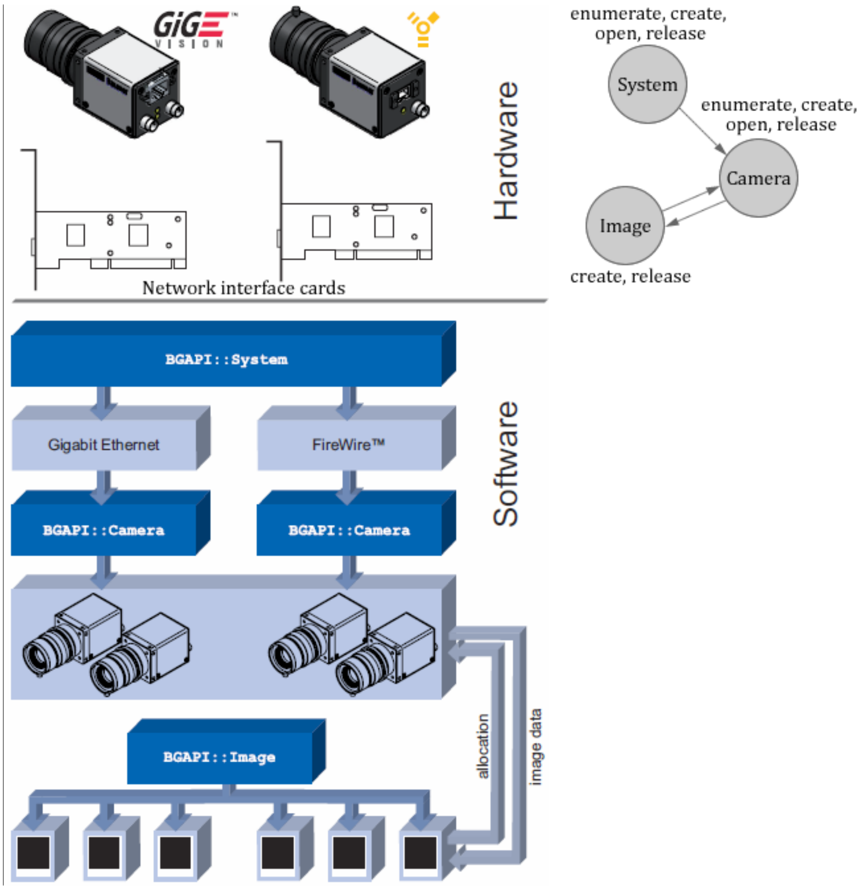


Figure 39: Baumer-GAPI complete system and object classes [35]

4.4 3D volume generation

For injecting computer generated virtual 3D model from patient’s anatomy into the microscope eyepieces, the in DICOM module loaded 2D patient images should be converted into a 3D model [45] and prepared for the overlaying. For this purpose the threshold-based 3D surface construction algorithm Marching Cube [46] with using VTK library class *vtkMarchingCubes()* [47] is implemented, which extracts a polygonal mesh of an isosurface from 3D voxels. The 3D generation module in AROSCOPE takes the processing of this task with the possibility that iteratively generated models are represented as surface, wireframe, or point cloud and even listed in augmented reality module with properties such as labeled model name and construction time, which can be selected and injected immediately in desirable eyepiece of microscope. Furthermore serves this module options such decimating the quality by generating of models to reduce the processing time of algorithm and cut function in order to extract and overlay the interested region (ROI) of anatomy in eyepieces. Figure 40 shows a segmented surface with edge visibility, which is obtained from a phantom plastic skull.

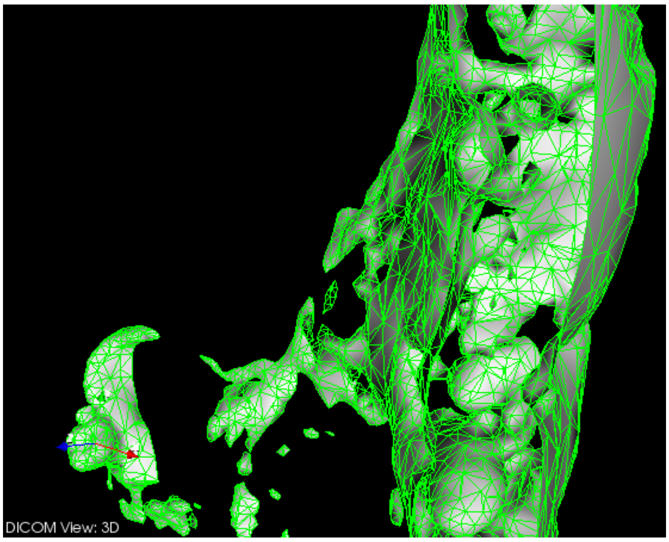


Figure 40: Generated 3D model with edge visibility

4.5 Overlaying and rendering

In order to overlay computer generated or imported and processed 3D model in the desired microscope ocular accurately, the model should be rendered and sent into the microscope beam splitters. For this merge operation the VTK classes such *vtkRenderer()* and *vtkExternalOpenGLRenderer()*; to render 3D model in a *vtkRendererWindow()* or *vtkExternalOpenGLRenderWindow()*, virtual cameras classes such *vtkCamera()* and *vtkExternalOpenGLCamera()* in order to place the overlayed object according changed microscope and patient location and regarding calculated camera calibration matrices on the FOV of microscope by applying transformation chain and perspective projection matrix in real-time and *vtkActor()*; to represent the whole process in the rendered scene, are implemented.

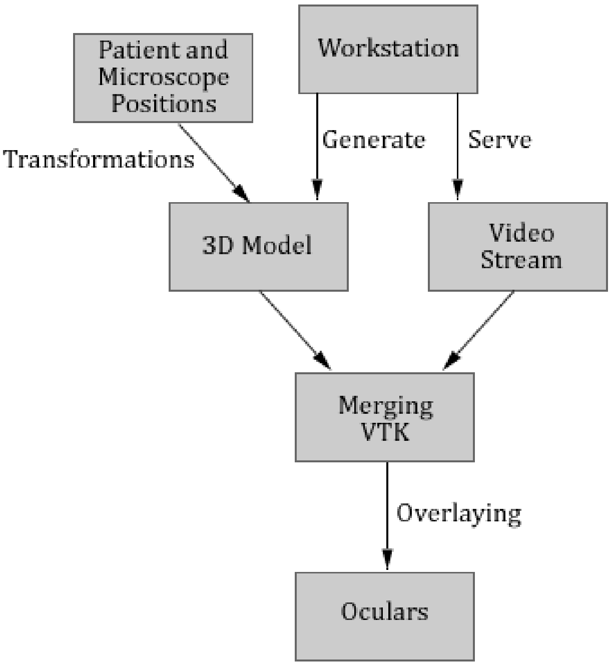


Figure 41: Data flow diagram of overlaying workflow

In background of the rendering process of vtk classes stands the OpenGL perspective projection and model view matrices [48], which are a linear transformation in homogeneous space. If a 3D scene is rendered by OpenGL, the clipping plane in a frustum should be calculated in order to have a depth-precision, and set to the virtual camera, where *right, left, far, near, top, bottom* represents the boundary values of the clipping planes.

The frustum (Figure 42 a and b) transforms all vertex data from the eye coordinates to the clip coordinates. After that clip coordinates are transformed to the normalized device coordinates (NDC) (Figure 42 c) by dividing with *w* component of the clip plane, in order to represent the projected coordinates in a normalized space between $[-1,\ +1]$. That means; it projects the world coordinates to the unit cube without having knowledge of the screen dimension and the location of the *near* and *far* planes.

To calculate the perspective projection transformation it is required 10 parameters:

- Image width and height: Physical size (chip in camera) of the camera.
- FOV: The horizontal angle of the camera through which the principal point looks at the world.
- Near Z plane: Used to clip objects that are too close to the camera.
- Far Z plane: Used to clip objects that are too distant from the camera.
- Aspect ratio: The ratio between the width and the height of the rectangular area which controls the horizontal viewing angle relative to the vertical.
- Right, left, top, bottom: Boundary parameters of the clipping plane.

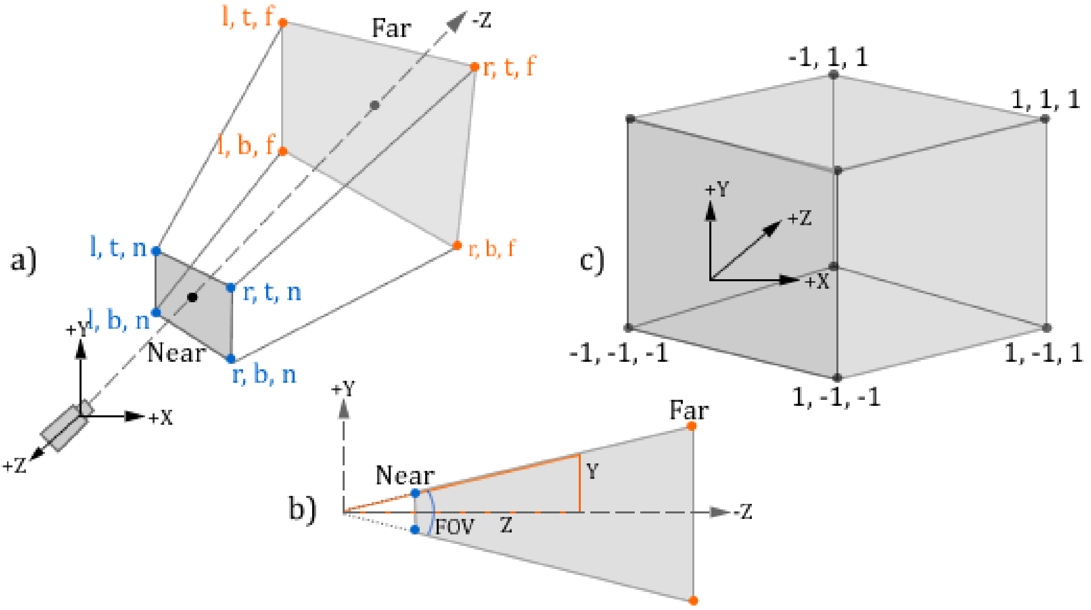


Figure 42: Perspective frustum and normalized device coordinates

The perspective projection matrix *P* is calculated:

$$P = \begin{pmatrix} \frac{2.near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2.near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2.far.near}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix}, \text{where}$$

$imageW = 776,\ imageH = 578,\ FOV = 17.235,\ near = 1,\ far = 3000,$ $aspectRatio = \frac{imageH}{imageW},\ right = near.\tan\left(\left(\frac{pi}{180}\right).\frac{FOV}{2}\right), left = -right,\ top = \frac{right}{aspectRatio},\ bottom = -top.$

The rendering process of the overlayed 3D model in the left ocular with the VTK function *vtkExternalOpenGLCamera()* looks like then:

```
// Initialize virtual camera
m_CamL= vtkSmartPointer<vtkExternalOpenGLCamera>::New();
m_CamL->SetViewAngle(m_FieldOfViewLeft);
m_CamL->SetDistance(m_FocalPointLeft);
m_CamL->UseHorizontalViewAngleOn();
m_CamL->SetParallelProjection(false);

// Visualize
m_RendererL= vtkSmartPointer<vtkExternalOpenGLRenderer>::New();
m_RenderWinL=
vtkSmartPointer<vtkExternalOpenGLRenderWindow>::New();
m_RenderWinL->AddRenderer(m_RendererL);
m_RendererL->AddActor(m_ActorL);
m_RendererL->SetActiveCamera(m_CamL);
m_RendererL->ResetCameraClippingRange();

// Create projection matrix
m_ProjectionML->SetElement(0, 0, 2 * near / (right - left));
m_ProjectionML ->SetElement(0, 1, 0);
m_ProjectionML ->SetElement(0, 2, (right + left)/(right - left));
m_ProjectionML ->SetElement(0, 3, 0);

m_ProjectionML ->SetElement(1, 0, 0);
m_ProjectionML ->SetElement(1, 1, 2 * near / (bottom - top));
m_ProjectionML ->SetElement(1, 2, (top + bottom)/(top - bottom));
m_ProjectionML ->SetElement(1, 3, 0);

m_ProjectionML ->SetElement(2, 0, 0);
m_ProjectionML ->SetElement(2, 1, 0);
m_ProjectionML ->SetElement(2, 2, -(far + near) / (far - near));
m_ProjectionML ->SetElement(2, 3, -(2*(far * near))/(far - near));

m_ProjectionML ->SetElement(3, 0, 0);
m_ProjectionML ->SetElement(3, 1, 0);
m_ProjectionML ->SetElement(3, 2, -1);
m_ProjectionML ->SetElement(3, 3, 0);

// Change the virtual camera behavior regarding updated
// transformation chain
m_CamL->SetViewTransformMatrix(m_ImageToCameraInverseM);
m_CamL->SetProjectionTransformMatrix(m_ProjectionML);
m_CamL->SetEyeAngle(m_EyeAngleValue);
m_CamL->SetEyeSeparation(m_EyeAngleValue);

m_RendererL->ResetCameraClippingRange();
```

Table 13: Overlaying and rendering with VTK, option 1

And the rendering process of the overlayed 3D model in the left ocular with the VTK function *vtkCamera()*:

```
// Initialize virtual camera
m_CamL= vtkSmartPointer<vtkCamera>::New();

// Visualize
m_RendererL= vtkSmartPointer<vtkRenderer>::New();
m_RenderWinL= vtkSmartPointer<vtkRenderWindow>::New();
m_RenderWinL->AddRenderer(m_RendererL);
m_RendererL->AddActor(m_ActorL);
m_RendererL->SetActiveCamera(m_CamL);

// Change the virtual camera behavior regarding updated
// transformation chain
m_CamL->SetPosition(m_ExtrinsicX, m_ExtrinsicY, m_ExtrinsicZ);
m_CamL->SetViewUp(m_CamViewUpLX, m_CamViewUpLY, m_CamViewUpLZ);
m_CamL->SetDistance(m_CamFocalPointL);
m_CamL->SetViewAngle(m_FieldOfViewL);
m_CamL->SetEyeAngle(m_EyeAngleValue);
m_CamL->UseHorizontalViewAngleOn();
m_CamL->SetParallelProjection(false);

m_RendererL ->ResetCameraClippingRange();
```

Table 14: Overlaying and rendering with VTK, option 2

4.6 Validation

To validate and analyze the correctness of the built transformation chain (described in 2.2.8 and 4.2), a MATLAB script is implemented, which overlays estimated 2D image points and calculated perspective projection matrix on calibration pattern for each taken calibration pattern image respectively. For performing the validation, two different checkerboard patterns are used (First: 8x5 squares with 10.0 mm edge length with 15 images. Second: 9x6 squares, with 5.0 mm edge length, with 20 images). At the same time for each taken image the poses from DRF-, microscope- and checkerboard-tool are saved individually.

First the intrinsic and the hand-eye calibration is calculated. Therefore the object points are transformed in the microscope tool coordinates. Secondly the extrinsic parameters are estimated respectively. Thirdly the transformation chain from calibration image to the camera is built and finally estimated 2D image points using DLT and calculated perspective projection matrix are visualized on the calibrated checkerboard image.

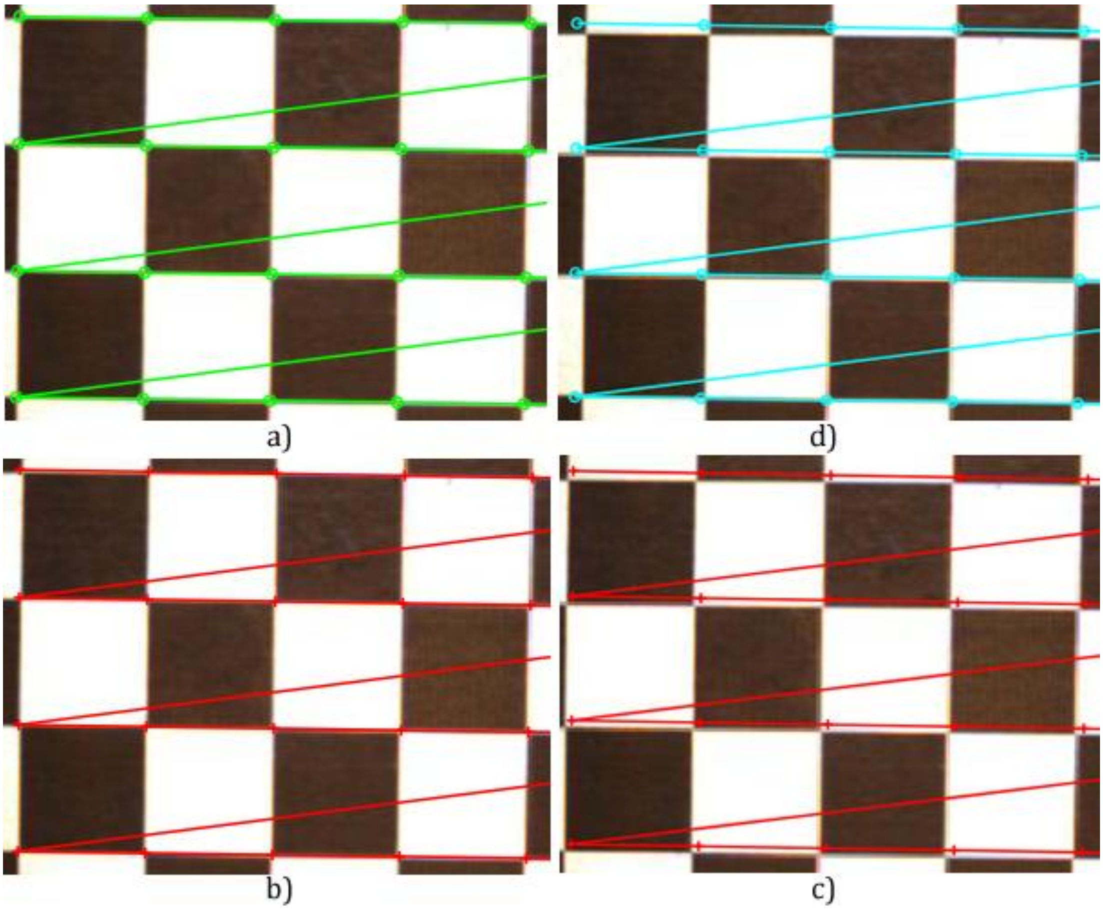


Figure 43: Validation of AR transformation chain using first checkerboard pattern

The green lines with “o” symbol on image a) illustrates the detected image points for the first image after camera calibration process (described in 3.4) and red lines with “+” symbol on image b) shows the estimated image points using DLT for the first calibrated image. The image c) shows the averaged image points estimation using DLT from all 15 calibrated images on the first image, which covers camera calibration-, tracking- and hand-eye calibration-errors. Finally the cyan lines on image d) shows overlayed image points of averaged estimation using DLT from all 15 calibrated images using applied perspective projection matrix on the first image.

The same validation task is applied with the same conditions for the second checkerboard pattern images (Figure 44). The results show: The projection or overlaying error rates are almost the same for the cases a), b), c) and d). The deviations by overlaying with averaged error parameters are not greater than 2 mm on all images. There is no difference in the overlaying deviation of two checkerboard pattern with respect to square size, edge length and number of calibrated images.

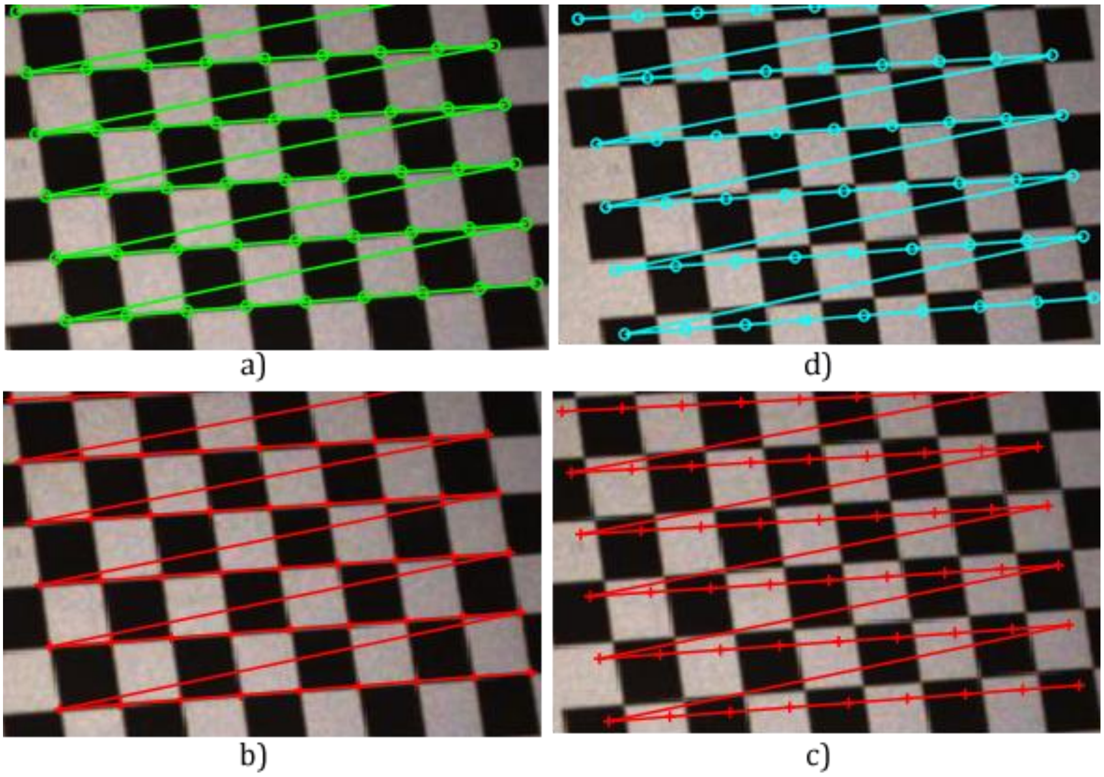
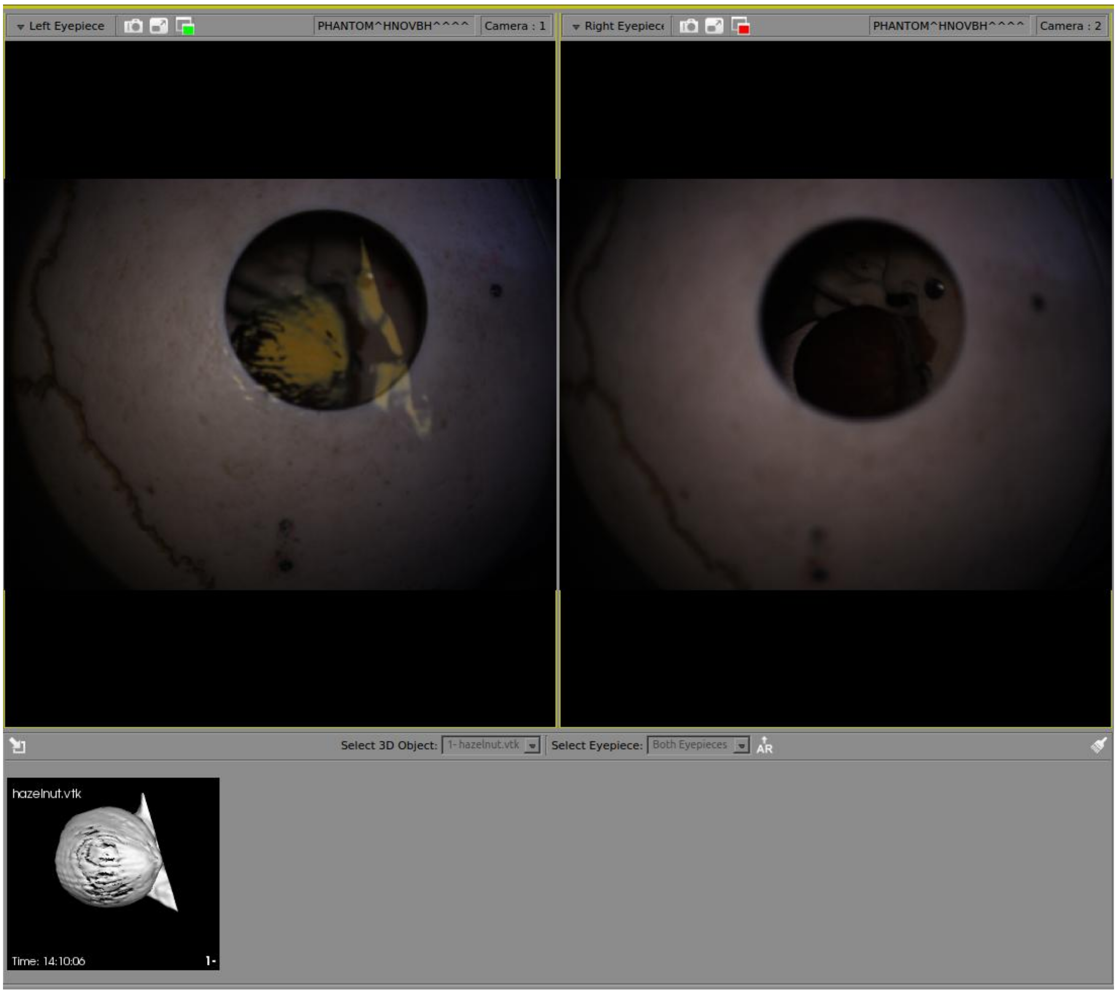


Figure 44: Validation of AR transformation chain using second checkerboard pattern

4.7 Visualization

Performing the necessarily operations for stereo overlaying in AROSCOPE takes place under the augmented reality module. This module interacts with the left and right projectors, which are adapted to the microscope and connected to the workstation's graphic cards via DVI interface. The processed 3D models are injected into the oculars and rendered by using the essentially VTK classes and updated regarding the changed transformation chain, in order determining and applying new pose for the overlayed model in used virtual camera immediately. Furthermore offers this module importing and processing of models (clipping, coloring, kind of different representations, etc.), that processed in current session or saved in previous sessions.

Figure 45, which is a screenshot from augmented reality module (described in 5.7.5), illustrates in bottom scene an imported 3D model from a hazelnut, which was placed in the tracked phantom skull and symbolizes a tumor in brain. After segmentation and clipping operations it was saved in previous session with labeled name, number and generating time of the model. At the main scene it is then illustrated the overlaying of imported virtual model on the real patient in the left ocular during current session.



Select 3D Object: 1-hazelnut.vtk

Select Eyepiece: Both Eyepieces

Figure 45: Visualization of an overlaying in a phantom patient

The applied virtual camera class *vtkCamera()*, which generates perspective projection and model view matrices internally regarding given parameters (described in 4.5), cause a deviation of overlayed model by rotating of patient or microscope head, which is not a case for the translations. The *vtkExternalOpenGLCamera()*, which is a concrete implementation of the abstract class *vtkCamera()* interfaces directly to the OpenGL rendering library and it is released in newest VTK version 7.0.0. It handles given perspective projection and model view matrices, which is created by the developer externally, does not visualize the object on the scene. To use the functionalities of *vtkExternalOpenGLCamera()* in the currently used VTK version 5.10.1 by AROSCOPE, the essential classes from version 7.0.0 are integrated in the version 5.10.1. Since this newest version of VTK is currently not compatible with the used IGSTK library version 5.0. A possible problem can be here maybe the compatibilities of the integrated classes. Hence this trouble is still open and will be handled in the next version (described in 5.9) of AROSCOPE as long as the newest VTK library is supported by the IGSTK library.

4.8 Results

In this chapter is given a short overview of augmented reality with a surgical binocular microscope in medical use. The estimation of 2D points from 3D world coordinates according pose estimation algorithm is described and used methods for communicating with the surgical microscope and individual components are explained, in order to serve the real-time video stream. The rendering of 3D models, which are obtained from CT images and the created valid transformation chain for the overlaying, gives acceptable and suitable results for a proof of concept. The results show: there is no existence of an appreciable lag during the video streaming or refreshing/relocation the overlayed object pose after any movement of patient or microscope head during the intervention.

With reference to results of validation in patient-image registration (explained in 2.4), of validation in camera calibration and pose estimation (explained in 3.7 and 4.6) the overall error rate by the overlaying is not greater than 3 mm, including all possible error sources such camera calibration, patient registration, hand-eye calibration, pose estimation and tracker measurements.

CHAPTER 5 - AROSCOPE

5.1 Introduction

AROSCOPE (version 1.0) is a proof of concept (POC), which realizes the certain methods with their feasibility to fulfill all needed requirements for minimal invasive interventions with navigated surgical microscope in ENT or neurosurgery operations. AROSCOPE supports 3D intraoperative patient-, microscope navigation and augmented reality beside patient image viewing, camera calibration, 3D segmentation, and image-patient registration possibilities.

AROSCOPE is platform independent open-source software, which currently runs on Linux OS. It is developed and tested under standard 64-bit Linux PC (distributions: Debian 8 and Ubuntu 14.04 LTS) with programming language C++ using singleton pattern and other cross-platform open-source standard libraries such as IGSTK, ITK, VTK, OpenCV, Open Graphics Library (OpenGL), GUI Development Framework (Qt) and Build System, (CMake) with extensive functionalities. The developing process is realized with Eclipse Indigo using C/C++ integrated development environment (IDE).

This section gives an overview of main components based on structural hardware- and software design and developing stages of AROSCOPE. Described are the used frameworks and open-source libraries as well as implemented modules in the context of this thesis.

Figure 46: AROSCOPE

5.2 System architecture

The concept includes five modules that work dependent with each other. Preoperative radiological patient images such as CT are loaded into DICOM module using IGSTK and visualized accordingly. Image data can be represented in orthogonal views; axial, sagittal, coronal, or multiplanar and simple image processing functions such as windowing, distance measurement, reslicing, orientation, segmentation and 3D visualization, etc. are available.

The navigation module offers standard navigation of the patient with the landmark based registration, in which the focus of the microscope, the patient and an instrument is navigated relative to a DRF over the optical tracker optotrak. Probes are calibrated usually with pivot calibration technique. To transform the 3D coordinate system of the microscope in the coordinates of the 2D image plane is the projection matrix of both industrial cameras calculated for each binocular view. At the same time the real-time navigation of the patient and the microscope head is effected, in which the transformations are produced with respect to the tracker.

In calibration module both cameras are calibrated together with the associated stereo channel of the microscope individually. Thereby the whole parameter space of the zoom and focus positions must be covered as possible.

Anatomical structures are segmented from the CT data in 3D generation module with threshold based marching cubes algorithm and meshes are generated, which are offered as a virtual 3D objects to overlay with the real patient scene. These reconstructions are represented on two image injection interface in the optical beam path of the stereomicroscope over two LED panels in augmented reality module.

5.3 Hardware setup

The hardware setup of entire system looks like as follows:

1. 1: Connecting the optotrak to the workstation via USB interface.
2. 2, 3: Connecting the first and second camera to the workstation via LAN interface.
3. 4, 5: Connecting the first and second beam splitter (projector) to the workstation via DVI interface.

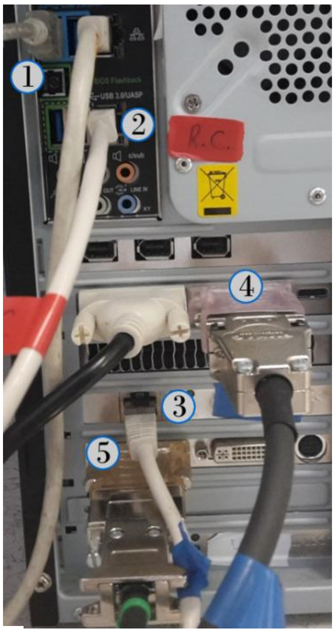


Figure 47: AROSCOPE's hardware setup

To include both cameras into the workstation, the network connection interface should be configured properly with the IP addresses of cameras. In order to adapt both beam splitters to workstation the configuration of X.Org Server should be manipulated. The X.Org Server configuration to handle the monitor on workstation and two beam splitters:

```
Section "ServerLayout"
Identifier     "X.org Configured"
Screen      0  "Screen0" 0 0
Screen      1  "Screen1" RightOf "Screen0"
```

```
Section "Monitor"
  Identifier   "Monitor0"
  VendorName   "Monitor Vendor"
  ModelName    "Monitor Model"
EndSection
Section "Device"
  Identifier  "Card0"
  Driver      "nouveau"
  BusID       "PCI:1:0:0"
EndSection
Section "Screen"
  Identifier "Screen0"
  Device     "Card0"
  Monitor    "Monitor0"
EndSection
```

```
Section "Monitor"
  Identifier   "Monitor1"
  VendorName   "Monitor Vendor"
  ModelName    "Monitor Model"
EndSection
Section "Device"
  Identifier  "Card1"
  Driver      "nouveau"
  BusID       "PCI:2:0:0"
EndSection
Section "Screen"
  Identifier "Screen1"
  Device     "Card1"
  Monitor    "Monitor1"
EndSection
```

Table 15: X.Org Server configuration

5.4 Software architecture

AROSCOPE software architecture is built using singleton pattern mechanism in C++ [49]. That means, in every classes of each module exactly one instance is available for the use in other classes. To create a single instance in a class and access to the implemented methods in that class from other classes:

Create

```
// Create and declare instance
static className *instance();
static className *m_className;

// Create method to
className
*className::instance()
{
 if(m_className== NULL)
  m_className= new className
();
  return m_className;
}
```

Access

```
//Access over SIGNAL SLOT mechanism
connect(m_GUI.button,
SIGNAL(clicked()),className::instan
ce(), SLOT(methodInClassName()));

// Access directly
className::instance()-
>methodName("Argument");
```

Table 16: Singleton pattern mechanism

The customizable modular graphical user interface (GUI) in AROSCOPE is implemented using cross-platform application framework Qt (version 4.8) with included essential Qt Multimedia and created stylesheet file for each Qt class elements to create an interface between user, modules and used libraries. The ITK library (version 3.20.1) is used to get DICOM-Tags attributes such as modality, patient name, study date, manufacturer, institution name, image orientation, etc. from a loaded image data set and to calculate matrix operations in navigation- and calibration modules. VTK library (version 5.10.1) takes the operations to generate 3D models and their visualizations using user-defined Qt class *QVTKWidget()*. The IGSTK library (version 5.0) is used for navigation and visualization of radiological image purposes using user-defined *igstk::QTWidget()* class in Qt. The library OpenCV (version 2.4.10) is applied for camera calibration and AR processes. The build system CMake (version 3.0.2) is applied to managing and control the build process of entire software elements with other used libraries using a *CMakeList* file. The API's for the tracker optotrak (liboapi) and Baumer cameras (libbgapi, version 1.7) are used to access and manipulate libraries for desirable operations.

To manage the whole code better by implementing and facilitate the revision control, each important process is separated from each other. E.g. for the visualization operations it is exist *visualizationManager*, for tracking processes *trackingManager*, to handle the microscope *microscopeManager*, for camera calibration *calibrationManager*, to handle the logging messages *loggingManager*, for loading of patient images to the scene *openFileManager*, for managing of setting operations *settingsManager*, etc. The visualization of errors, warnings, and information management occurs by the use of popup message boxes that appear after an undertaken operation with the outcome result. Figure 48 represents the entire software architecture and used components of AROSCOPE.

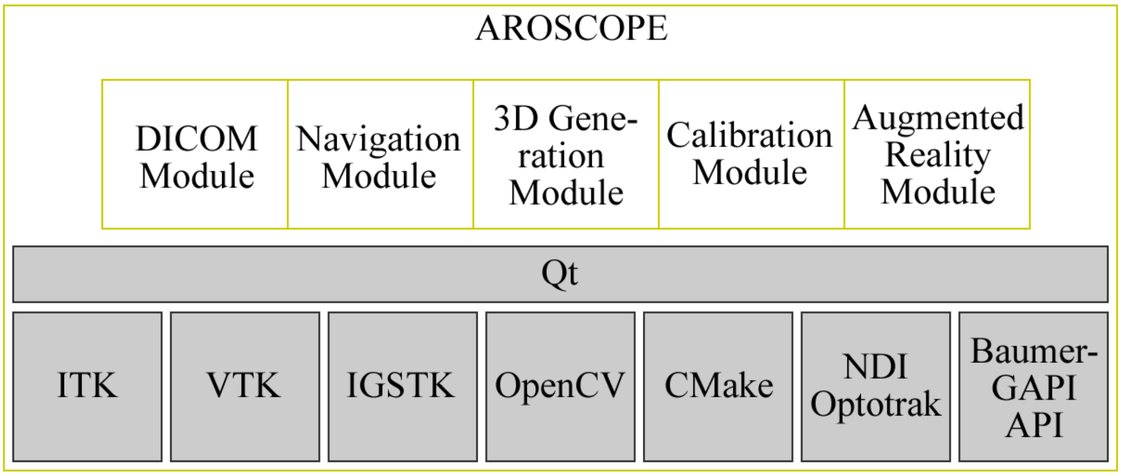


Figure 48: AROSCOPE's software architecture

5.5 Installation and configuration

After installation of all required libraries with mentioned versions, it should be created a *CMakeList.txt* file that manages the build process of software [50]. A basic *CMakeList* file example for linking e.g. Qt and VTK libraries looks like as follows to generate native *Makefile* and workspaces that can be used in the compiler environment:

```
PROJECT(aroscope)
cmake_minimum_required (VERSION 2.6)

#Add and use Qt library
FIND_PACKAGE(Qt4 REQUIRED)
IF (QT_USE_FILE)
 INCLUDE (${QT_USE_FILE})
ELSE (QT_USE_FILE)
```

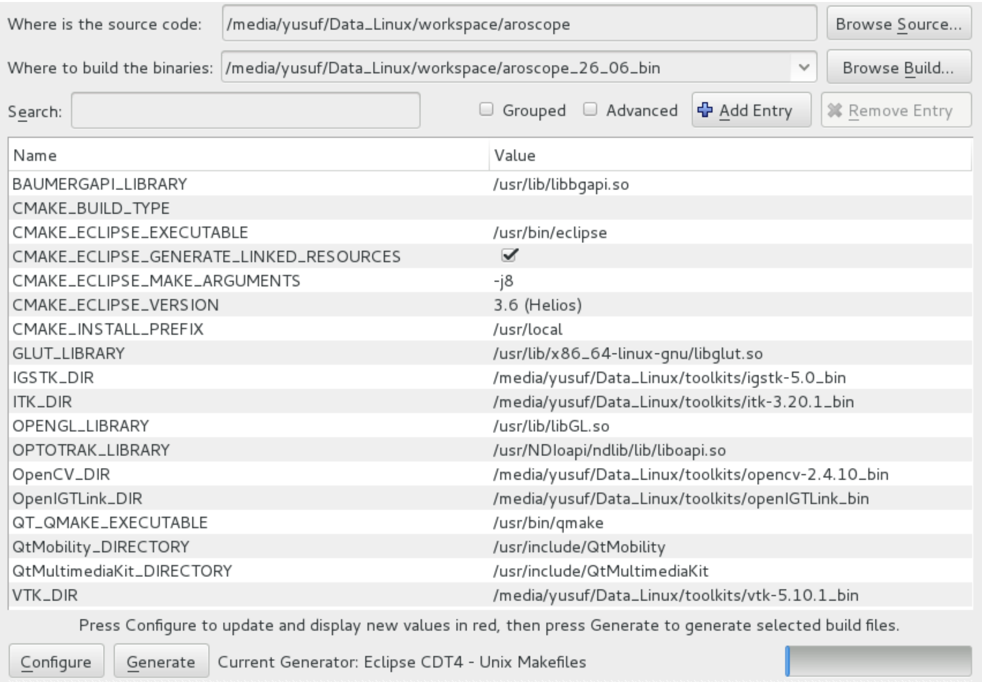
```
 MESSAGE (FATAL_ERROR "ERROR: This application requires Qt. This component is missing. Please verify configuration!")
ENDIF (QT_USE_FILE)
# Include directories, in which located following files
INCLUDE_DIRECTORIES(
  ${aroscope_SOURCE_DIR}
  ${aroscope_BINARY_DIR}
  ${aroscope_SOURCE_DIR}/source/trackerManager …)

# Add and use VTK Library
FIND_PACKAGE(VTK)
IF(VTK_USE_FILE)
      INCLUDE(${VTK_USE_FILE})
ELSE (VTK_USE_FILE)
      MESSAGE (FATAL_ERROR "ERROR: This application requires VTK.
This component is missing. Please verify configuration!")
ENDIF (VTK_USE_FILE)

# To add other libraries same procedure is followed

# Include Qt user interface files
SET(aroscope_GUI_SRCS
      aroscopeMainWindowGUI.ui …)

# Generate rules for building source files from the resources
QT4_ADD_RESOURCES(aroscope_RCC_SRCS ${aroscope_RCCS})
QT4_WRAP_UI(aroscope_QT_UI_SRCS ${aroscope_GUI_SRCS})

# Include source files of project
SET(aroscope_SRCS
  main.cxx
  ${aroscope_SOURCE_DIR}source/trackerManager/trackerManager.cxx …)

#Include header files of project
SET(aroscope_HDRS
  ${aroscope_SOURCE_DIR}source/trackerManager/trackerManager.h …)

QT4_AUTOMOC(${aroscope_SRCS} ${aroscope_HDRS})

# Add executables to project using given files
ADD_EXECUTABLE(aroscope ${aroscope_QT_UI_SRCS} ${aroscope_RCC_SRCS}
${aroscope_SRCS} ${aroscope_HDRS})

# Link project to given libraries
TARGET_LINK_LIBRARIES(aroscope ${QT_LIBRARIES} ${VTK_LIBRARIES} …)
```

Table 17: CMake file example

After creating the file, the project should be build using CMake GUI application (Figure 49). The top two entries are the source code and binary directories. They allow the user to specify where the source code is located that should be compiled and where the resulting binaries should be written. After inserting both directory paths, the binary directories of installed libraries should be set respectively. To generate binaries for the project it is used in this case the Eclipse CDT4 - Unix Makefiles generator that allows creating the project files for Eclipse platform. Then the appropriate *make* program can build the project through the default *make* target. So afterwards the user can start the application with command *./aroscope* from a terminal.

5.6 Logging

After starting a session, AROSCOPE creates a log directory named *logFiles* and a file in this directory with the patient name and session time under defined path in settings window. The log file contains list of debug information to find of any undertaken action during the current session. If a 3D model from patient images is generated, the model is saved in this directory in poly data file format to import it in other sessions for AR operations. At the same time it creates in the same path two directories for camera calibration and hand-eye calibration processes named *cameraCalibrationFiles* and for user-defined settings parameters named *configurations.* In the first directory are located the captured images of both cameras and result files from this process. In the second directory the configuration parameters are saved. Hence the logged/saved outcomes from current session can be imported and used for the purpose in other sessions.


Where is the source code: /media/yusuf/Data_Linux/workspace/aroscope

Where to build the binaries: /media/yusuf/Data_Linux/workspace/aroscope_26_06_bin

Search:

☐ Grouped ☐ Advanced

Name	Value
BAUMERGAPI_LIBRARY	/usr/lib/libbgapi.so
CMAKE_BUILD_TYPE	
CMAKE_ECLIPSE_EXECUTABLE	/usr/bin/eclipse
CMAKE_ECLIPSE_GENERATE_LINKED_RESOURCES	☑
CMAKE_ECLIPSE_MAKE_ARGUMENTS	-j8
CMAKE_ECLIPSE_VERSION	3.6 (Helios)
CMAKE_INSTALL_PREFIX	/usr/local
GLUT_LIBRARY	/usr/lib/x86_64-linux-gnu/libglut.so
IGSTK_DIR	/media/yusuf/Data_Linux/toolkits/igstk-5.0_bin
ITK_DIR	/media/yusuf/Data_Linux/toolkits/itk-3.20.1_bin
OPENGL_LIBRARY	/usr/lib/libGL.so
OPTOTRAK_LIBRARY	/usr/NDIoapi/ndlib/lib/liboapi.so
OpenCV_DIR	/media/yusuf/Data_Linux/toolkits/opencv-2.4.10_bin
OpenIGTLink_DIR	/media/yusuf/Data_Linux/toolkits/openIGTLink_bin
QT_QMAKE_EXECUTABLE	/usr/bin/qmake
QtMobility_DIRECTORY	/usr/include/QtMobility
QtMultimediaKit_DIRECTORY	/usr/include/QtMultimediaKit
VTK_DIR	/media/yusuf/Data_Linux/toolkits/vtk-5.10.1_bin

Press Configure to update and display new values in red, then press Generate to generate selected build files.

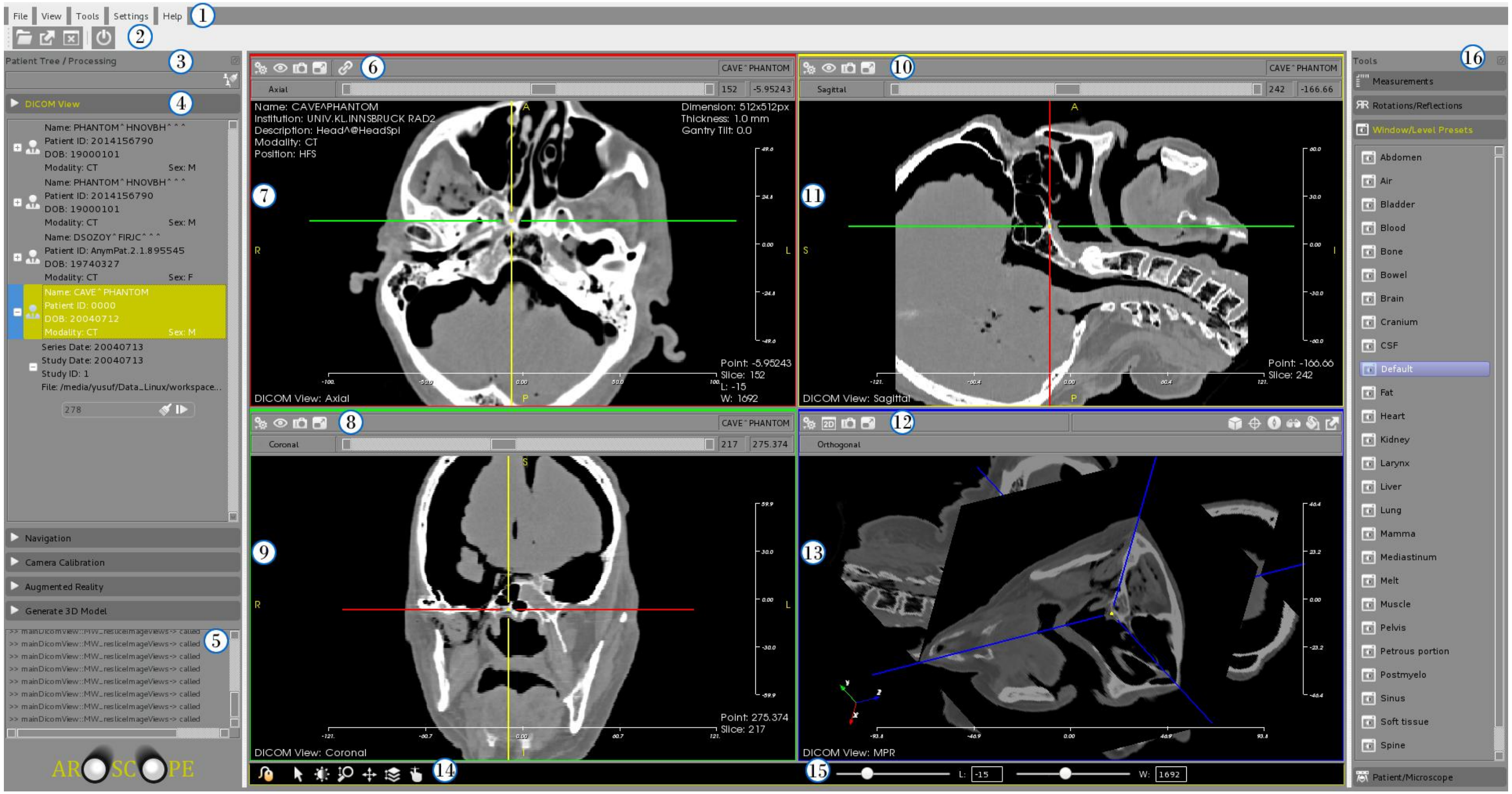
Current Generator: Eclipse CDT4 - Unix Makefiles

Figure 49: CMake configuration

5.7 5 Modules

In the following pages are demonstrated and described the modules that build AROSCOPE’s structural and functional design in detail.

5.7.1 DICOM module



Name: PHANTOM^HNOVBH^^^
Patient ID: 2014156790
DOB: 19000101
Modality: CT Sex: M

Name: PHANTOM^HNOVBH^^^
Patient ID: 2014156790
DOB: 19000101
Modality: CT Sex: M

Name: DSOZOY^FIRJC^^^
Patient ID: AnymPat.2.1.895545
DOB: 19740327
Modality: CT Sex: F

Name: CAVE^PHANTOM
Patient ID: 0000
DOB: 20040712
Modality: CT Sex: M

Series Date: 20040713
Study Date: 20040713
Study ID: 1
File: /media/yusuf/Data_Linux/workspace...

278

```
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
```

Name: CAVE^PHANTOM
Institution: UNIV.KL.INNSBRUCK RAD2
Description: Head^@HeadSpi
Modality: CT
Position: HFS

Dimension: 512x512px
Thickness: 1.0 mm
Gantry Tilt: 0.0

Point: -5.95243
Slice: 152
L: -15
W: 1692

Point: -166.66
Slice: 242

Point: 275.374
Slice: 217

Figure 50: Overview of DICOM module

Table 18 lists and describes the functionalities of all elements on main application window and especially on DICOM module, which are number on above figure:

Area	Icon / Text	Functionality
1	**Menu bar**	Horizontal menu bar, which consists of a list of pull-down menu items such File, View, Tools, Settings, and Help. Many operations in DICOM module or other modules that listed on each menu and sub-menus can be processed either trough menu items or buttons on modules.
2	**Main toolbar**	A modular toolbar that can be placed on top, bottom, left, or right sides of main window, and contains following buttons on it.
-	📂	Offers the possibility to open the window that loads patient images to the patient tree and main scene.
-	↗ *	Export Wizard.
-	☒	Clears the main scene so represented orthogonal views.
-	⏻	Ends the application.
3	**Processing Sidebar**	A modular sidebar (left frame) that can be placed on left or right sides of main window. It contains a total of five pages. For each suitable module a special page is available, in which are located adapted functionalities. If the module pages are changed, the main window jumps to the view (main frame) for the selected module. At begin the DICOM module is displayed as default in both frames.
-	✕	Clears the patient tree entries, so loaded patient image data set from the tree.
4	**DICOM View**	Represents the loaded patient image data set to load to the scene or remove it from the tree. It contains many information about the patient such patient- name, sex, birthday, image modality, number of loaded images, etc. It can be loaded max. 25 patient image data set in a session.
-	👤	Head of patient image pages. A loaded data set in tree consists of three pages. In each page different information about image is displayed.
-	🖌	Clears the represented image dataset from main scene.

-	▮▶	Loads the current data set in main scene.
5	**Logging**	The application contains a logging window, in which each user operation is displayed in rich text format to get feedback of undertaken user operations.
6	**Axial Toolbar**	Serves image processing functionalities to handle certain operations for the axial view.
-	⚙ ↶ T 0 1 2 ⌖ 🖌	Opens a menu for axial view and provides below functionalities: Resetting view to the default. Show/Hide orientation annotations. Show/Hide Dicom tag annotaitons. Show/Hide scale information. Show/Hide cross hair. Change lookup color of view.
-	👁	Show/Hide current axial view in MPR view representation.
-	📷	Takes a snapshot from the current axial view.
-	⤢ ⤡	Expands and minimizes the current axial view.
-	🔗	Link/Unlink function offers possibility to apply an operation, which is undertaken in axial toolbar, in other views at the same time.
-	Slider	Reslice the axial images. The slider range lies between 0 and max slice number of images.
-	Change View *	Changes orthogonal view from axial to other kind of representations.
-	Image Slice Nr.	Indicates the current slice number for represented axial view.
-	Image Point Nr.	Indicates the current image point number for represented axial view.
7	**Axial View**	Represents the patient images as axial.
-	Mouse Interactions	Left button: (Click) Picking on an image. (Hold down and move): Scroll image or turn 3D model. Middle button: (Scroll wheel) Zoom in/out an image or 3D model. (Hold down and move) Drag image or 3D model in window.

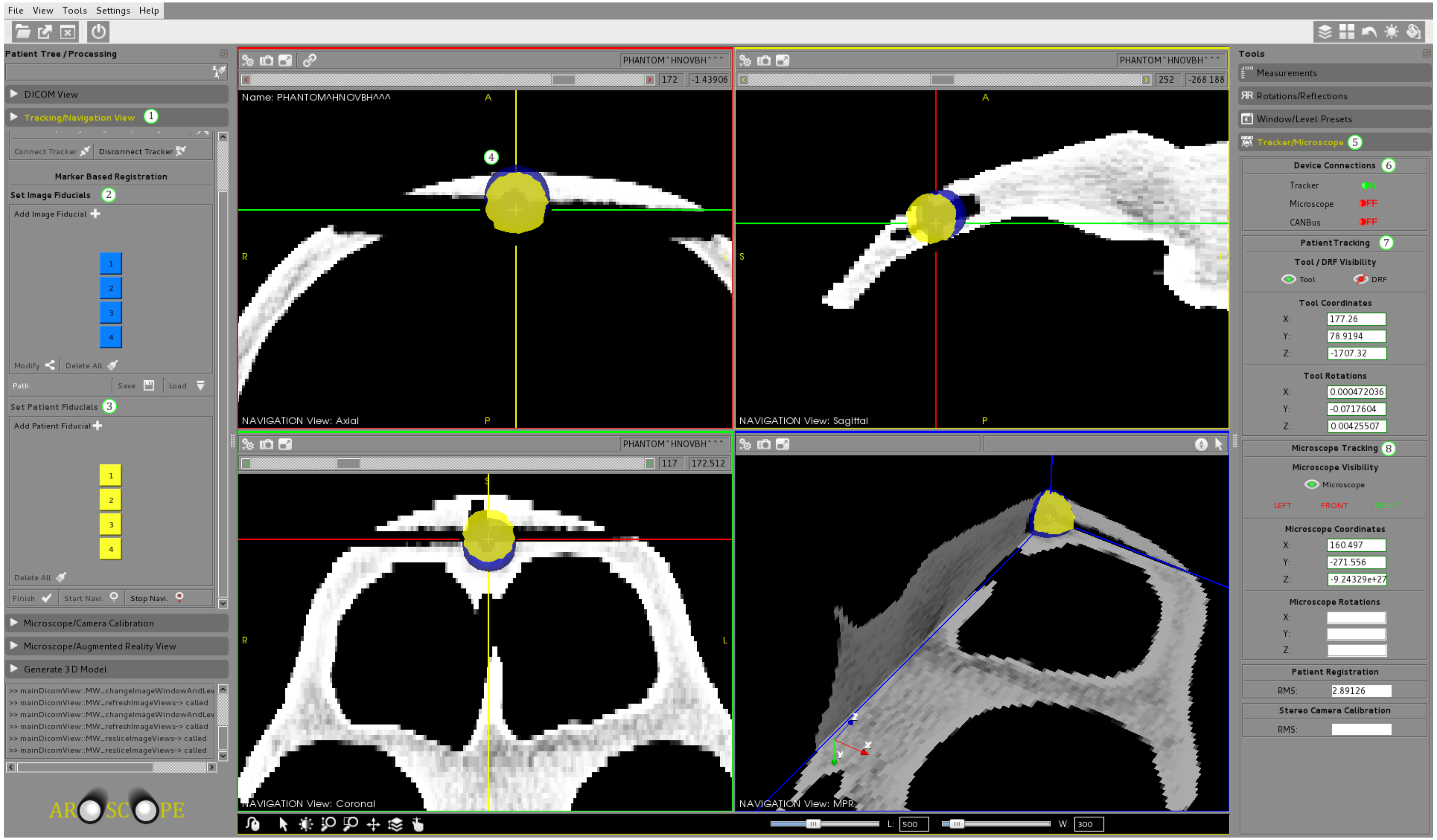
		Right button: (Hold down and move) Zoom in/out an image or 3D model.
-	Text Annotations	Represents information about loaded image data set such as: Patient name, image dimensions, thickness, gantry tilt, modality, etc.
-	2D Orientations	Gives feedback about the head orientation reference for each view. A: Anterior, I: Inferior, L: Left, R: Right, P: Posterior, and S: Superior.
-	Cross Hair	Indicates a position in each view. E.g. when it is clicked on a place in axial view, the sagittal and coronal views jump to the same slice and so show the same image in suitable orientations.
-	Distance Scale	Performs an indication of the scale of the scene for each view.
8	**Coronal Toolbar**	Serves image processing functionalities to handle specific operations for the coronal view. The available buttons have same effect like in the axial toolbar.
9	**Coronal View**	Represents the patient images as coronal. It has same cross hair and distance scale functionalities like in axial view.
10	**Sagittal Toolbar**	Serves image processing functionalities like in coronal toolbar for the sagittal view. The available buttons have same effect like in the axial toolbar.
11	**Sagittal View**	Represents the patient images as sagittal. It has same cross hair and distance scale functionalities like in axial view.
12	**MPR/3D Toolbar**	Serves image processing functionalities to handle certain operations for the multiplanar or 3D view.
-	2D 3D	Turns view from MPR to 3D, if any 3D model is created and vice versa.
-	⬡	Defines a region of interest for created 3D model, which is represented by an arbitrarily oriented hexahedron with interior face angles of 90 degrees (orthogonal faces). The box creates 7 handles that can be moused on and manipulated. It provides a cut option.
-	⌖	Show/Hide cross hair on MPR view.

-	🧭 *	Enables compass for 2D orientation.
-	👓 *	Enables stereo view for 3D model.
-	🎨	Changes the color of created 3D model.
-	↗ *	
13	**MPR/3D View**	Displays the patient images as multiplanar (axial, sagittal, coronal) representation or 3D model.
-	3D Orientations	Performs orientation support by MPR or 3D viewing. Available is either an axes or segmented 3D model representations.
14	Mouse Modes *	Lock/Unlock multiple mouse modes. If this modus is activated the image manipulation operations (picking, drag, zoom, reslicing, etc.) are performed only with left mouse button by clicking appropriate button.
15	**Windowing**	Serves the possibility to change the grey value of images according Hounsfield-Scale. The changes are applied in DICOM- and navigation modules.
-	L and W	L defines level value using a slider between min -1000 and max 3000. W defines width value using a slider between 0 and (max level value - min level value).
16	**Tools Sidebar**	A modular sidebar (right frame) that can be placed on left or right sides of main window. It contains a total of four pages. The first three pages serve image manipulation options for DICOM module and the last page gives feedbacks from outcomes of navigation- and calibration modules.
-	📏 *	Contains measurement and drawing options on currently shown images.
-	ЯR *	Lists rotation and reflection options for currently shown images.
-	▣	Contains defined grey value presets for the represented images. At begin of any session or changing to the Dicom View from left frame, this page is shown as default.

Table 18: Functionalities in main window and DICOM module

*: It has no functionality in the current version.

5.7.2 Navigation module



Name: PHANTOM^HNOVBH^^^

Axial: 172 -1.43906

Sagittal: 252 -268.188

Coronal: 117 172.512

L: 500 W: 300

Device Connections

Tracker ON
Microscope OFF
CANBus OFF

Tool Coordinates

X: 177.26
Y: 78.9194
Z: -1707.32

Tool Rotations

X: 0.000472036
Y: -0.0717604
Z: 0.00425507

Microscope Coordinates

X: 160.497
Y: -271.556
Z: -9.24329e+27

Microscope Rotations

X:
Y:
Z:

Patient Registration

RMS: 2.89126

Stereo Camera Calibration

RMS:

```
>> mainDicomView::MW_changeImageWindowAndLev
>> mainDicomView::MW_refreshImageViews-> called
>> mainDicomView::MW_changeImageWindowAndLev
>> mainDicomView::MW_refreshImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_resliceImageViews-> called
```

Figure 51: Overview of navigation module

Table 19 lists and describes the functionalities of certain elements, which are labeled on figure of navigation module:

Area	Icon / Text	Functionality
1	**Navigation**	Offers interaction with tracker and marker based registration method using image- and patient fiducials.
-	Combo box	Selects the tracker, with which the user will work. This version interacts only with optotrak tracker.
-	🛠	Opens settings window, to reconfigure settings for the selected tracker.
-	🔌 🔌	Connects/Disconnects tracker with/from the workstation.
2	**Registration**	Serves marker based registration method to realize patient navigation.
-	✚	Adds new image or patient fiducials on patient images with using left mouse button or on patient skin with using probe tool.
-	Image Fiducials ⬜ 🟦	The field illustrates set and consecutively numbered image fiducials by user. Max. 25 fiducials is allowed. Symbolizes empty image fiducial on the image fiducials field. Symbolizes added image fiducial on the image fiducials field.
-	<	Modifies the position of previously set and selected image fiducial on the view.
-	🧹	Removes all defined image fiducials.
-	💾	Saves set image fiducials in a selected directory to import them in other sessions.
-	▼	Loads saved image fiducials for the current session.
-	Patient Fiducials ⬜ 🟨	The field represents set and consecutively numbered patient fiducials by user with the tracked probe tool. Max. 25 fiducials is allowed. Symbolizes empty patient fiducial on the patient fiducials field. Symbolizes added patient fiducial on the patient fiducials field.
-	🧹	Removes all defined tracker fiducials.

-	💾	Saves set tracker fiducials in a selected directory to import them in other sessions.
-	▼	Loads saved tracker fiducials for the current session.
-	✓	Registers image- and patient fiducials with each other.
-	📍 📍	Start/Stop patient navigation.
3	**View Toolbars**	Has same functionality like in DICOM module.
4	**Views**	Has same functionality like in DICOM module.
	🔵	Symbolizes defined image fiducial on view.
	🟡	Symbolizes defined patient fiducial on view.
-	Mouse Interactions	Have same interaction possibilities like in DICOM module.
5	**Tools Sidebar**	
-		Displays feedbacks about device connection state, tool visibility during navigation, coordinates of tracked tools, and shows outcome errors from patient-image registration, camera calibration and image projection.
	OFF	The devices (Tracker or Microscope) are not started.
	ON	The devices run.
	👁	The tool is not in FOV of tracker.
	👁	The tool is in FOV of tracker.
	Coordinates	Gives current poses of tracked tools (DRF, Probe, Microscope and Checkerboard) in Cartesian coordinate system.

Table 19: Functionalities in navigation module

5.7.3 3D generation module

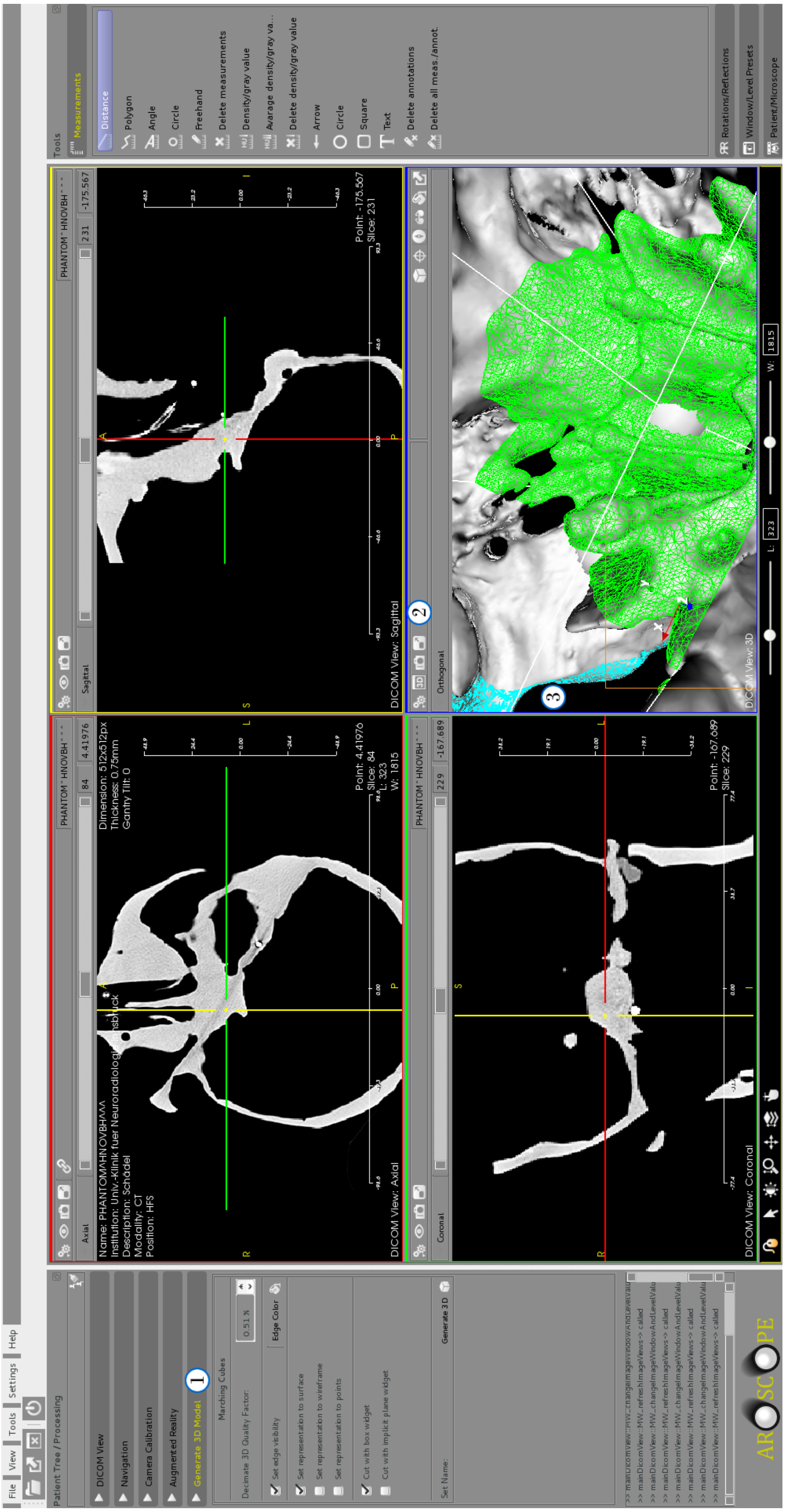


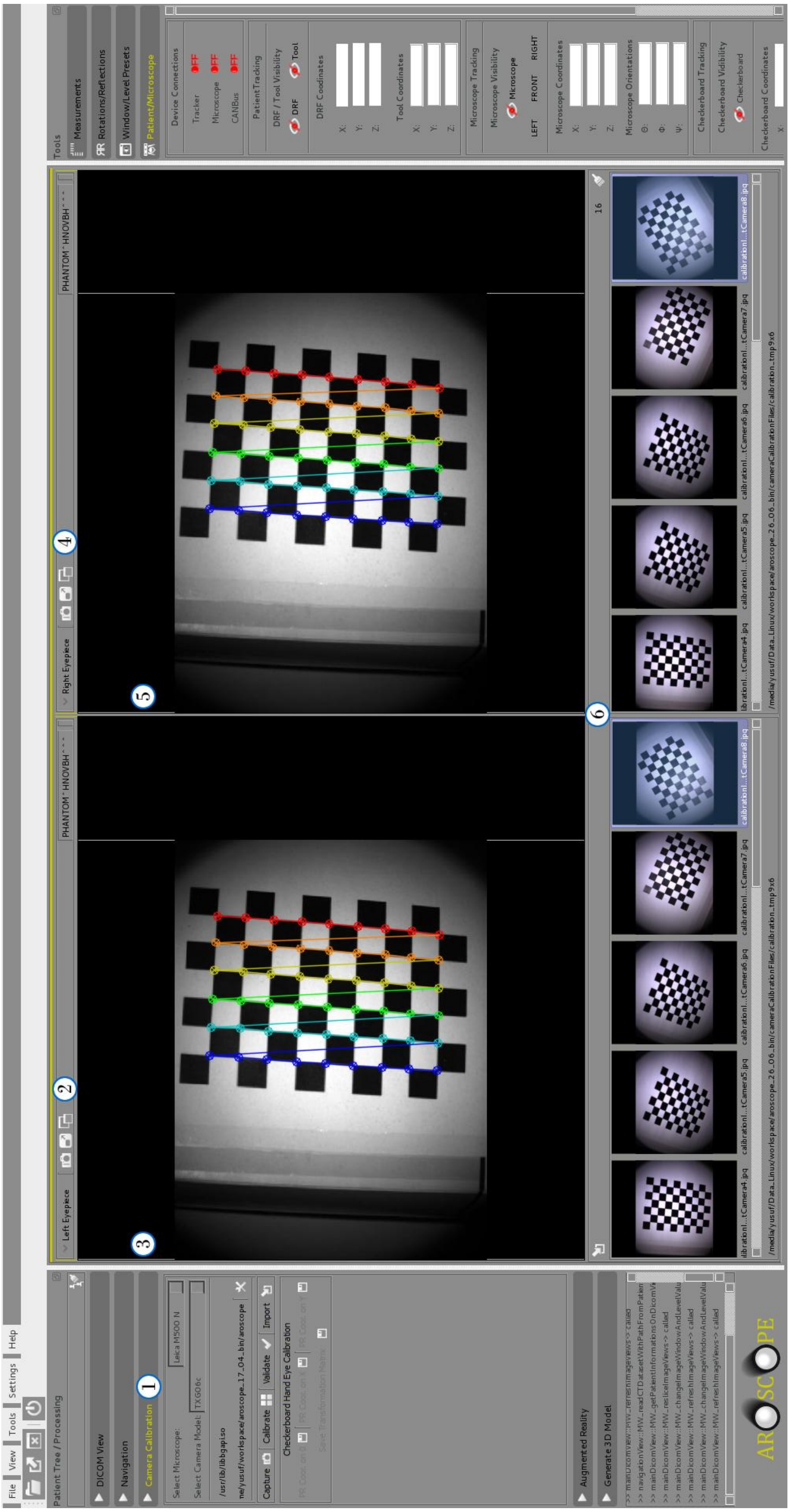
Figure 52: Overview of 3D generation module

Table 20 lists and describes the functionalities of specific elements, which are labeled on figure of 3D generation module:

Area	Icon / Text	Functionality
1	**Generate 3D Model**	Uses loaded patient images in DICOM module to generate 3D models iteratively.
-	Decimate Quality	Decimates the quality of created 3D model in factor by decreasing or increasing the number of triangles.
-	Edge Visibility	Turns on/off the visibility of edges on 3D model.
-	🪣	Sets color for displayed edges.
-	Representations	Offers possibility to represents 3D model as surface, wireframe or point cloud.
-	Clipping Options	Offers possibility to cut created model either with ROI box or implicit plane, which represents an infinite plane defined by a normal and point in the context of a bounding box. Through interaction with the widget, the plane can be manipulated by adjusting the plane normal or moving the origin point.
-	Model Name	Defines special name for created 3D model.
-	📦	Generates a 3D model with set properties.
2	**3D View Toolbar**	Has same functionality like in DICOM module.
3	**3D View**	Represents created 3D model.

Table 20: Functionalities in 3D generation module

5.7.4 Calibration module



Select Microscope: Leica M500 N

Select Camera Model: TXG06c

/usr/lib/libbgapi.so

ne/yusuf/workspace/aroscope_17_04_bin/aroscope

/media/yusuf/Data_Linux/workspace/aroscope_26_06_bin/cameraCalibrationFiles/calibration_tmp9x6

```
>> mainDicomView::MW_refreshImageViews-> called
>> navigationView::MW_readCTDatasetWithPathFromPatien
>> mainDicomView::MW_getPatientInformationsOnDicomVi
>> mainDicomView::MW_resliceImageViews-> called
>> mainDicomView::MW_changeImageWindowAndLevelValu
>> mainDicomView::MW_refreshImageViews-> called
>> mainDicomView::MW_changeImageWindowAndLevelValu
>> mainDicomView::MW_refreshImageViews-> called
```

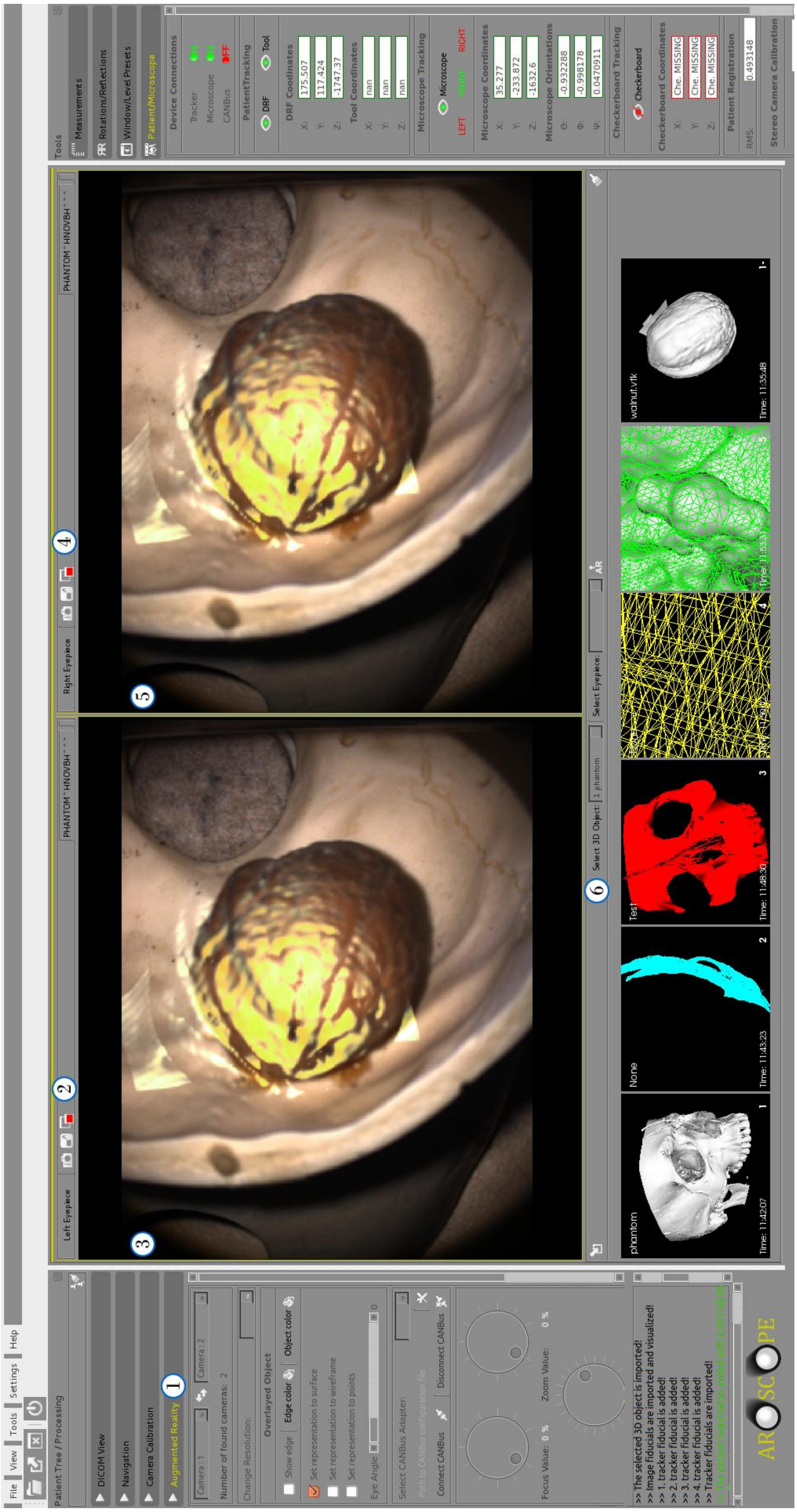
Figure 53: Overview calibration module

Table 21 lists and describes the functionalities of certain elements, which are labeled on figure of calibration module:

Area	Icon / Text	Functionality
1	**Calibration**	Interacts with microscope, provides stereo camera calibration and hand-eye calibration functionalities.
-	Combo box	Selects the microscope and build cameras on it, with which the user will work. This version interacts only with Leica M500 N microscope and Baumer TXG06c cameras.
-	🔌 🔌	Connects/Disconnects microscope with/from the workstation.
-	📷	Captures image from first and second camera for the calibration.
-	✚	Calibrates both cameras and saves calibration results in files.
-	↘	Imports calibration parameters from selected directory into the current session for the use in augmented reality module, which are saved in previous session or imports taken images from both cameras in image list, to calibrate cameras without capturing any new images.
	✓	Validates the calibration process.
-	Hand eye calibration	Performs hand eye calibration.
2	**Left Eyepiece Toolbar**	Provides functionalities to handle certain operations for the real-time video stream from first camera.
-	Change View	Switches the view from left to right camera on the selected window or vice versa.
3	**Left Eyepiece View**	Displays live video stream from first camera or displays images during camera calibration, which are taken with first camera.
4	**Right Eyepiece Toolbar**	Provides functionalities to handle certain operations for the real-time video stream from second camera.
5	**Right Eyepiece View**	Displays live video stream from second camera or displays images during camera calibration, which are taken with second camera.
6	**Image List**	Contains captured or imported images from first and second camera with the directory path, where the images are located and with number of captured or imported images.
-	🖌	Clears the calibration image list.

Table 21: Functionalities in calibration module

5.7.5 Augmented reality module



Camera : 1

Camera : 2

Number of found cameras: 2

Eye Angle: 0

Focus Value: 0 %

Zoom Value: 0 %

Select 3D Object: 1 phantom

Time: 11:42:07

Time: 11:43:23

Time: 11:48:30

Time: 11:35:48

DRF Coodinates
X: 175.507
Y: 117.424
Z: -1747.37

Tool Coordinates
X: nan
Y: nan
Z: nan

Microscope Coordinates
X: 35.277
Y: -233.872
Z: -1632.6

Microscope Orientations
θ: -0.932288
Φ: -0.998178
Ψ: 0.0470911

Checkerboard Coordinates
X: Che. MISSING
Y: Che. MISSING
Z: Che. MISSING

Patient Registration
RMS: 0.493148

```
>> The selected 3D object is imported!
>> Image fiducials are imported and visualized!
>> 1. tracker fiducial is added!
>> 2. tracker fiducial is added!
>> 3. tracker fiducial is added!
>> 4. tracker fiducial is added!
>> Tracker fiducials are imported!
>> [illegible]
```

Figure 54: Overview augmented reality module

Table 22 lists and describes the functionalities of all elements, which are labeled on figure of augmented reality module:

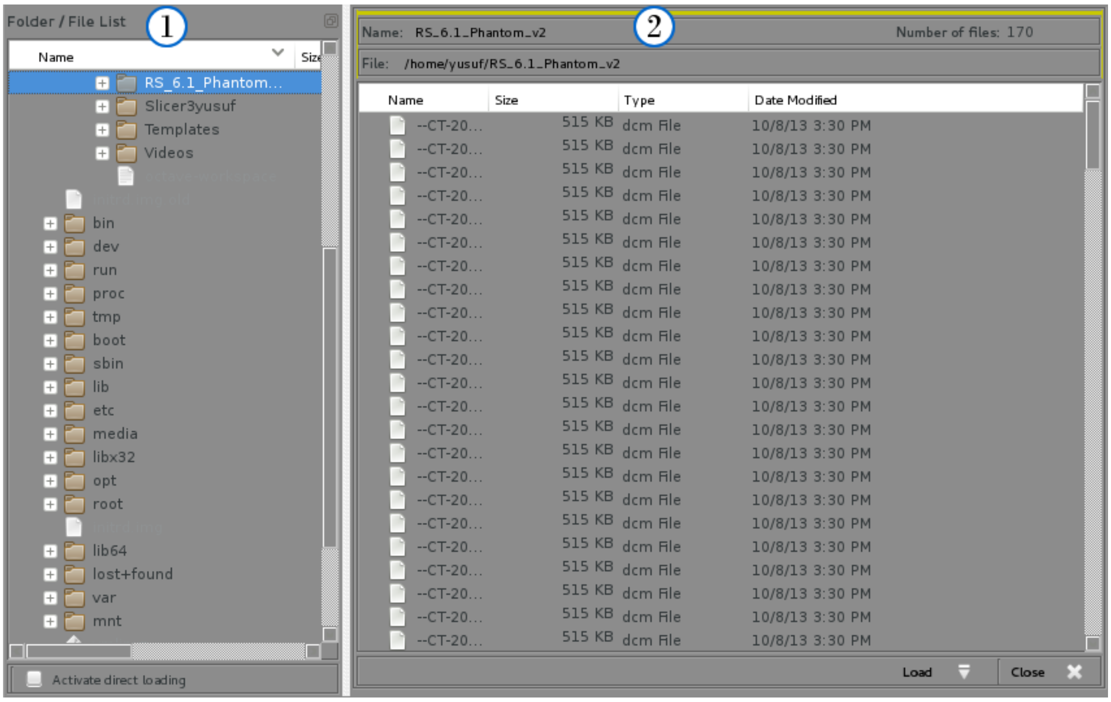
Area	Icon / Text	Functionality
1	**Augmented Reality**	Offers processing options to realize intraoperative AR.
-	⇆	Switches the view from left to right camera on the selected window or vice versa.
-	Combo box *	Changes resolution of cameras or set used CAN-Bus adapter.
-	✕	Opens settings window, to reconfigure settings for the selected microscope.
-	✗ ✗ *	Connects/Disconnects CAN-Bus with/from the workstation.
-	Dials *	Manipulates certain microscope settings (focus, zoom, working distance, etc.) over workstation.
2	**Left Eyepiece Toolbar**	Provides functionalities to handle particular operations for the real-time video stream from first camera.
-	🟩 🟥	Turns on/off the overlaying for the selected ocular.
3	**Left Eyepiece View**	Displays live video stream from first camera with overlayed object.
4	**Right Eyepiece Toolbar**	Provides functionalities to handle particular operations for the real-time video stream from second camera.
5	**Right Eyepiece View**	Displays live video stream from second camera with overlayed object.
6	**3D Model List**	Lists generated 3D models, which are produced by 3D generation module or imported 3D models, which are logged in previous sessions.
-	↘	Imports 3D models from selected directory into the current session for the use overlaying operations, which are saved in other session.
-	Select Model	Provides possibility to choose listed 3D model for overlaying.
-	Select Eyepiece	Provides possibility to choose listed eyepieces, in which the overlaying will occurs.
-	↑ AR	Injects selected model into the selected ocular.
-	🧹	Clears the 3D model list.

Table 22: Functionalities in augmented reality module

5.8 Sub-windows and functionalities

Beside main application window, there exist other sub-windows in AROSCOPE, named open file- and settings window to load patient image data sets from selected directory into the main window for desired operations or to set and control user-defined settings for the managing of AROSCOPE’s components.

5.8.1 Open file



Name: RS_6.1_Phantom_v2 Number of files: 170

File: /home/yusuf/RS_6.1_Phantom_v2

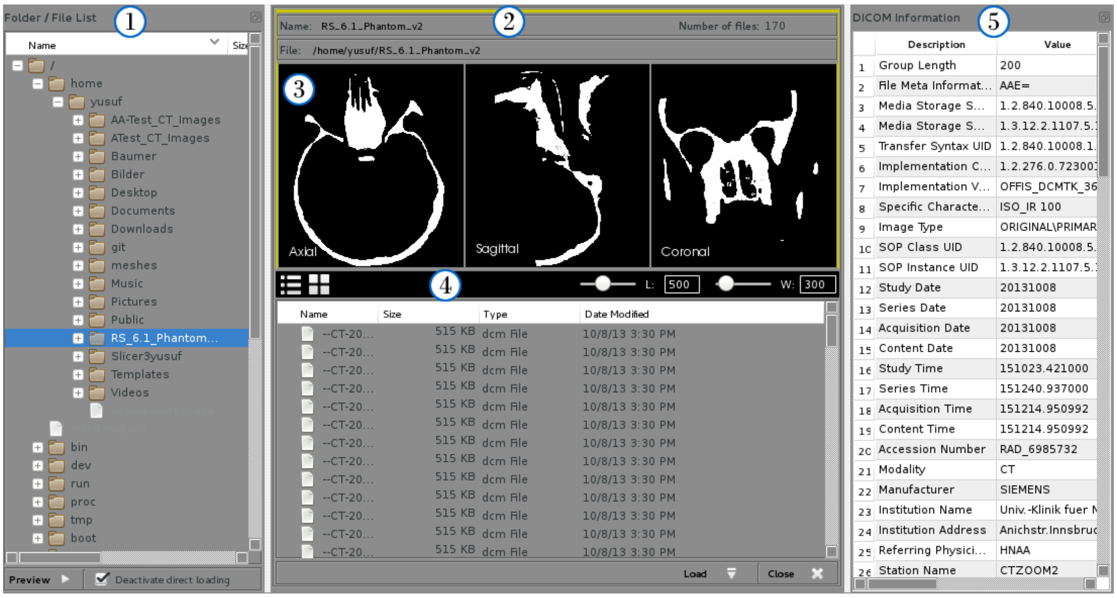
Name	Size	Type	Date Modified
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM

Figure 55: Open file window first modus

Table 23 lists and describes the functionalities of all elements on open file window first modus, which are labeled on above figure:

Area	Icon / Text	Functionality
1	**Folder/File List**	A modular sidebar (left frame) that lists the directory tree in workstation hierarchically.
-	Direct Loading	Turns on/off the modus of open file window.
2	**Folder Content**	Lists the content of selected directory with information such directory- name, path and number of files in directory. It displays only the files with extension .DCM (DICOM file format); if there any file exists with this filtered extension.
-	▼	Loads the content of selected directory, which contains DICOM files into the patient tree and main scene on main window.

Table 23: Functionalities in open file window first modus



Name: RS_6.1_Phantom_v2

Number of files: 170

File: /home/yusuf/RS_6.1_Phantom_v2

L: 500

W: 300

Name	Size	Type	Date Modified
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM
--CT-20...	515 KB	dcm File	10/8/13 3:30 PM

	Description	Value
1	Group Length	200
2	File Meta Informat...	AAE=
3	Media Storage S...	1.2.840.10008.5.
4	Media Storage S...	1.3.12.2.1107.5.
5	Transfer Syntax UID	1.2.840.10008.1.
6	Implementation C...	1.2.276.0.72300
7	Implementation V...	OFFIS_DCMTK_36
8	Specific Characte...	ISO_IR 100
9	Image Type	ORIGINAL\PRIMAR
10	SOP Class UID	1.2.840.10008.5.
11	SOP Instance UID	1.3.12.2.1107.5.
12	Study Date	20131008
13	Series Date	20131008
14	Acquisition Date	20131008
15	Content Date	20131008
16	Study Time	151023.421000
17	Series Time	151240.937000
18	Acquisition Time	151214.950992
19	Content Time	151214.950992
20	Accession Number	RAD_6985732
21	Modality	CT
22	Manufacturer	SIEMENS
23	Institution Name	Univ.-Klinik fuer
24	Institution Address	Anichstr.Innsbruc
25	Referring Physici...	HNAA
26	Station Name	CTZOOM2

Figure 56: Open file window second modus

Table 24 lists and describes the functionalities of all elements on open file window second modus, which are labeled on above figure:

Area	Icon / Text	Functionality
1	**Folder/File List**	Has same functionality like in the first modus.
-	Direct Loading	Turns on/off the modus of open file window.
-	▶	Loads the selected DICOM files into the preview view, before loading them in main window.
2	**Folder Content**	Has same functionality like in the first modus.
3	**Views**	Represents the images as orthogonal views in appropriate windows with the same mouse interaction possibilities like in DICOM module.
4	**Views Toolbar**	Provides functionalities to handle certain operations for the visualized images.
-	☰	Enables to show the DICOM images in selected directory content as list.
-	▦ *	Enables to show the DICOM images in selected directory content as thumbnail view.
-	L and W	L defines level value using a slider between min -1000 and max 3000. W defines width value using a slider between 0 and (max level value - min level value).
-	▼	Has same functionality like in the first modus.

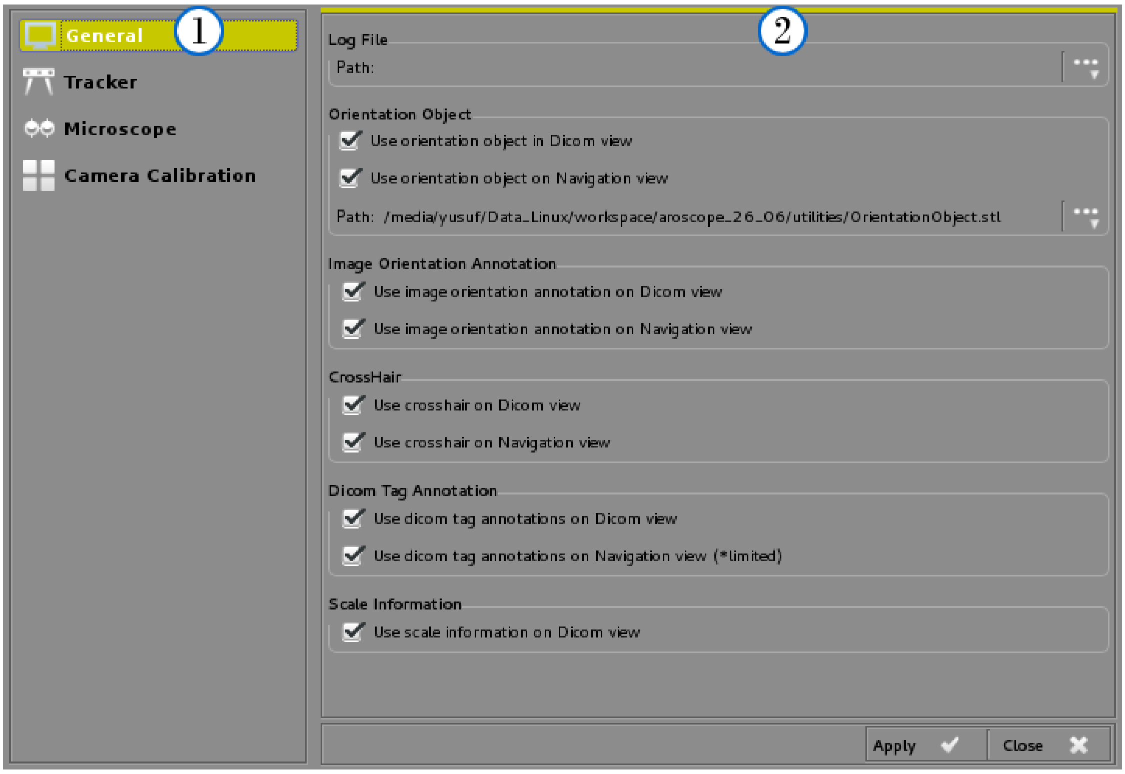
5	**DICOM Tags**	Lists all DICOM tags defined in its header with description, value and group entries. It is a modular sidebar (right frame) that can be placed on left or right sides of open file window.

Table 24: Functionalities in open file window second modus

5.8.2 Settings

To characterize the application according to the user needs, the control of functional treatment in AROSCOPE occurs by managing of settings files in simple .INI file formats, which create a basic structure composed of sections (groups), properties, and values. The applied settings are categorized and saved under three different files in a directory named *configurations:*

1. *settingsGlobal*: Contains user-defined certain configuration parameters for the main application. The parameters in file are read at begin of any session or by loading of a new patient image to the main view and handled accordingly.
2. *settingsTrackerConfigurationFile*: Contains user-defined specific configuration parameters related to the used tracker. The set values in file are read and handled by starting the tracker for the navigation.
3. *settingsMicroscopeConfigurationFile*: Contains user-defined specific configuration parameters related to the used microscope and camera.



Log File

Path:

Orientation Object

☑ Use orientation object in Dicom view

☑ Use orientation object on Navigation view

Path: /media/yusuf/Data_Linux/workspace/aroscope_26_06/utilities/OrientationObject.stl

Image Orientation Annotation

☑ Use image orientation annotation on Dicom view

☑ Use image orientation annotation on Navigation view

CrossHair

☑ Use crosshair on Dicom view

☑ Use crosshair on Navigation view

Dicom Tag Annotation

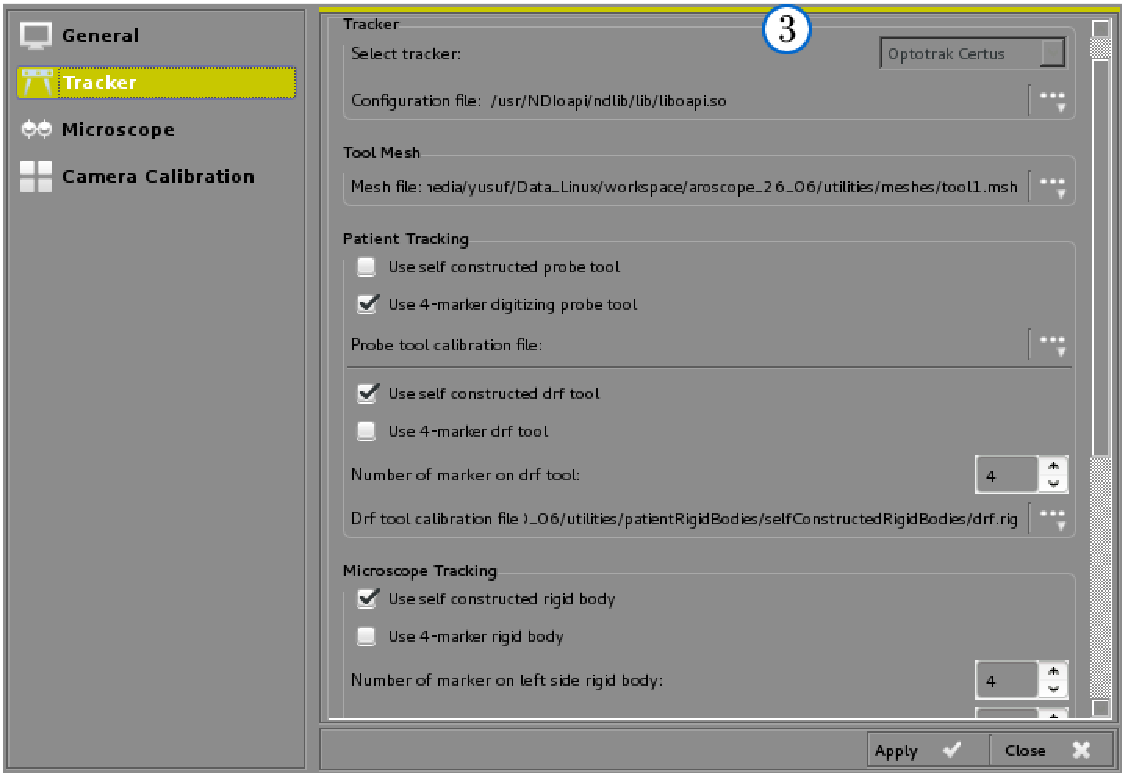
☑ Use dicom tag annotations on Dicom view

☑ Use dicom tag annotations on Navigation view (*limited)

Scale Information

☑ Use scale information on Dicom view

Figure 57: Main application settings



Tracker

Select tracker: Optotrak Certus

Configuration file: /usr/NDIoapi/ndlib/lib/liboapi.so

Tool Mesh

Mesh file: ıedia/yusuf/Data_Linux/workspace/aroscope_26_06/utilities/meshes/tool1.msh

Patient Tracking

☐ Use self constructed probe tool

☑ Use 4-marker digitizing probe tool

Probe tool calibration file:

☑ Use self constructed drf tool

☐ Use 4-marker drf tool

Number of marker on drf tool: 4

Drf tool calibration file)_06/utilities/patientRigidBodies/selfConstructedRigidBodies/drf.rig

Microscope Tracking

☑ Use self constructed rigid body

☐ Use 4-marker rigid body

Number of marker on left side rigid body: 4

Figure 58: Tracker settings

Table 25 lists and describes the functionalities of most significant elements on settings window, which are labeled on above figures:

Area	Icon / Text	Functionality
1	**Categories**	Lists different categories of the setting components.
2	🖥	Provides the possibility to manage the main application settings by using check- or combo boxes. Offers configuration possibilities e.g. defining the location of logging directory, orientation object, cross hair, etc.
3	⊓	Provides the possibility to manage the used tracker settings. Serves configuration possibilities for e.g. kind of used tools for patient and microscope tracking or rather characterize the used tools, and set the .RIG files, etc.
4	👓	Offers the possibility to manage the used microscope settings.
5	⊞	Offers the possibility to manage settings for camera calibration process. E.g. defining the size of calibration pattern.
-	⋯	Opens file chooser to set required file.
-	✓	Applies the changes and saves them in related configuration file.

Table 25: Functionalities on settings window

5.9 Further development

In the next version of AROSCOPE, it is planned to complete the all missing functionalities, troubleshooting of virtual camera and implement whole project within including the plug-in framework of The Common Toolkit (CTK), in order to implement the modules and other necessary classes as separated plug-ins. Since the design of the CTK framework allows the creation of modular CAS applications. Furthermore the application should be able to work on multiple OS. Especially the CAN-Bus protocol should be adapted to the system, in order to get the current zoom and focus parameters from microscope control unit in real-time and according obtained value the representation of augmented reality should be handled.

For this it is planned to offer the CTK part of project as a bachelor thesis to other students to improve the utility and for further development.

CHAPTER 6 - Summary

6.1 Conclusions

In this present thesis is presented, a concept for ARS in minimal invasive microscopic surgeries for ENT or neurosurgery operations with a navigated surgical microscope and developing procedures with used components.

In the center of this work stood, first the image-patient registration, navigation of patient and surgical microscope that uses an optical tracking system to perform DRF based standard patient and device navigation using special algorithms and techniques in open-source libraries, which allowed the submillimetric accuracy in surgical intervention procedures. Secondly core area was the calibration of cameras on microscope head that served live video stream from microscope to the workstation and 2D projections of prepared 3D objects in microscope oculars using obtained parameters, which provided entire ARS with millimetric accuracy in overlaying. Some of used components and algorithms were modeled schematically, formally and explained mathematically in accordance with the focus of this work and evaluated in terms of accuracy.

Another important focus of development was on the building the best possible user-friendly interface between the user, modules, and workstation to meets the requirements depending on conducted surgery. Thus AROSCOPE offers integration, connections of all main- and sub-components, acceptable guidance in microscopic surgeries with developed clever modular user interface, which leads the surgeon in progression from DICOM visualization until ARS and it allows for more precise, and more targeted interventions, especially in critical operations.

Other special feature of the work is the choice of soft- and hardware components and algorithms for a complete scenario to build an ARS. Noteworthy are the special problems of acceptance and the high demands on the accuracy and real-time capability, which correlate particularly in operations with the security.

6.2 References

[1] P. Edwards, A. King, J. Maurer CR, D. de Cunha, D. Hawkes, D. Hill, R. Gaston, M. Fenlon, A. Jusczyzck, A. Strong, C. Chandler and M. Gleeson, Design and Evaluation of a System for Microscope-Assisted Guided Interventions (MAGI), IEEE Transactions On Medical Imaging, Vol. 19, No. 11, November 2000.

[2] M. Moche, R. Trampel, T. Kahn and H. Busse, Navigation concepts for MR image-guided interventions, J Magn Reson Imaging, 27(2), pp. 276-91, DOI: 10.1002/jmri.21262, 2008.

[3] E. Geist and K. Shimada, Position error reduction in a mechanical tracking linkage for arthroscopic hip surgery, Int J Comput Assist Radiol Surg. 6(5):693-8, DOI: 10.1007/s11548-011-0555-7. Epub 2011 Apr 2, September 2011.

[4] B. Wood, H. Zhang, A. Durrani, N. Glossop, S. Ranjan, D. Lindisch, E. Levy, F. Banovac, J. Borgert, S. Krueger, J. Kruecker, A. Viswanathan and K. Cleary, Navigation with Electromagnetic Tracking for Interventional Radiology Procedures, J Vasc Interv Radiol. 16(4): 493–505, DOI: 10.1097/01.RVI.0000148827.62296.B4, April 2005.

[5] Z. R. Bardosi, Y. Özbek, C. Plattner, F. Kral and W. Freysinger, Auf dem Weg zum Heiligen Gral der 3D-Navigation: submillimetrische Anwendungsgenauigkeit im Felsenbein, Wolfgang Freysinger (Ed.), Proceedings of the 12th Annual Conference of the German Society for Computer- and Roboter-Assisted Surgery (CURAC), 2013.

[6] F. Kral, E. Puschban, H. Riechelmann, F. Pedross and W. Freysinger, Optical and electromagnetic tracking for navigated surgery of the sinuses and frontal skull base, Rhinology;49(3):364-8, DOI: 10.4193/Rhino10.177, August 2011.

[7] A. Colchester, J. Zhao, K. Holton-Tainter, C. Henri, N. Maitland, P. Roberts, C. Harris and R. Evans, Development and preliminary evaluation of VISLAN, a surgical planning and guidance system using intra-operative video imaging, Department of Neurology, UMDS, Guy's Hospital, London, UK: Medical Image Analysis, March 1996.

[8] A. L. Janin, K. Zikan, D. Mizell, M. Banner and H. A. Sowizral, Videometric head tracker for augmented reality applications, Boston, MA: Proc. SPIE 2351, Telemanipulator and Telepresence Technologies, 308, DOI:10.1117/12.197323, December 1995.

[9] W. Birkfellner, J. Hummel, E. Wilson and K. Cleary, Tracking Devices, T. Peters and K. Cleary (eds.), Image-Guided Interventions, Springer Science + Business Media, LLC, 2008.

[10] O. Schuppe, An optical tracking system for a microsurgical training simulator, Stud Health Technol Inform.,173:445-9., 2012.

[11] Optotrak Certus User Guide Revision 6, 103 Randall Dr. Waterloo, Ontario Canada N2V 1C5: Northern Digital Inc. IL-1070106, November 2014.

[12] NDI 6D Architect User Guide Revison 4.0, Ontario Canada: Northern Digital Inc. Part Nr:IL-1070059, March 2004.

[13] W. Onprasert, A Novel Method on Tool Tip Calibration for Biomedical Application, The World Congress on Computer Science and Information Engineering, pp. 650-653, 2011.

[14] C. R.Maurer, J. Fitzpatrick, M. Y.Wang, R. L. Galloway, M. Robert J. and G. S. Allen, Registration of head volume images using implantable fiducial markers, IEEE Transaction on Medical Imaging, 16(3), pp. 447-462, 16, 1997.

[15] K. Cleary, P. Cheng, A. Enquobahrie and Z. Yaniv, Image Guided Surgery Toolkit: The Book, For release 4.2, Insight Software Consortium, 2009.

[16] G. Strang, Introduction to Linear Algebra, Wellesley-Cambridge Press, ISBN-10: 0980232716, 2009.

[17] B. K. Horn, Closed-form solution of absolute orientation using unit quaternions, Journal of the Optical Society of America, Vol. 4, No.4, pp. 629-642, April 1987.

[18] J. M. Fitzpatrick, L. G. H. Derek and J. Calvin R. Maurer, *Introduction to Biomedical Imaging and Image Analysis*, Pittsburgh, PA, 15213: Carnegie Mellon University, 2011.

[19] Y. Özbek, "wopsys.com," 25 March 2012. [Online]. Available: http://www.wopsys.com/?p=1265. [Accessed 21 06 2015].

[20] P. J. Besl and N. McKay, A Method for Registration of 3-D Shapes, Los Alamitos, CA, USA: IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume:14 , Issue: 2), pp. 239-256, DOI:10.1109/34.121791, 1992.

[21] L. Dorst, D. Fontijne and S. Mann, Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry, Oxford Elsevier LTD, ISBN-13: 978-0123749420, April 2007.

[22] R. R. Shamir, L. Joskowicz and Y. Shoshan, Optimal landmarks selection and fiducial marker placement for minimal target registration error in image-guided neurosurgery, Medical Imaging: Visualization, Image-Guided Procedures and Modeling, Proc. of SPIE Vol. 7261, DOI: 10.1117/12.810897, 2009.

[23] R. Tsai und R. Lenz, A new technique for fully autonomous and efficient 3D robotics hand-eye calibration, IEEE Trans. Robotics and Automation, vol. 5, pp. 345-358, June 1989.

[24] Ö. Güler, M. Perwoeg, F. Kral and W. Freysinger, Quantitative error analysis for computer assisted navigation: A feasibility study, Article in Medical Physics, Impact Factor: 2.64, DOI: 10.1118/1.4773871, Source: PubMed, February 2013.

[25] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision Second Edition, The Edinburgh Building, Cambridge, CB2 2RU, UK: Cambridge University Press, 2003.

[26] Z. Zhang, A Flexible New Technique for Camera Calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), pp.1330-1334, 2000, December 1998.

[27] T. Roger Y., A Versatile Camera Calibration for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses, IEEE Journal of Robotics and Automation, Vol. RA-3, No.4, August 1987.

[28] B. Cyganek and J. P. Siebert, An Introduction to 3D Computer Vision Techniques and Algorithms, John Wiley and Sons, Ltd. ISBN: 978-0-470-01704-3, 2009.

[29] L. G. Roberts, Homogeneous matrix representation and manipulation of n-dimensional constructs, Massachusetts Institute of Technology Lincoln Laboratory, Document No. MS1045, May 1965.

[30] A. L. COMT, Optics, Retinoscopy, and Refractometry, Slack Incorporated, 2nd edition, ISBN-10: 1556427484, December 2005.

[31] d. t. opencv, "opencv documentation," 2011-2014. [Online]. Available: http://docs.opencv.org/index.html. [Accessed 30 06 2015].

[32] C. Harris and M. Stephens, A combined corner and edge detector, United Kingdom: Proceedings of the 4th Alvey Vision Conference, pp. 147-151, 1988.

[33] Y. Abdel-Aziz and H. Karara, Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry, Falls Church, VA: American Society of Photogrammetry: In Proc. Symp. Close-Range Photogrametry, pp.1-18, 1971.

[34] D. G. Luenberger, Least-Squares Estimation.Optimization by Vector Space Methods, New York: ISBN 0-471-18117-X, pp. 78-102, 1997.

[35] Baumer-GAPI SDK v1.7, Programmer‘s Guide, Radeberg, Germany: Baumer Optronic GmbH, 2012.

[36] M. Aschke, C. R. Wirtz, J. Raczkowsky, H. Worn and S. Kunze, Augmented reality in operating microscopes for neurosurgical interventions, Proceedings of the first International IEEE EMBS Conference on Neural Engineering, DOI: 10.1109/CNE.2003.1196913, March 2003.

[37] Y. Masanori, T. Saito, T. Kin, D. Nakagawa, H. Nakatomi, H. Oyama and N. Saito, A Microscopic Optically Tracking Navigation System That Uses High-resolution 3D Computer Graphics, Neurol. Med. Chir. (Tokyo), 55(8), pp. 674-679, DOI:10.2176/nmc.tn.2014-0278, Jul 2015.

[38] S. Vogt, Real-Time Augmented Reality for Image-Guided Interventions, PhD Thesis, Nürnberg: Der Technischen Fakultät der Universität Erlangen, 2009.

[39] M. Paul, H. Takemura, A. Utsumi and F. Kishino, Augmented Reality: A class of displays on the reality-virtuality continuum, Kyoto, Japan: Proceedings of Telemanipulator and Telepresence Technologies. pp. 2351-34, Retrieved 2007-03-15, 1994.

[40] J. P. Rolland and H. Fuchs, Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization, MIT Press Cambridge, MA, USA: Teleoperators and Virtual Environments, Volume 9 Issue 3, pp. 287-309, June 2000.

[41] M. Figl, C. Ede, J. Hummel, F. Wanschitz, R. Ewers, H. Bergmann and W. Birkfellner, A Fully Automated Calibration Method for an Optical See-Through Head-Mounted Operating Microscope With Variable Zoom and Focus, IEEE Transactions on Medical Imaging, Vol. 24, No. 11, 2005.

[42] Leica M525 MC1 User Manual, Heerbrugg, Schweiz: Leica Microsystems (Schweiz) AG, Ref. 10 714 387, 2006.

[43] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, ISBN 10: 0596516134, 2008.

[44] Baumer TXG06c Technical Data, Badstrasse 30 DE-01454 Radeberg, Germany: Baumer Optronic GmbH, June 2013.

[45] Y. Özbek, CTTranslator, Computertomographie Bildbetrachtungs- und Segmentierungsprogramm: Bachelor Thesis, Innsbruck: Leopold Franzens Universität Institut für Informatik and Medizinische Universität Innsbruck Universitätsklinik für HNO, November 2008.

[46] W. E. Lorensen and C. Harvey E., Marching Cubes: A high resolution 3D surface construction algorithm, New York 12301: Computer Graphics, Vol. 21, Nr. 4, July 1987.

[47] W. Schroeder, K. Martin and B. Lorensen, Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition, Kitware, ISBN 10: 193093419X, 2006.

[48] D. Shreiner, The OpenGL Programming Guide: The Official Guide to Learning OpenGL Version 3.0 and 3.1, Addison-Wesley Professional, ISBN-10: 0321552628, 2009.

[49] A. Alexandrescu, Modern C++ Design: Generic Programming and Design Patterns Applied, Addison-Wesley Professional, ISBN-10: 0201704315, 2001.

[50] K. Martin and B. Hoffman, Mastering CMake, Kitware, Incorporated; 3.1 edition, ISBN-10: 1930934319, January 2015.