Faculty of Science and Technology of the University of Coimbra

Surgical Navigation using an Optical See-Through Head Mounted Display

João Pedro Silva Falcão

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Electrical and Computer Engineering, supervised by João Pedro de Almeida Barreto and Paulo Jorge Carvalho Menezes and presented to the Faculty of Science and Technology of the University of Coimbra.

July of 2019

UNIVERSIDADE D
COIMBRA

Agradecimentos

Em primeiro lugar começo por agradecer aos meus orientadores, o Professor João Pedro de Almeida Barreto e Professor Paulo Jorge Carvalho Menezes, por me terem concedido a oportunidade de realizar este trabalho e toda a partilha de conhecimento durante a realização do trabalho. Agradeço também à coorientadora Carolina Raposo por toda a ajuda prestada ao longo de toda a dissertação, todos os aconselhamentos fornecidos e acompanhamento durante todo o trabalho. Agradeço a todos os meus colegas da Universidade de Coimbra e em especial os colegas do laboratório de Automação pela partilha de todo o seu conhecimento e auxílio. Por fim, agradeço aos meus Pais por todo o seu apoio ao longo dos anos, toda a dedicação e incentivo prestados.

Coimbra, Julho de 2019

Resumo

Navegação Cirúrgica é um conceito recente que consiste em guiar um cirurgião em procedimentos médicos, tendo vários benefícios desde a melhoria de resultados clínicos, até à redução do tempo da aprendizagem do cirurgião. Recentemente, a Perceive3D desenvolveu o in.nav que consiste num sistema de navegação cirúrgica baseado em vídeo, que utiliza realidade aumentada para guiar procedimentos médicos. Com a utilização de uma camara monocular e de vários pequenos marcadores, o in.nav supera as duas principais desvantagens inerentes aos sistemas de *Optical Tracking*: o elevado investimento de capital necessário e a dificuldade de preservar informação no campo de visão do cirurgião na sala de operações. O sistema in.nav atualmente funciona em computadores e tablets, o que leva à necessidade de haver uma pessoa dedicada a segurar o sistema. Este trabalho tem como foco a implementação deste sistema em óculos de realidade aumentada, permitindo assim o melhoramento da visibilidade do ecrã pelo cirurgião (sem necessidade de desviar o foco da anatomia) e a remoção da necessidade segurar um tablet ou computador. Este trabalho consiste no desenvolvimento de uma aplicação que serve de interface entre in.nav e um *HMD*, pela qual toda a informação necessária é transmitida. Dois modos foram desenvolvidos: um primeiro modo com o objetivo de virtualizar um ecrã convencional (de *tablet* ou *PC*) e outro modo onde é visualizada informação de navegação cirúrgica sobreposta à anatomia. Por forma a possibilitar uma boa experiência com a utilização do segundo modo, foi também necessária uma calibração adequada dos óculos. Todos os testes e consequentes resultados de calibração foram realizados na parte final da dissertação, com uma descrição detalhada presente neste documento.

Apesar de todos os problemas inerentes da utilização de *HMD* (conflito de vergência acomodação), este trabalho teve como objetivo o desenvolvimento de uma aplicação simples e de fácil utilização num *HMD*.

Palavras-Chave: Navegação Cirurgica, in.nav, DreamGlass, *HMD*, Realidade Aumentada

Abstract

Surgical navigation consists in guiding the surgeon during medical procedures, having important benefits such as the improvement of the clinical outcome and the decrease in the surgeon's learning curve. in.nav is an intra-operative video-based navigation system developed by Perceive3D that provides augmented reality guidance for medical procedures, running on standard PCs and tablets. By using only a monocular camera and small fiducial markers, it overcomes the two main drawbacks inherent to the common Optical Tracking systems: the high cost and the difficulty in providing the necessary information in the view field of the surgeon during procedures in the Operating Room.

This dissertation is targeted at enabling in.nav to be used with an Head Mounted Display (HMD). This brings important advantages that include the improvement of the visibility of the screen by the surgeon, enabling him/her not to move sight from anatomy, and the fact that it removes the need for an extra pair of hands for holding the screen. This resulted in the development of an application that provides an interface between in.nav and the HMD, through which all the necessary information is transmitted. This application's user interface presents two different modes: one where the HMD serves to virtualize a conventional 2D display and another where the rendering of guidance information is done directly over the patient's anatomy. In order to provide a pleasant and useful user experience, the latter approach requires an accurate calibration of the HMD. This is accomplished in the last part of this thesis, as demonstrated by the thorough experimental validation.

Although the technology of HMDs still presents some problems, such as the vergence-accommodation conflict, the work developed in this thesis yielded a simple, accurate and easy to use AR headset for surgical navigation.

Keywords: Surgical Navigation, in.nav, DreamGlass, HMD, Augmented Reality

Contents

List of Acronyms

AR	Augmented Reality
VR	Virtual Reality
HMD	Head Mounted Display
TCP/IP	Transmission Control Protocol/Internet Protocol
SPAAM	Single-Point Active Alignment Method
MPAAM	Multiple-Point Active Alignment Method
OST	Optical See-Through
TKA	Total Knee Arthroplasty
P3D	Perceive3D
OT	Optical Tracking
CT	Computed Tomography
MRI	Magnetic Resonance Imaging
VAC	Vergence-Accommodation Conflict
FoV	Field of View
IPD	Inter Pupillary Distance
API	Application Programming Interface
I/O	Input/Output
DOF	Degrees of freedom
PC	Personal Computer

List of Figures

List of Tables

1 Introduction

Surgical navigation [20] aims to safely guide the surgeon during medical procedures, having important benefits both in terms of improving clinical outcome and in terms of decreasing the surgeon's learning curve, which is particularly useful for novice surgeons [23]. Surgical navigation typically requires the real-time localization of tools and instruments with respect to a common reference frame. To accomplish this, most solutions make use of Optical Tracking (OT) systems that consist of a stationary tower equipped with at least two infrared cameras for tracking a set of fiducial markers that are rigidly attached to the instruments and the patient's anatomy (bone or organ). Despite its obvious clinical benefits, the current penetration of surgical navigation is still low because of two main reasons: i) existing solutions require a significant investment in capital equipment due to the necessity of acquiring a base station, and ii) these solutions require that the base station has a clear line-of-sight to all the markers, constraining both the layout of the Operating Room and the movements of the medical team members [18].

In order to overcome these drawbacks inherent to OT systems, Perceive3D (P3D) recently introduced the concept of vision-based navigation where small fiducial markers with known patterns are attached to the instruments and the patient's anatomy and tracking is performed using a single monocular camera [7]. This not only avoids the need of a high capital investment, but also makes the preservation of the lines-of-sight much easier to satisfy since all measurements are performed at close range. Relevant information for guiding the surgeon through the medical procedure is provided using Augmented Reality (AR). Figure 1.1 shows P3D's system in.nav^{TM} being applied to the orthopedic procedure Total Knee Arthroplasty (TKA).

The interface between user and virtual world in AR can be done with various types of devices. in.nav currently runs in standard PCs and tablets, having the disadvantages that an extra pair of hands is required for holding the device and that it may be difficult for the surgeon to have good visibility of the screen. Combining in.nav with AR headsets would

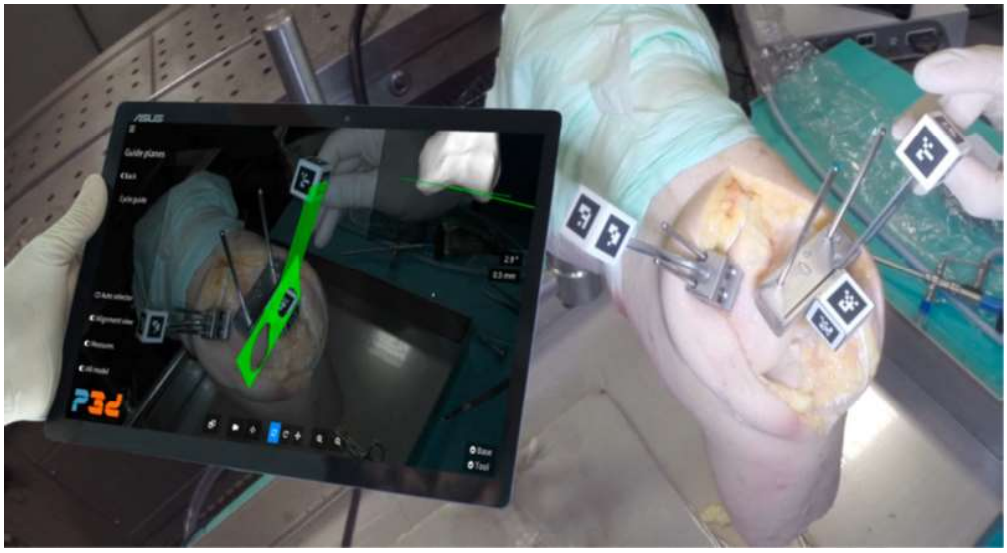


Figure 1.1: in.nav running on a Tablet/PC, being applied in TKA

effectively tackle these problems, and thus it comes as a very natural development.

AR headsets are a popular type of tool because of their handleability and immersion. In the latest years, the development of headsets for AR has increased conjointly with the hardware development, creating the opportunity for new innovative projects. Microsoft Hololens™, Magic Leap One™, Daqri™, Meta2™, DreamGlass™ and Varjo™ are some of the most popular AR see-through headsets released, with each headset having different features as described in section 1.2.1.

Although integrating in.nav in an AR headset seems like a straightforward development, there are important perceptual challenges to overcome that stem from the need of making insertions at close range. The challenges are mainly due to vergence-accomodation and accuracy issues. The vergence-accomodation problem consists in the conflict between the normal human eye view and the Virtual Reality (VR) or AR unnatural view of objects. This conflict can generate side effects that need to be taken into account in order to provide a healthy experience during long uses. Since in.nav is a system for surgical navigation, its successful operation requires highly accurate measurements. Thus, a proper calibration of the AR headset is necessary, for which several solutions exist in the literature [11], with the most accurate ones including eye-tracking [14, 11].

Although there are reasonable doubts that the technology of head-mount display (HMD) is mature enough for short range rendering with a perceptual quality and comfort that enable its use in surgery [9], the objective of this thesis is to carry a first attempt at implementing in.nav in a modern HMD (the DreamGlass from DreamWorld). The remainder of this section provides an overview of the in.nav system as well as a description of the most relevant existing

HMDs. Section 1.3 details the objective and main contributions of this thesis.

1.1 in.nav

in.nav^{TM} is a system for intra-operative navigation based in video that combines intelligent image processing with AR to safely guide the surgeon through different medical procedures. This section briefly describes how it works and the required equipment. in.nav starts by tracking visual markers in real time [19], one attached to the patient to work as the world marker and another attached to a touch probe, referred to as tool marker. While the probe touches the anatomy, 3D contours are reconstructed and an on-the-fly registration algorithm is used for aligning the pre-operative virtual model with the patient's anatomy (Fig. 1.2). The 3D virtual model is obtained by segmenting a pre-operative image of the bone, which can be a computed tomography (CT) or a magnetic resonance imaging (MRI) scan. Lastly, guidance information is provided by transforming all the measurements performed before the surgery into the world markers's reference frame, and showing them using AR accurately overlaid on the targeted anatomy. Relevant to this project is the fact that in.nav takes as input a 3D model of the patient's bone and outputs the poses of all visible markers at every instant as well as the rigid transformation that aligns the 3D model with the anatomy.

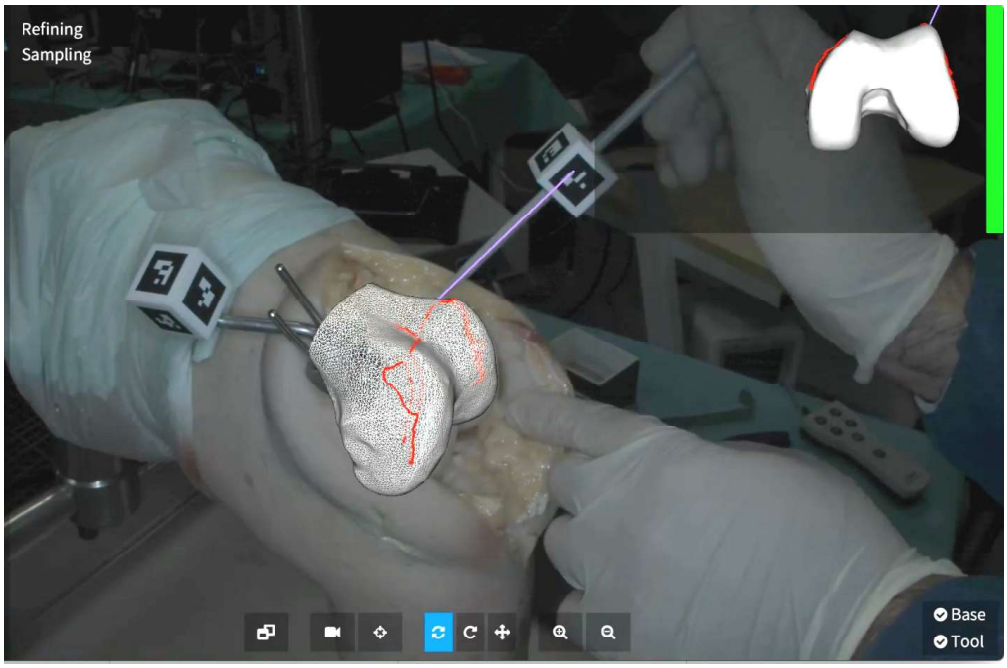


Figure 1.2: in.nav applied to the TKA procedure showing the pre-operative model overlaid with the anatomy, as well as the reconstructed 3D curves used for registration

1.2 Background

1.2.1 Headsets

The timeline in Fig. 1.3 illustrates the development of new technologies that arose in the past 30 years, since the creation of AR.

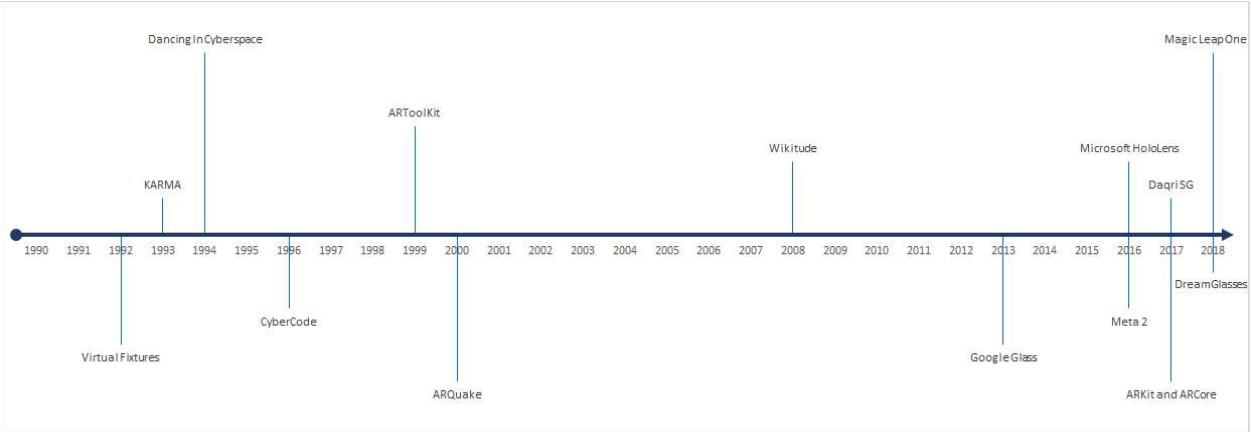


Figure 1.3: Timeline of the development of new technologies related with AR.

One important factor for the observable sudden development of AR headsets in recent years is justified by the development of better hardware and the improvement it generates for new implementations that could not be done before. Better eye-tracking, improved calibration methods and higher quality displays are part of the improvements that led to more accurate applications. Table 1.1 shows the most recent headsets and a comparison between features and prices. Some features are unique for some headsets. As an example, Hololens is equipped with gaze tracking and automatic calibration, which improve user experience (Quality of Life improvements) by not requiring any keys to be pressed during calibration (the user just needs to use the finger). Also, Daqri and Hololens are equipped with a Depth Camera, which provides additional tools for user developments. All these features, however, have a cost that is reflected in the price of the headset. In general, all headsets have similar prices, with the exception of the Daqri headset that is a high end headset and DreamGlass that is a low cost option.

1.2.2 Medical Applications

Recently, there has been an increasing trend to use HMDs for medical applications. The most relevant solutions are described next.

Head Mounted Displays Offers		
HMDs	Main Features	Price (euros)
Hololens Microsoft	Gaze tracking, Gesture control, Depth Camera, IMU, Automatic pupillary distance calibration, 4 environment understanding cameras and Bluetooth connection	2642
Magic Leap One	LRA Haptic Device, Magnetic Controller, IR Camera, 6 DoF tracking and Bluetooth	2022
Daqri	44º Diagonal FOV, Dual LCoS Optical Displays, 90 fps Optics, Bluetooth, 6-DOF tracking, Depth Sensor Camera (30-90 fps, 0.4 to 4 meters), AR Tracking and Color Camera (30 fps)	More than 10000
Meta2	90º FoV, Hand interaction, positional tracking sensors and 720p RGB camera	1755
DreamGlass	90º diagonal FoV, Hand gesture recognition, 3 DoF head tracking, IR camera and 1080p front-facing RGB camera	546

Table 1.1: Head Mounted Displays availability and features

Augmedics

Augmedics is an Israeli startup working on AR headsets for surgeons performing spinal surgery. Their product, Xvision™ (Fig. 1.4a), consists in a surgical navigation system with application in neurosurgery and spine procedures providing surgeons with 3D rendered information, more specifically, trajectory projections in the customizable user interface. The authors claim that it reduces the time necessary for surgeries as well as radiation exposure, being beneficial for the user. Its system provides the visualization of the spinal vertebra, leading to the possibility of being applied in several surgeries, ranging from spine issues to deformities and other cases [5].

OpenSight

Opensight™ (Fig. 1.4b), designed by Novarad, is a pre-surgery planning system that provides several solutions by overlaying 2D and 3D patient anatomy images with the patient's body. The headset used for this software is the Microsoft Hololens™ and it is possible to share the information between multiple headsets, facilitating the training process for less experienced users [16].

ARSpectra

ARSpectra (Fig. 1.4c) created glasses for surgical navigation applied in the resection of cancerous tissues in open surgery. The designed glasses and software improve the information displayed by allowing to examine diagnostic images inside the field of view of the user (overlaid on patient body). Since the headset was designed by the same company, it is optimised for the methods developed [4].

Insight Medical

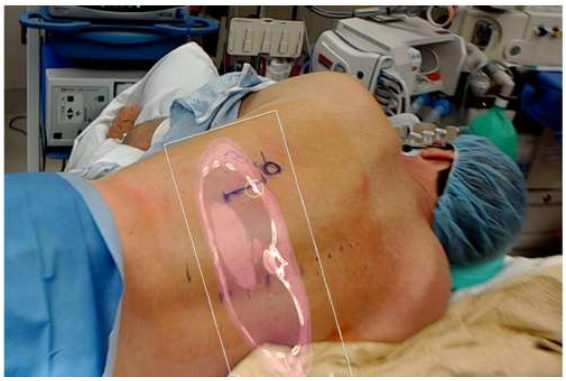
Insight Medical created ARVIS^{TM} (Fig. 1.4d), an AR headset designed for display of virtual models of the patient's anatomy, during both preoperative planning and surgery. Additional critical information is displayed during the procedure, such as the virtual models of the implants and instruments [22].

Although there are several companies working in the same area of application of this project, the medical field is vast and diverse, allowing several different developments. All of the existing solutions differ in some way. Augmedics focused their product on spinal surgery, OpenSight from Novarad is a pre-surgery solution used for planning, ARSpectra's product was developed for open surgery with resection of cancerous tissues being their area of intervention. ARVIS from Insight Medical is the closest solution to this project in terms of concept as it provides guidance information to the surgeon using AR. In general these implementations work well at large distances, mostly aimed for surgery planning with the downside of decreasing performance when working at close range, where error is more noticeable. The goal of this work is to develop a new tool with high accuracy and better rendering in close range for orthopaedic surgery, even though it is known that close range implementations present several problems. Its development introduces a first approach to better understand the difficulties present in this type of implementation.

1.2.3 Challenges

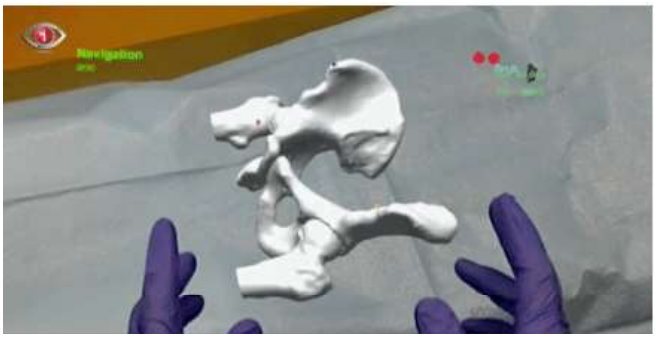
AR was the term created by Tom Caudell in 1990 when he described the display of virtual data in real world vision through a digital display used on aircrafts. The term corresponds to the process of augmenting the perception of the user about the reality that he/she is involved in with additional information. This information can be of various types, with virtual objects being the most well-known option.

(a) Xvision



(b) Opensight

(c) ARSpectra



(d) ARVIS

Figure 1.4: HMDs for medical applications

The rendering of virtual objects in real world can be done through various interface devices, which can be divided into 4 types [12]:

1. Heads Up Displays, categorised by the projection of information on transparent screens in front of the user;

2. Holographic Displays use light diffraction to create three dimensional forms of virtual objects in real space and do not require any gear to be worn by the user, Holovect^{TM} is a good example of this interface;

3. Smart Glasses, which consists in two main types: Optical See-Through (OST), where the user sees the AR overlaid with the real world with transparent lenses, and Video See-Through (VST), where the AR is overlaid with video captured by a camera and the result is seen in the headset display;

4. Handheld/Smartphone Based, where a smartphone or tablet is used for AR experiences;

As discussed, this project will focus on OST due to its relevance in surgical navigation. However, several problems emerge as a result of the necessity of having good perceptual

quality. The vergence-accommodation conflict (VAC) is a very well-known problem when combining AR and VR with HMDs. The vergence of the human eye corresponds to the point where the direction of the eyes converge and the accommodation corresponds to the process of changing the focal length in order to focus an object. Normally, both distances are the same, resulting in the focus of one image part and a blur effect on the surroundings. The conflict occurs when in an HMD the vergence and accommodation distances are different, as demonstrated in Fig. 1.5, resulting in the display of 3D objects within the same eye focus even if they are at different distances. This problem can cause confusion and serious side-effects [13].

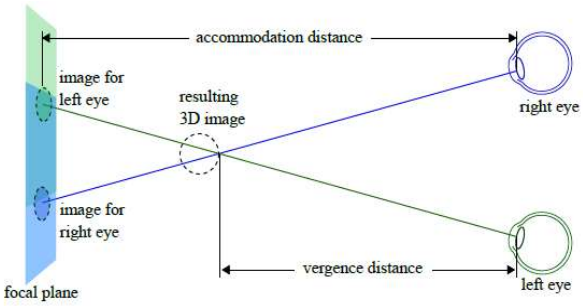


Figure 1.5: Vergence accommodation conflict

Figure 1.6: Foveated rendering from NVIDIA and SMI

For several years a lot of research has been done in this field, which resulted in several solutions. Foveated rendering, developed by NVIDIA and SMI [17], simulates the human eye when rendering objects. This method consist in the application of a filter to objects according to the user eye focus that blurs the objects off focus and maintains the definition of the objects within focus, as shown in Fig. 1.6. This method is mostly applied when eye-tracking is available.

Other solutions are available in the literature [13] that require modifications of the hardware (mostly in the lenses). Sliding Optics is one solution that requires the replacement of static lenses for relay lenses that can be moved by the user, changing the focal length of the virtual image while maintaining the angular FOV.

Calibration constitutes another problematic subject for OST-HMDs due to the lack of proper calibration evaluation metrics. User feedback is the most reliable option to obtain an accurate evaluation but it is not quantifiable. In the literature, most implementations consider an evaluation of the Inter Pupillary Distance (IPD) or the projection matrix [15], [6]. Additionally, recent methods also consider evaluations with repeatability and alignment of virtual cubes in order to retrieve an approximation of the errors present in the calibration [6].

1.3 Objectives and Contributions

The overall objective of this dissertation is to implement a surgical navigation system in the DreamGlass OST-HMD. For this purpose, the in.nav system created by Perceive3D, that currently runs on standard PCs and tablets, was considered. In order to accomplish this, an application was developed to transfer data from in.nav to the HMD's API (Unity). The development focused on improving usability and user-experience, and the main contributions of this thesis are the following:

- implementation of one mode of operation that serves to render a virtual display, creating a hands-free solution;
- implementation of another mode of operation that aims to provide real insertions at close range, by rendering AR directly over the patient's anatomy;
- investigation of the most suitable calibration approach for the DreamGlass HMD, which is required since DreamGlass does not come with an internal calibration procedure. Besides errors in the calibration of the HMD, another important source of inaccuracies and poor user-experience is the VAC. However, the solutions for this problem presented in the literature are complex and often rely on changes in the hardware, being outside the scope of this thesis;
- evaluation of different calibration approaches in a thorough experimental validation that includes intra-subject and inter-subject measurements.

2 Dreamglass

From the several options of HMDs that could be used for implementation in this project, the chosen device is the DreamGlass. In this section, it is described the reason for the option taken, how the headset works and which features are available.

2.1 DreamGlass out of the shelf

Since Google Glasses released to public, various see-through AR products emerged with a wide range of functionalities:

- Microsoft Hololens contributed with several extra functionalities like controls with voice and gestures but its price is too high for this project. Additionally, this HMD does not work well in close-range;

- Magic Leap One has several features including lightweight design, responsive and fast real-time meshing and a 6 DoF controller but is not obtainable in Europe, making the purchase hard and very expensive;

- Daqri has prohibitive cost (more than 10000 euros), resulting in out off range purchase limit for experimental research.

Dreamglass, a low-cost see-through AR system, presented several hardware properties that were required for this project: good quality of display and a wide field of view (FoV). Even though the quality of the RGB camera does not fulfil the requirements needed for this project, this HMD system in general provides the necessary tools without the extra advanced sensors (depth camera, magnetic controller, etc.) that are not need in a prototype implementation. In general, most products have tools not needed at this stage for the price asked, resulting in choosing DreamGlass.

2.2 DreamGlass Hardware

- Display: 2.5 k resolution, 60 Hz, 90-degree diagonal FoV;
- Infrared camera for hand gesture recognition;
- 3 degree-of-freedom head tracking;
- 1080p front-facing RGB camera with 1920x1080 dimension;
- Connectivity: two USB 3.0 ports and one HDMI 1.4 port;

2.3 The DreamGlass Software

DreamGlass requires the installation of Unity since the API is supported by this software. It is advised to use Windows as the operating system since Unity is mostly supported for Windows and Mac, the Linux version is only experimental.

2.4 Performed Adaptations

Specifications of computer and software used:

- Intel Quad Core i7 4700HQ 2.4Ghz
- 8Gb of Memory
- NVIDIA GeForce 840M
- Windows 10
- Unity 2018 3.0
- DreamGlass SDK 3.4 and 3.6

The different components of the glasses were tested and we came to the conclusion that the incorporated RGB camera could not be used due to its high level of noise, which resulted in low detection rate of the markers used by in.nav. Thus, we decided to use an external camera with the appropriate specs required by in.nav, which had to be attached to the glasses. For this, we designed 2 supports for holding the camera using solidworks and printed them. The supports have different inclinations according to their uses:

- One higher inclination (66°) to be used in the mode 1 of the project, the rendering of a virtual display. This consist in the rendering of one virtual plane with display of images generated by in.nav, which is running in the background independently;
- One lower inclination (40°) to be used in the mode 2, the rendering of virtual objects on the anatomy;

To notice that switching between each support requires an additional calibration of the HMD (described in the following chapters) due to the different camera positions. Fig. 2.1 shows the two camera supports, with one already attached to the glasses.

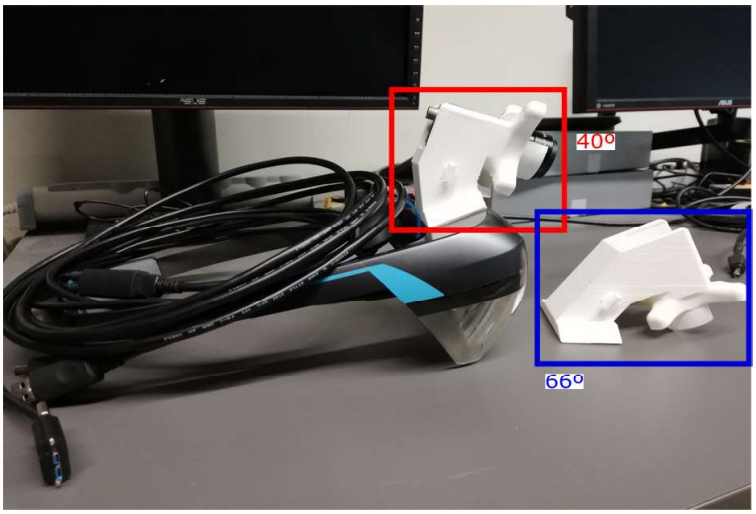


Figure 2.1: Designed camera supports to be attached to the HMD

2.5 Reverse engineering of DreamGlass

In order to develop an HMD platform and have accurate results, it is crucial to understand how the system works. The study of DreamGlass required reverse engineering due to code encapsulation, dynamic libraries with important information that cannot be accessed and no program documentation.

The first tests involved understanding where the variables that characterise the displaying model (one screen for each eye) are defined and how we could change them. These variables are presented in Table 2.1. Using Unity, 3D points were created and their 2D projections on each screen were retrieved using the emulator. This yielded a set of 3D-2D correspondences that helped in understanding the projection procedure used by DreamGlass. Besides this, it was required the dissection of all the classes and their variables in order to understand and formulate the projection method used by DreamGlass.

DreamGlass variables used for rendering	
Variables	Value
$Resolution(x, y)$	2800, 1500
$AspectRatio$	1.8667
$FoV_{diagonal}$	47.45°
$NearPlane(zNear)$	0.1 m
$FarPlane(zFar)$	1000 m
IPD	0.064 m
$Tilt$	10°
$Focallength$	0.05 m
$EyeHeight$	0.15 m
$EyeDepth$	0.1 m
s	-1 or 1

Table 2.1: Variables used in the rendering process

The model consists in two cameras[1] defined in the Unity coordinate system. Each camera is translated half the IPD from the origin along the x-axis, having the left camera a displacement with a positive value and the right camera a negative value. For each camera a perspective projection matrix and a world-to-camera matrix are defined,corresponding to the perspective model presented in Fig. 2.2. The perspective projection matrix defines the intrinsic parameters to be applied (equal to both cameras), creating the frustum that confines the 3D space with all the points that can correspond to a point in the screen, in other words the correlation between virtual world and screen. The world-to-camera matrix consists in the transformation applied to both cameras and the translation of half IPD and a rotation for establishing the relative orientation of the cameras with each other.

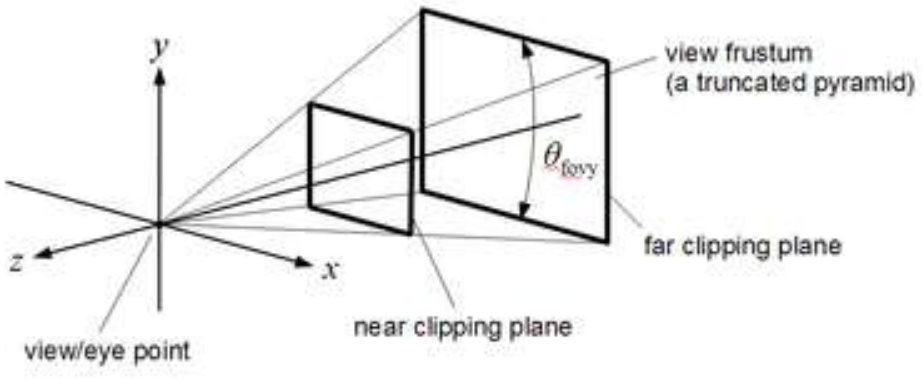


Figure 2.2: Camera frustum [2]

In order to transform a 3D point P_{3d} in homogeneous coordinates, $P_{3d} = [P_x, P_y, P_z, 1]^T$, into a 2D point in screen coordinates the following is done. First, the perspective projection

[1]In this section, camera refers to the Unity camera, not a real camera

H and world to camera T matrices are computed:

$$H = \begin{bmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{zFar + zNear}{zNear - zFar} & \frac{2 \times zNear \times zFar}{zNear - zFar} \\ 0 & 0 & -1 & 0 \end{bmatrix}, \text{ with } f = \cot(\frac{FoV_{diagonal}}{2}) \tag{2.1}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & \frac{s \times ipd}{2} \\ 0 & cos(tilt) & sin(tilt) & 0 \\ 0 & sin(tilt) & -cos(tilt) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Afterwards, the 3D point P_{3d} is transformed into screen coordinates by $P = H \cdot T \cdot P_{3d}$. The obtained point is normalized, yielding $P_n = \frac{P}{P(4)}$, where $P(i), i = 1, .., 4$, is the ith component of P. The screen coordinates in pixels are obtained with the following formula:

$$P_{screen} = \begin{bmatrix} \frac{xResolution}{2} \times P_n(1) + \frac{xResolution}{2} \\ \frac{yResolution}{2} \times P_n(2) + \frac{yResolution}{2} \end{bmatrix} \tag{2.3}$$

In order to measure the possible working volume provided by the headset, a Matlab program was implemented to predict the 2D locations in both left and right screens of 3D points at reasonable distances from the headset.

Currently, considering the size of the markers used by in.nav (15 to 22 mm), good performance is obtain at a distance of up to 1 meter. Also, and since the headset is farther from the anatomy than a handheld camera, the minimum working distance is about 40 cm.

Thus, using the projection model, the 3D working volume was measured by finding which points were projected inside both screens. To give an idea, at 40 cm from the headset the x and y maximum distances are 20 and 10 cm, and at 1 m from the headset, it is possible to work at 40 and 25 cm in x and y directions, respectively.

Fig. 2.3 shows the relevant reference frames considered by the DreamGlass headset, as well as a set of 3D points projected in its screens. Due to the tilt of the lenses, the 3D points, which are symmetric with respect to the z axis, are not centred on the screens.

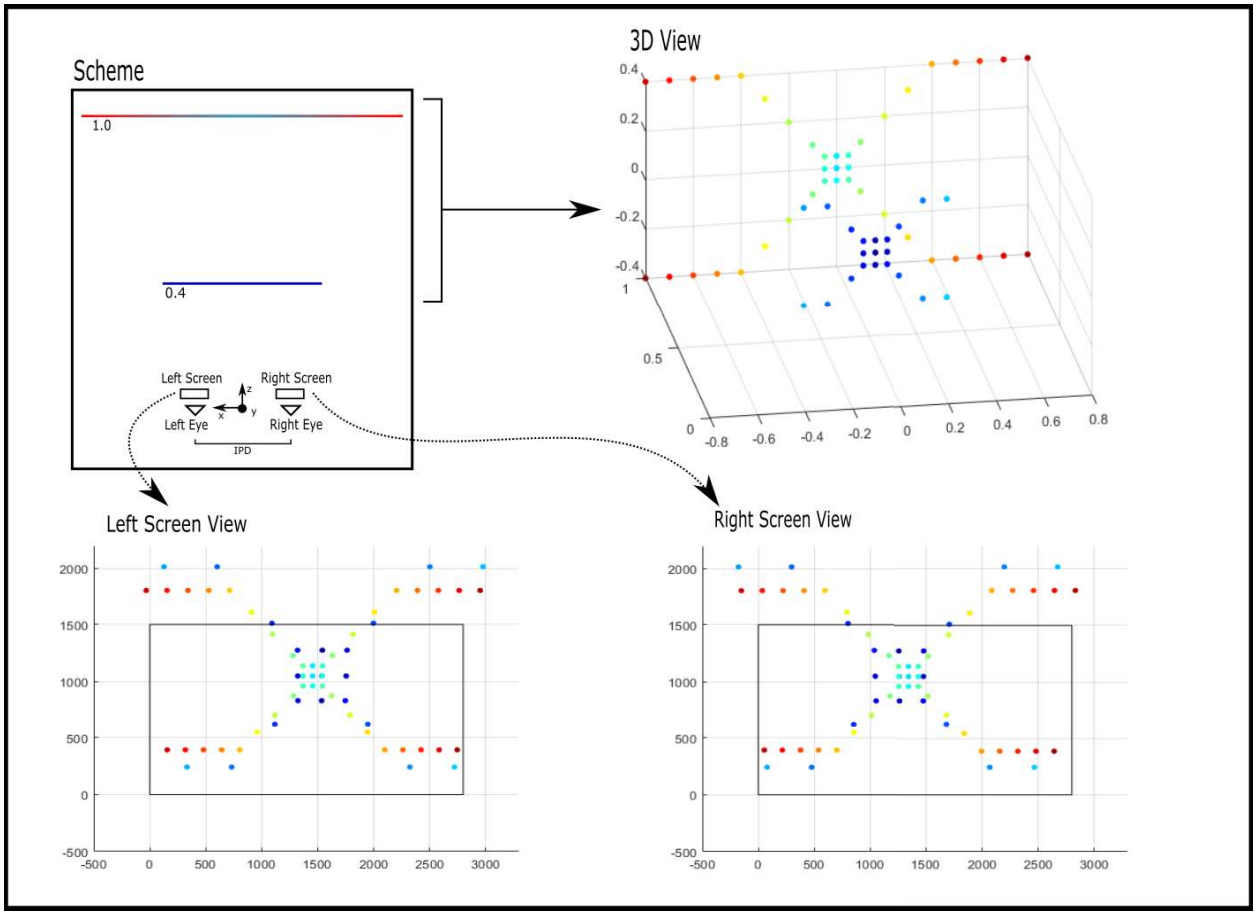


Figure 2.3: Scheme of the correlation between 3D world points and screen points with 10° tilt

2.6 System conversions

Although Unity makes use of the Opengl perpective projection matrix, most of the literature information about calibration (Chapter 5) makes use of a different matrix, K. This matrix corresponds to the intrinsic parameters of the Pinhole Camera model.

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The parameters of K are well known:

- f_x and f_y correspond to the focal length in the x-axis and y-axis, respectively;
- c_x and c_y corresponds to the offset of the principal point location in the image plane;
- γ is the skew, which corresponds to the coefficient obtain from the angle between x and the y axis, α, which is different from zero if the axis are not perpendicular.

In order to convert from one representation to the other,the following conversion was considered [21]:

$$Kogl = \begin{bmatrix} 2 * \frac{f_x}{width} & 0 & 1 + \frac{-2 * c_x + 2 * x_0}{width} & 0 \\ 0 & Y * -2 * \frac{f_y}{height} & \frac{Y * -2 * c_y + 2 * y_0}{height} + Y & 0 \\ 0 & 0 & \frac{zFar + zNear}{zNear - zFar} & \frac{2 * zFar * zNear}{zNear - zFar} \\ 0 & 0 & -1 & 0 \end{bmatrix}, \tag{2.5}$$

with *width* and *height* being the camera screen dimensions and Y the value that defines if the y-axis is up or down: if the y-axis is up, Y has a unitary positive value; otherwise it is negative. This implementation is an approximation of the Pinhole Camera Model to the OpenGL Projection matrix by defining the same width and height of the image plane in the near clip plane dimensions. The extrinsic parameters for both conventions were considered the same and the γ value of K is not taken in account for this approximation.

Additionally, in.nav works with right-hand systems and Unity is a left-hand system. This requires an additional conversion from Right-Hand System to Left-Hand System, which can be done through various ways: changing the sign the x value of the 3D coordinates provided by in.nav, switching the x and z coordinates, etc.. The first option ended up being the option of choice due to the easier implementation not requiring change of axes.

3 Architecture

As previously mentioned, the objective of this thesis is to implement a surgical navigation system in an HMD. In order to obtain high accuracy and a pleasant user-experience, the HMD must be properly calibrated, which is not trivial, especially in the DreamGlass headset due to its hardware and software limitations. DreamGlass does not possess eye-tracking, which would allow the implementation of more accurate and less tedious calibration methods, a proprietary calibration procedure, nor high performance displays and other hardware (as an example, the lenses can be easily bent, introducing undesired errors). For this reason, the development of this project was divided into two parts: mode 1 and mode 2. Mode 1 consists in a simpler implementation that focuses on rendering a virtual display and thus does not require a highly accurate calibration. This mode renders the video provided by in.nav augmented with all the necessary guidance information, and contains several different display options as described in Chapter 4. Mode 2, on the other hand, serves to render models and the necessary guidance information directly over the patient's anatomy, necessitating a proper calibration of the HMD.

The implementation process in each mode was divided into two parts: the in.nav and the HMD sides, which run independently of each other. In order for the two interfaces to communicate, it was required the introduction of a system with TCP/IP protocol that managed all the transmitted data. Fig. 3.1 depicts a user performing the TKA guided procedure on a printed model, showing all the components that are part of the setup.

3.1 Mode 1

The first mode consists in transferring information from in.nav to the glasses API (Unity) and displaying it with different setups: Display with AR or AR/VR, option to zoom in the desired area of interest and change display position with keyboard keys. The connection is made with a TCP/IP socket and the information to be passed will consist in video feed

Figure 3.1: Surgical guidance running on an HMD for performing the TKA procedure

and marker poses. The video feed consists in the video captured by the camera and all the information provided by in.nav overlaid with the video, having the option of zooming in the area where the rendering is occurring. The prediction of the area of interest is calculated using the provided markers pose together with the virtual 3D model of the anatomy and its 3D pose. Additionally, another option was introduced that allows the user to view both the video and the 3D model of the anatomy in VR.

The scheme of this stage is shown in Fig. 3.2, including 2 programs and several modules:

- Camera Driver, which captures images used in in.nav;
- Tracker for marker detection in the image;
- Registrator for computation of 3D model pose;
- OpenGL used in the overlaying of the 3D model with the video captured and other relevant information;
- Computation of ROI corresponding to the area where the rendering is occurring;
- Warp Video module where the video is cropped in order to display only the area of interest;
- Render Video Display for the render of the virtual display and the video.

The main execution line consists in the video being captured by the Camera Driver, which is then transmitted to the OpenGL module for video display and to the tracker for 3D pose computation, at the same time the Registrator calculates the pose of the given 3D model. After rendering the model with OpenGL, the augmented video is displayed in a Monitor and

also captured for transmission to Unity. The video and 3D Pose is streamed with a TCP/IP connection and the 3D Model Pose is read from disk. The calibration data is both given to in.nav and Unity, which is also read from disk. Both 3D Model Pose and Calibration data are read using JSON files and YAML files (YAML is used for in.nav and JSON for unity). After computation of the area of interest in the ROI module, the video is cropped in the Warp module and rendered with the Video Display module.

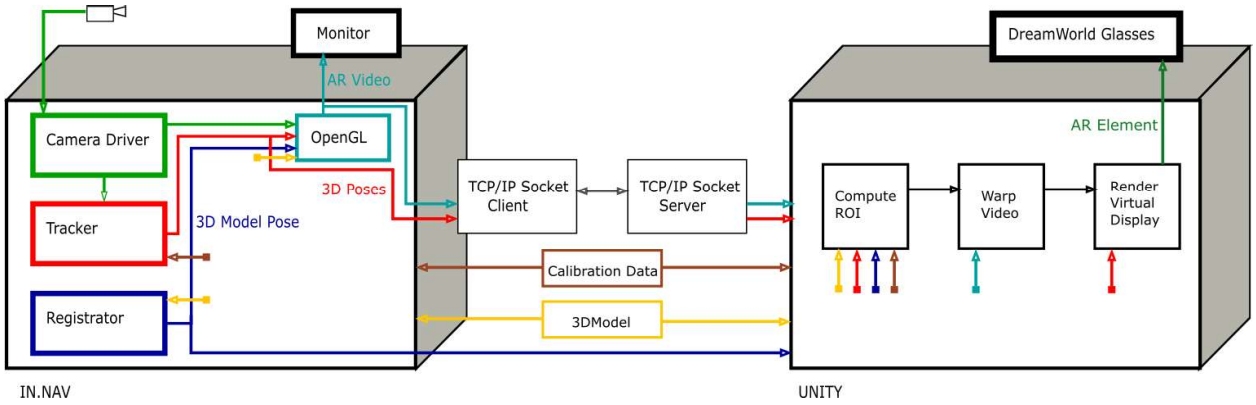


Figure 3.2: Scheme of Mode 1

This Mode improves the overall usability when compared to running in.nav in a tablet or PC:

- No requirement for an extra pair of hands to hold a tablet or PC;
- Zoom Option allows the user to have a better view and control over the displayed information;
- If working with alignments in the 3D model, the option with the VR model allows the user to better align the tool with the model (e.g. in guiding the perforation).

3.2 Mode 2

The second stage consists on enabling the experience in both rendering a virtual display (mode 1) or directly rendering elements over the anatomy (mode 2). It was required additional information from in.nav (positions, angles, etc..) and an extra adaptation for loading models and calibration parameters from several files (in addition to Mode 1 was required the glasses calibration and additional 3D models).

The scheme for Mode 2 is shown in Fig. 3.3. All the inputs are connected with the Navigation Pipeline module where all computations are performed and the guidance information is defined. After processing the information, the rendering occurs in the Renderer and the

AR is directly displayed with the patient's anatomy. Additionally, another mode is available which also includes the option of showing a virtual display with the AR information (Mode 1). The rendering process is locked to one mode and a switch option is available for switching modes.

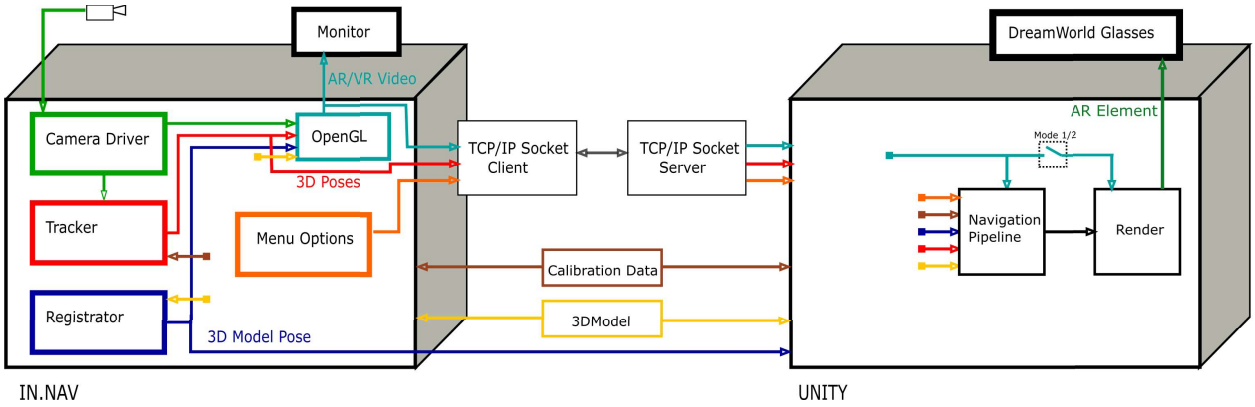


Figure 3.3: Scheme of Mode 2

Implement this mode such that it provides reliable information and a pleasant user experience requires that the calibration of the HMD is accurate. Binocular HMDs require the calibration of two displays, one for each eye, which needs to take into account stereopsis, IPD and other requirements that are not present in a monocular HMD, tablet or PC.

4 Mode 1

This section describes the first stage of the project where Mode 1 was developed and the obtained results.

4.1 Development

As described in Chapter 3, this mode consists on rendering a virtual display that shows the video feed augmented with guidance information obtained from another program that performs surgical navigation. In addition to showing the virtual display, several functionalities were developed in order to enhance the usefulness of the software. Since the virtual display can be moved and more easily manipulated than a physical one (tablet, PC), the features described next were implemented.

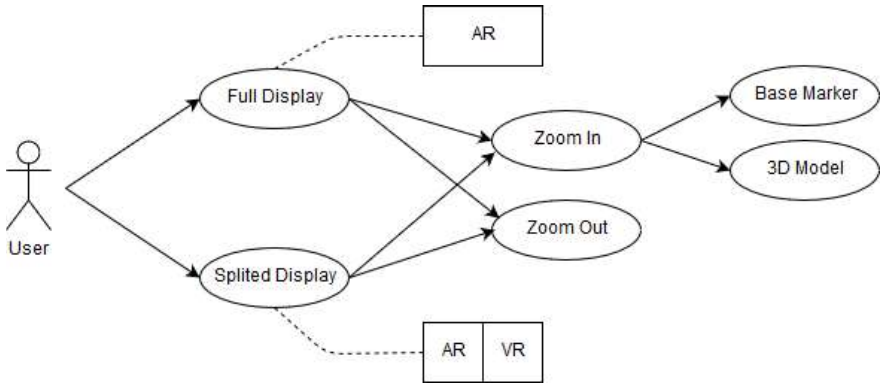


Figure 4.1: Diagram of the general options of the user

Three different sub modes were made for this mode: Simple display, Crop Mode and Side by Side. Since each one has different options and some of these are similar from one submode to another, a diagram of the user choices is presented in Fig. 4.1. The description of the modes is as follow:

- If on full display without zoom, the user is in Simple Display;
- If on full display with zoom, the user is in Crop Mode;

- If on split display, the user is in Side by Side;

Although zoom in and out is available in the split display, this option is only applied to the AR element.

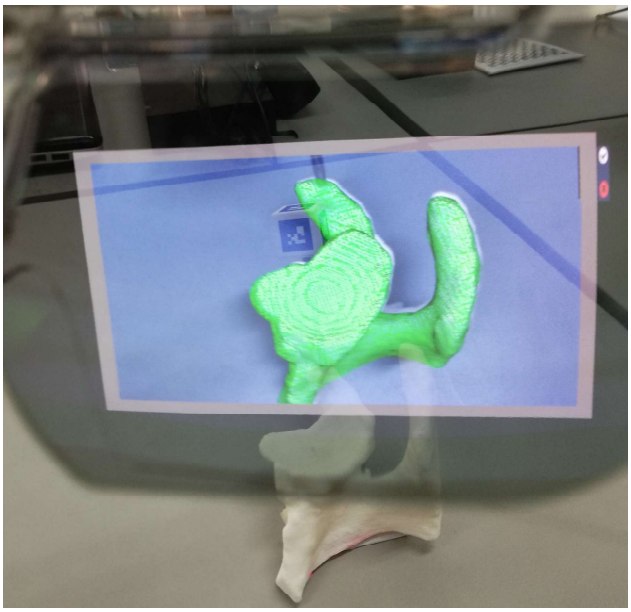


Figure 4.2: User's view from one of the OST-HMD's displays (right side)

Additionally, it was created an extra setup, Fixed Display, in order to render the display near the anatomy (fixed to the marker). This setup allows the surgeon to move the head and maintain the information close to the working area, making it a more interactive experience. This sub-mode is the only requiring a calibration of the HMD, even though it does not need to be very accurate. This step does not only enhance the user experience but also serves as an introduction to the Mode 2. Fig. 4.2 illustrates the user's view from one of the OST-HMD's displays when executing Mode 1.

4.1.1 Simple Display

The first option consists in displaying of the in.nav interface in a virtual monitor. This option allows the user to navigate in the application without the necessary extra pair of hands to hold the tablet or PC and provides all the information generated by in.nav to the user, decreasing the amount of hardware required. The only disadvantage is the display of all the frame, which comes with the cost of displaying also unnecessary information. The view of the OST-HMD is shown in Fig. 4.3.

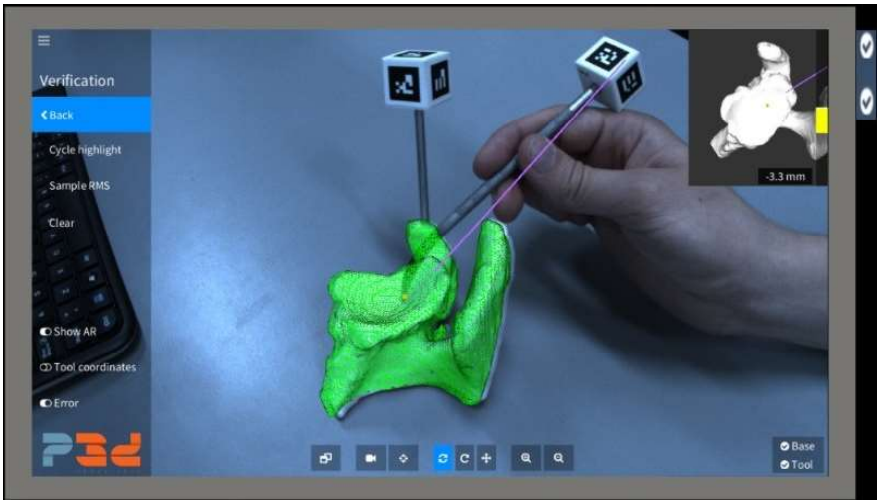


Figure 4.3: User's view from the Simple Display mode

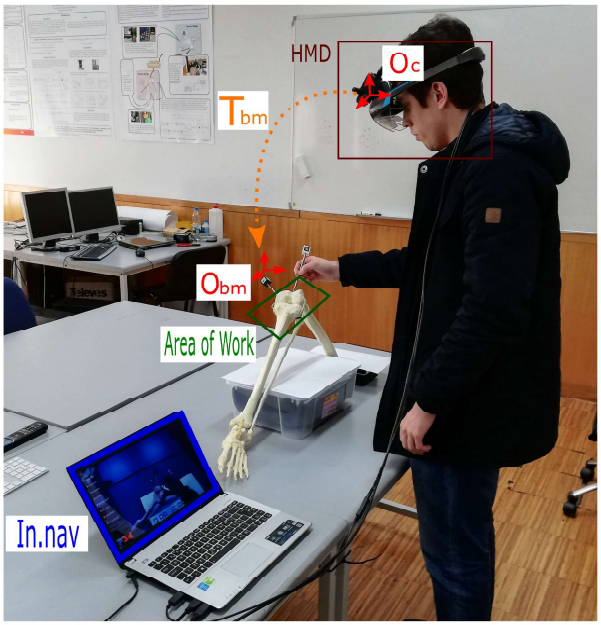


Figure 4.4: The expected Scheme and the transformation map

4.1.2 Crop Mode

A second mode is available, which makes use of the video feed coming from in.nav, and crops the image in order to zoom in the area of interest.

Determining the area of interest can be done in different ways, such as by using the base marker's position, the 3D model position in the image or the location of the touch probe. Since most of the time the focus of the user is either directed to the base marker or the 3D model, two options were created: zoom in in the base marker or in the 3D model displayed on the screen. An additional keyboard key was used to allow the user to switch between each option.

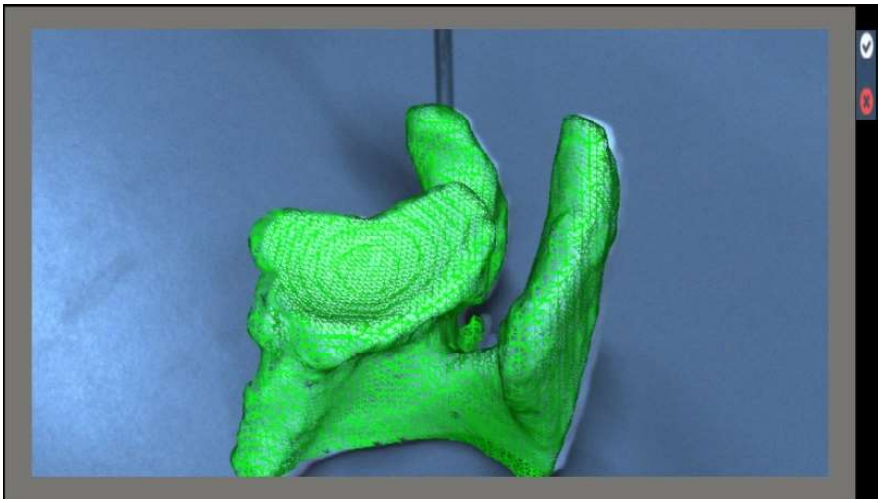


Figure 4.5: User view from the Cropped mode

The area of interest is computed by retrieving the object's (base marker or 3D model) center position in the in.nav screen (the 3D model geometric center is hard-coded). The image around this point is cropped, having the object in the center of the display. The coordinates of this point are obtained by transforming a 3D point P in the base marker coordinate system O_{bm} to the camera reference frame O_c using the pose T_{bm} provided by in.nav (refer to Fig. 4.4). The obtained point is then projected on the image as described in [8].

If the user wishes to zoom in the origin of the base marker, P has coordinates $P = [0, 0, 0]^{\mathsf{T}}$. On the other hand, if the zoom in is to be done on the center of the 3D model of the anatomy, another transformation T that maps points in model coordinates to the base marker's reference frame is used. This transformation is estimated by in.nav in the 3D registration stage. In this case, P is estimated by $P = T * P_{model}$, with P_{model} being a point selected from the virtual model.

The usefulness of the crop mode in the OST-HMD comes from the fact that the camera is considerably far from the anatomy and thus it is crucial for a proper visualization of the area of interest, as shown in Fig. 4.5.

4.1.3 Side by Side

Since in.nav has two main display modes (one using AR and another using VR), a third option is provided that splits the virtual display into two displays. On the left display is the AR element of in.nav with the overlay of the 3D model and the necessary information for the user (including the crop mode). On the right display is the VR element, with the 3D model and some measurements, that serves to guide the user during the surgical procedure.

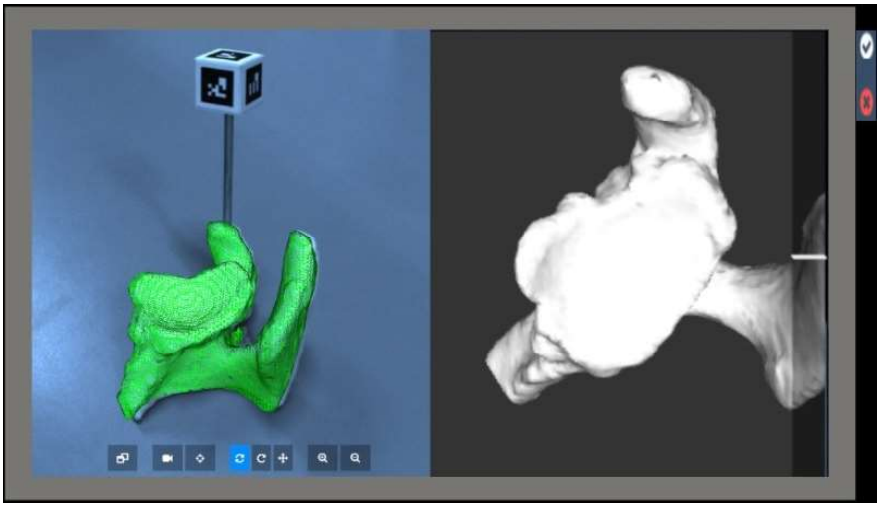


Figure 4.6: User view from the Side by Side mode

The AR and VR elements are justified by the fact that the VR element adds an easier and more clear view of the guidance measurements, allowing the user to view some of the crucial information with less effort. Although VR provides useful information for guidance, the AR element is always necessary since the visual guide is built for overlay with the real anatomy. The user view of this mode is present in Fig. 4.6 and Fig. 4.7 (with Guidance).

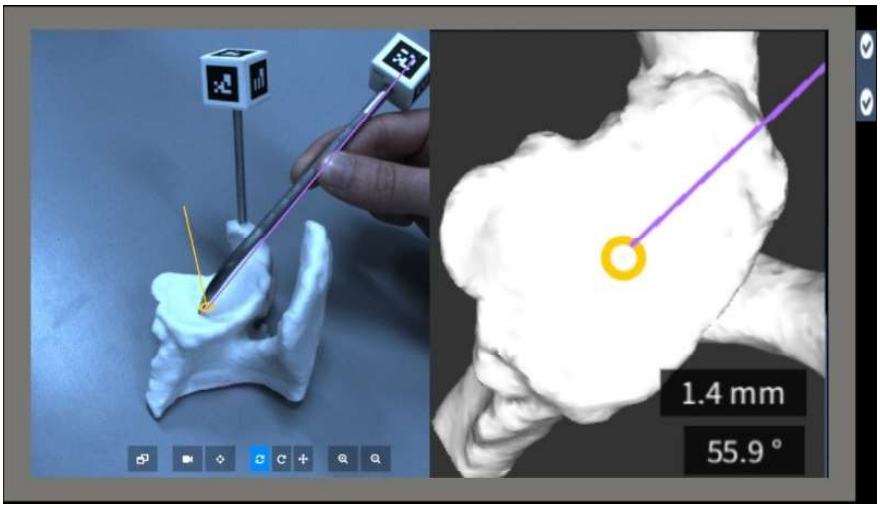


Figure 4.7: User view from the Side by Side mode with Guidance

4.1.4 Fixed Display

The last experiment consisted in rendering the display according to the world position of the base marker. Since the position of the display did not require high accuracy, a simple method that consisted on the alignment of a virtual cube with a physical cube was used as the calibration approach, similar to the method presented in [6]. A brief explanation of this method is given in the next chapter 5.

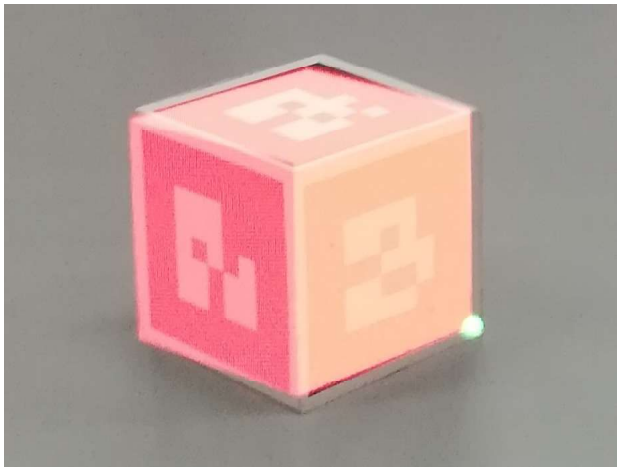


Figure 4.8: Calibration process with the alignment both cubes

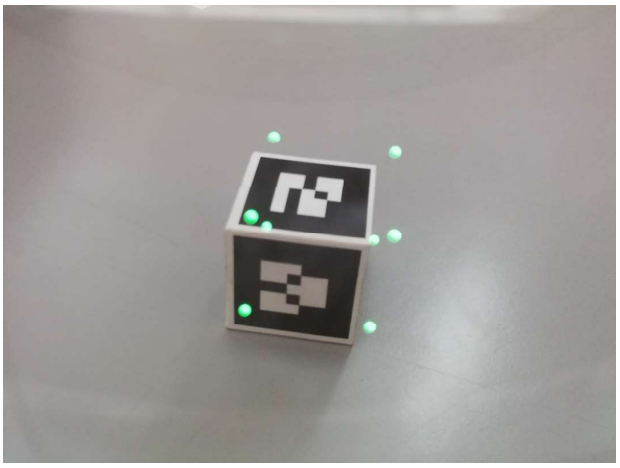


Figure 4.9: Results of calibration viewed from the right eye

For this calibration, a virtual cube (with known position in the glasses coordinate system) and a physical cube (tracked marker in the RGB camera coordinate system) were used, and the aim was to align both cubes (Fig. 4.8). The vertices of each cube in each coordinate system were determined and Horn's method for the estimation of a rigid transformation from 3D-3D correspondences was used. The obtained transformation T_c relates the tracker camera and the glasses coordinate systems.

In each frame, the 3D coordinates P_u of the cube in the glasses reference frame is computed by $P_u = T_c * S * T_{bm} * P$, with S being the matrix that converts between right-hand and left-hand systems (Eq. 4.1), T_{bm} being the transformation matrix between the marker (O_{bm}) and the tracker camera (O_c) and P being the 3D point in the cube system (Fig. 4.4). In this setup it was considered the point $P = [0 \quad 0 \quad 0]^{\mathbf{T}}$.

$$S = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Results

Several calibrations were tested by rendering virtual spheres on the vertices of the physical cube. The results presented in Fig. 4.9 show a slight deviation from what was expected, even after several calibration attempts. Despite not being highly accurate, this result is sufficiently good for positioning the virtual display close to the base marker.

5 Mode 2

5.1 Development

As described in Chapter 3, this mode comprises the options of both rendering a virtual display and guidance information directly overlaid with the patient's anatomy. A detailed scheme of software modules considered in the development of this mode is presented in Fig. 5.1.

In order to allow the development of software to expand and keep track of the different parts, three models are created according to the different functionalities:

- Main_controller is constituted by the main part of the program, it receives the information from in.nav and distributes it to the other models through shared memory, additionally it also loads the functions created on the dll files;
- Display_Menu is constituted by the several classes and functions that allow the virtual display to move and have the different options, Mode 1;
- Model_Menu constitutes the rendering of models, Mode 2.

The full project developed in the Unity API is available in [1].

To implement Mode 2, Model_Menu needed to be developed and some changes were required to the already existing I/O:

- Add an additional TCP/IP connection that allows in.nav to send additional information, more specifically the distances and angles required to implement a guidance sub-mode;
- Load files with different information that allows the software to require less tasks to update parameters, thus improving quality of life: in.nav camera calibration, HMD calibration, registration of virtual model with the patient's anatomy.

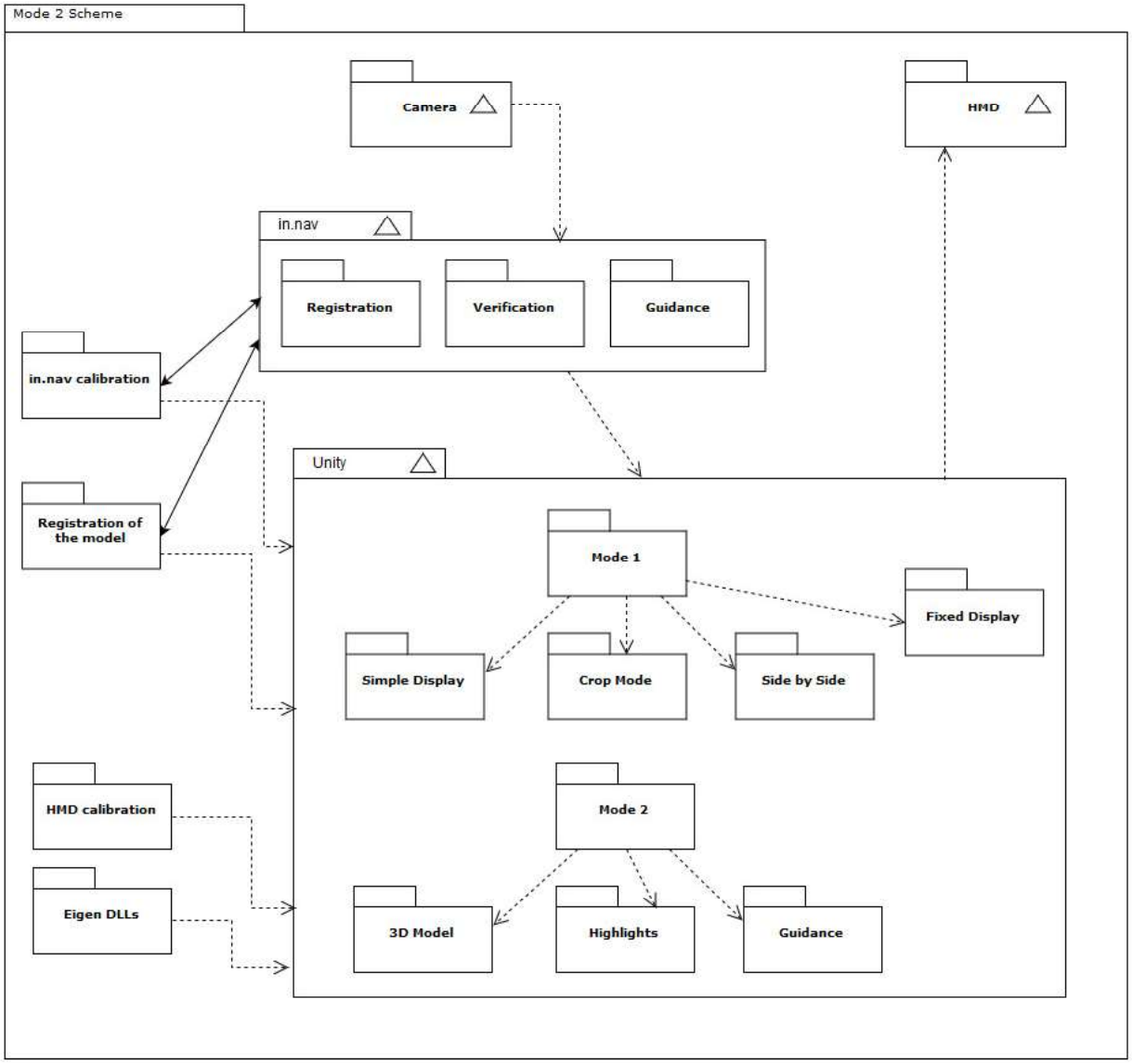


Figure 5.1: Scheme of the modules present in Mode 2

Three different sub-modes for the display of 3D models are available. The first sub-mode consists in displaying the 3D model, the second sub-mode displays the highlights of the 3D model and the last sub-mode implements the guidance option. Guidance is available in in.nav and is also implemented in Unity consisting in the alignment of a perforation tool with a specific vector defined pre-operatively in the 3D model's reference frame. Depending on the different positions and angles of the perforation tool, the color of the rendered vector changes in order to provide real-time guidance information to the user. If the alignment is not correct the color is red, if it is in place but with the wrong angle the color becomes yellow, and in place and correct angle, the color is green. If the tool is not touching the model it stays blue.

5.2 Calibration

5.2.1 Background

For the general case, most HMDs come with generic calibration parameters for the common user which is not viable for achieving the high levels of accuracy required by surgical navigation. In order to obtain more accurate results and adapt the specific parameters for each user, a calibration procedure is required. This section describes the most relevant calibration procedures in the literature.

Calibration in OST-HMDs can be divided into types: manual, automatic and semi-automatic. The difference between these methods is related to the user interaction during calibration [11]. Manual calibration requires the user to perform manual tasks every time the HMD is used. Semi-Automatic calibration corresponds to the subset of methods that try to update the calibration parameters, requiring less tasks when the same user is re calibrating the system. Finally, Automatic calibration corresponds to the group of methods that do not require user intervention, mostly by integration of an eye-tracker. Since DreamWorld AR glasses have neither an eye-tracker system nor a proprietary calibration procedure, the following manual calibration methods were considered in an attempt to achieve satisfactory accuracy in the integration with in.nav.

SPAAM

The first effective manual calibration for OST-HMD without the need for specific hardware was presented in 2000 [24]. This method, SPAAM, until today is one of the most used methods for monocular HMD. It consists in the alignment of several display points (at least 6) with known 3D positions obtained by a tracker. The method estimates the projection matrix for the eye with 2D-3D correspondences using a DLT method. Since SPAAM was initially developed for monocular HMDs, the first test for binocular HMD was presented in [10]. This method consisted in applying SPAAM for each eye, yielding good results except for the accuracy in terms of depth.

StereoSPAAM

More recently SPAAM has been adapted for stereo HMDs [15]. This method divides the nonius reticle used for calibration into parts, each part to be displayed to each "eye". This method allowed not only the user to retrieve points for each eye at the same time but

also produced better results than applying SPAAM since it mitigates the depth problem. According to tests performed [15], Stereo SPAAM has less deviation than SPAAM.

MPAAM

Another adaptation is the MPAAM method that utilises the same approach of SPAAM but instead of doing alignments one by one, it aligns multiple points at the same time. Despite being faster than SPAAM, this method increases the error in the calibration parameters.

Another method has followed the same path and applied the same principle of MPAAM for a binocular HMD for estimation of the transformation between HMD and tracking system without taking in account the intrinsic parameters [6]. This method consists in the alignment of a virtual cube with a physical cube. The position of the virtual cube is known in the virtual space and the physical cube is tracked with a camera and managed by the user.

Posture and type of tracking

Besides the number of correspondences and the method to estimate the camera parameters, it is also required to take into account the user posture (sitting or standing) and point acquisition procedure, which can influence the error introduced. When implementing the calibration procedure, the 2D-3D correspondences can be obtained with two options: user and environment distribution. User distribution consists in acquiring points using a tool that is managed by the user, environment distribution requires the user to align the HMD with a static tracked object [3]. Although environment distribution should result in better calibration results due to the less human error interference, it was concluded in literature [3] that user distribution ends up getting the best results. Further tests also show posture when executing calibration makes little difference to the results.

5.2.2 Experiments

From the previously described methods, SPAAM was the first calibration approach selected for testing. This decision was based on two facts. Firstly, the intrinsic parameters provided in the factory calibration could be inaccurate, leading to the necessity of re-estimating them, which was not possible with the calibration procedure employed in Mode 1. Secondly, MPAAM reports low accuracy and Stereo-SPAAM is a very time-consuming procedure, not as easily implemented as SPAAM as it requires stereo 2D correspondences to be generated, and not simple 2D points.

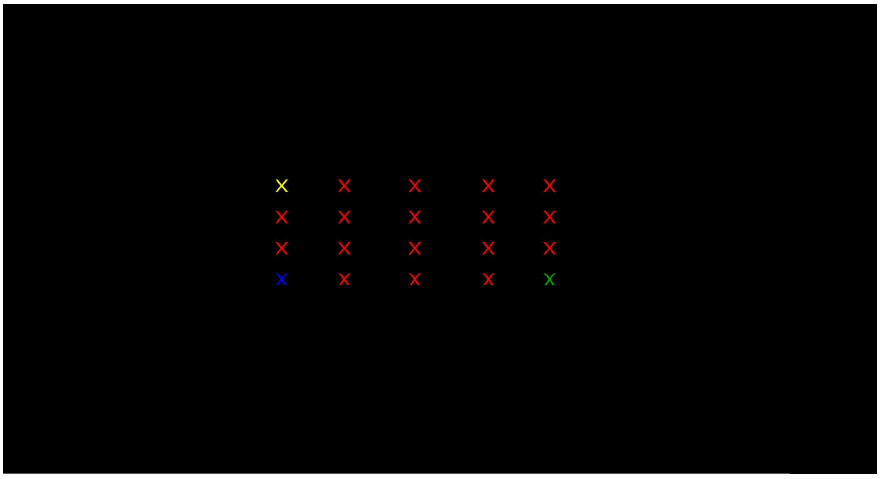


Figure 5.2: Calibration image

In order to implement the SPAAM method, the image shown in Fig. 5.2 was created and rendered in each of the HMD's screens. Then, the user was instructed to align the tip of the tool shown in Fig. 4.7, whose marker position is determined by in.nav, with each cross, generating 2D-3D correspondences. These were used to estimate a projection matrix, which was then decomposed into a rigid transformation and an intrinsic parameter matrix.

Since SPAAM is applied to each eye independently, two distinct and unrelated calibrations are obtained. SPAAM yields one 4×4 extrinsic parameters matrix T_{side} and one 3×3 intrinsic parameters matrix K_{side} for each eye, with $side = l, r$ for the left and right eyes, respectively. One can make some assumptions regarding the calibrations of the left and right eyes: the intrinsic parameters are the same, i.e., 3D points are projected into 2D points in a similar manner for both eyes, and the relative pose between the left and right eyes T_{lr} is defined by a rigid transformation with the format

$$T_{lr} = \begin{bmatrix} 1 & 0 & 0 & \lambda \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.1}$$

where λ is the IPD for a specific subject. This assumptions motivated the creation of three new calibration approaches that are based on the standard SPAAM method:

- Apply SPAAM for each eye, estimate an average intrinsic parameters matrix $K_{avg} = (K_l + K_r)/2$ and re-estimate the extrinsic parameters T_l and T_r (12 DOFs) by forcing $K_l = K_{avg}$ and $K_r = K_{avg}$. This method is referred to as SPAAM$_{avg}$;

- Apply SPAAM for each eye, set $K_l = K_{fact}$ and $K_r = K_{fact}$, where K_{fact} is the intrinsic

parameters that came with the HMD and re-estimate the extrinsic parameters (12 DOFs). This approach is called SPAAM$_{fact}$;

- Apply SPAAM for each eye, set $K_l = K_{fact}$, $K_r = K_{fact}$, and T_{rl} as defined in Eq. 5.1, and estimate T_l and λ (7 DOFs). This approach is named SPAAM$_{fixed}$.

The original SPAAM method applied to each eye estimates a total of 17 DOFs, 6 corresponding to each rigid transformation and 5 corresponding to the intrinsic parameters.

An additional problematic subject related with OST-HMD calibration is the metric for calibration accuracy since both images of the displays are rendered in the user brain, which causes several problems to classify how accurate it is. Two methods [6] were used to perform this evaluation:

1. Perform two or more calibrations to the HMD and compare the different T_l and T_r matrices obtained by estimation, allowing the evaluation of repeatability.

2. Display several virtual cubes with fixed position to the base marker (tracked object) in the HMD system (unity) and align another tracked cube with these cubes, measuring the error present in the projection.

Even though the second approach allows to retrieve some quantitative information about the calibration error, it is still not ideal because it is error-prone as errors are introduced in the manual alignment process.

Since the described calibration methods to not model factors such as eye movement or distance of focus, the alignment with virtual cubes described in metric 2 was performed in 3 stages:

1. The virtual cubes are rendered near the tracked base marker, allowing the alignment to be performed in the central area of the screens. This metric allows to quantify how precise the calibration is in the area where the user focus is during most of the procedure.

2. The virtual cubes are rendered in the peripheral areas and the user must perform the alignment without moving the head.

3. The virtual cubes are rendered as in 2. but the user can move freely in order to perform the alignment.

Figure 5.3: Virtual Cube alignment

In all cases, 6 virtual cubes were considered, as depicted in Fig. 5.3.

In addition to these two metrics, the reprojection error for each calibration attempt is computed, giving an intuition on how well the acquired data matches the calibration models.

5.2.3 Calibration Results

In order to assess how the calibration accuracy varies between subjects, 3 subjects were chosen to acquire calibration datasets. Each calibration consisted in the user performing 40 alignments, 20 for each eye. Subjects 1 and 2 acquired 6 datasets and subject 3 acquired 5 datasets, allowing to evaluate the repeatability of the different methods.

The remainder of this section shows the results obtained for each subject using the evaluation metrics described previously.

Reprojection Errors in Estimation

The results obtained for subject 1 are given in Fig. 5.4 and Fig. 5.5. It can be seen that across different datasets, the reprojection errors do no vary significantly, and that they present median values around 5pix. For subject 2, results (Fig. 5.6 and Fig. 5.7) are also similar for

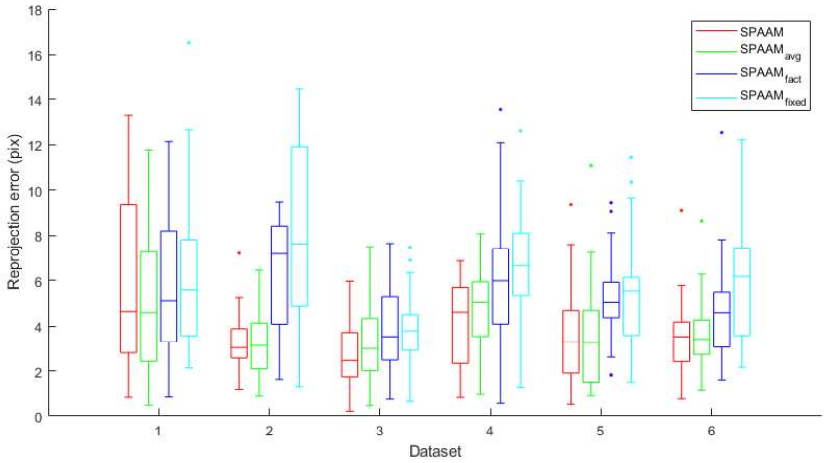


Figure 5.4: Reprojection errors for subject 1 (left eye). Each boxplot contains 20 elements, corresponding to 20 manual alignments.

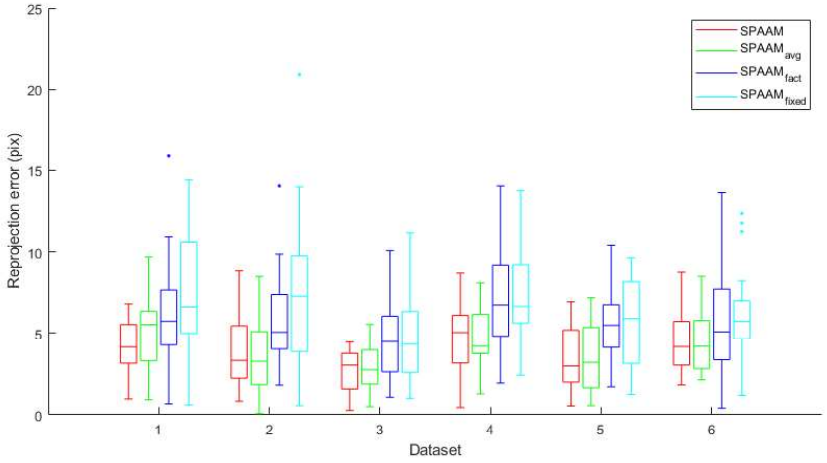


Figure 5.5: Reprojection errors for subject 1 (right eye). Each boxplot contains 20 elements, corresponding to 20 manual alignments.

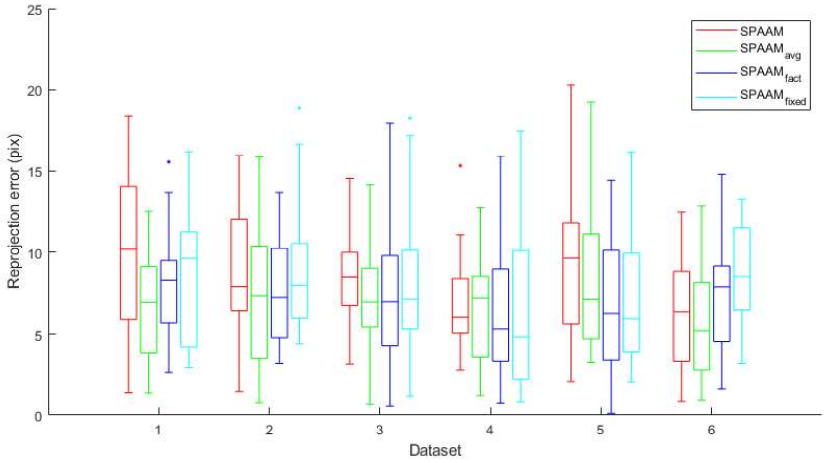


Figure 5.6: Reprojection errors for subject 2 (left eye). Each boxplot contains 20 elements, corresponding to 20 manual alignments.

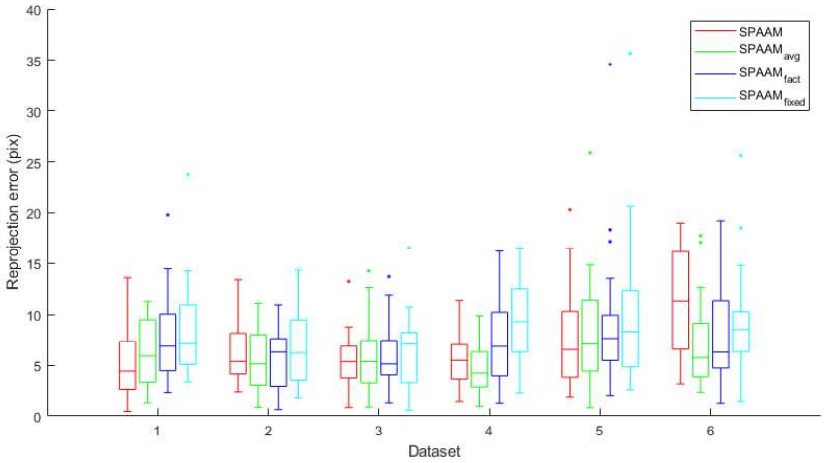


Figure 5.7: Reprojection errors for subject 2 (right eye). Each boxplot contains 20 elements, corresponding to 20 manual alignments.

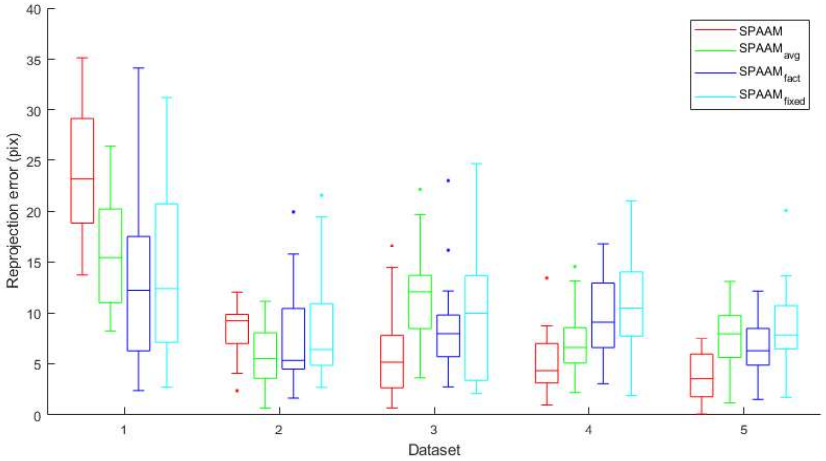


Figure 5.8: Reprojection errors for subject 3 (left eye). Each boxplot contains 20 elements, corresponding to 20 manual alignments.

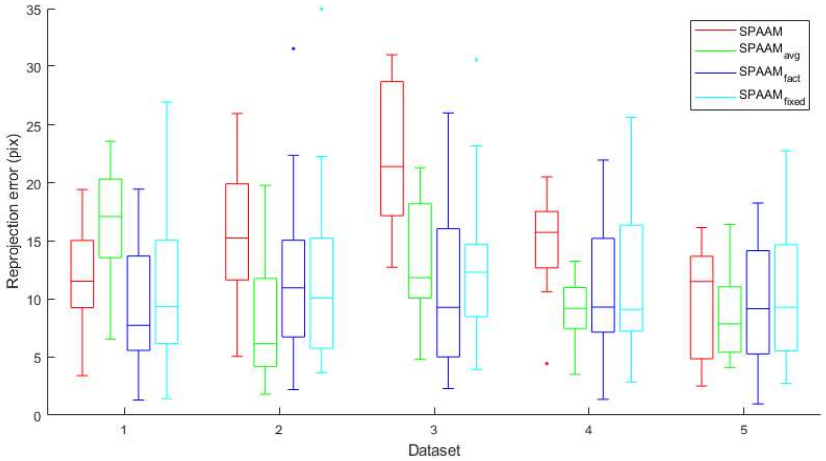


Figure 5.9: Reprojection errors for subject 3 (right eye). Each boxplot contains 20 elements, corresponding to 20 manual alignments.

the different datasets but reprojection errors increased, specially for the left eye. For subject 3 (Fig. 5.8 and Fig. 5.9) reprojection errors for the initial datasets are significantly higher, decreasing in the last calibration attempts. This behavior suggests that there is a learning curve, with the first dataset acquisitions being more difficult to execute. Overall, median reprojection errors are under 15pix, which corresponds to about 0.5% of the HMD screen diameter.

Results obtained for Subject 1 present lower and more consistent reprojection errors due to the fact that he is the most experienced user. Thus, his results can be considered for comparing the 4 different calibration methods without the interference of user inexperience. In both Fig. 5.4 and Fig. 5.5 it is observable that the higher the amount of parameters forced, the higher is the reprojection error (median). When comparing factory parameters (factory and fixed) to the average, it is also possible to conclude that forcing factory parameters increases the reprojection error. This fact means that this subject has calibration parameters different from the ones hardcoded on the HMD, which can be expected since each user has his/her own parameters.

Transformation Error in Estimation

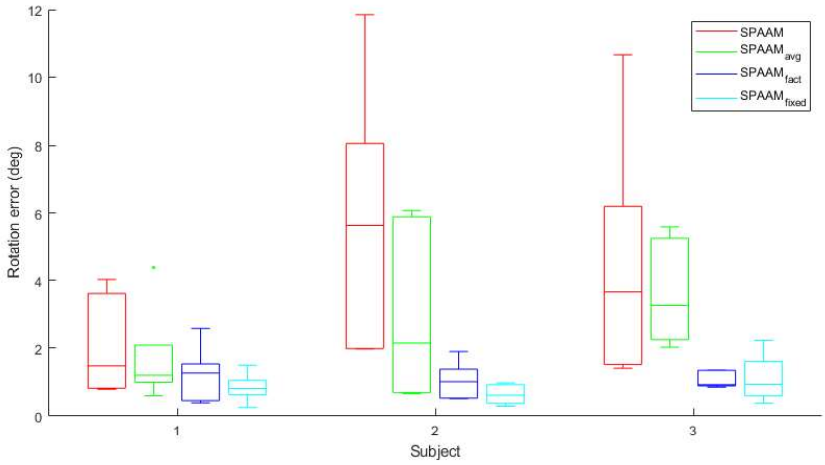


Figure 5.10: Rotation error for $T_{errorMed}$, 6 datasets per method with subject 3 having 5

In order to assess the repeatability of the calibration methods, the obtained rigid transformations T_l were compared to their median T_{med} by computing the error matrix $T^i_{errorMed} = T^{-1}_{med} * T^i_l$, with i being the index of the dataset, and retrieving the rotation and translation components of $T_{errorMed}$. The distribution of these errors is shown in Figs. 5.10 and 5.11 and it can be clearly seen that by reducing the number of DOFs leads to a higher consistency across calibration results. Concerning SPAAM$_{avg}$ and SPAAM$_{fact}$, the same number

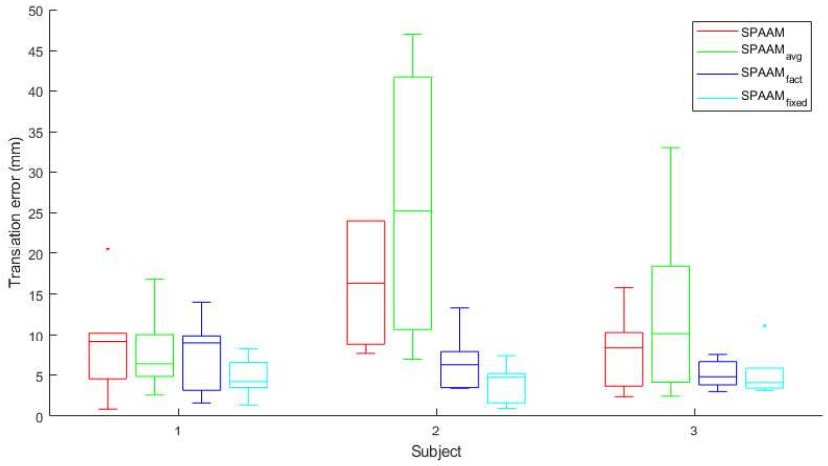


Figure 5.11: Translation error for $T_{errorMed}$, 6 datasets per method with subject 3 having 5

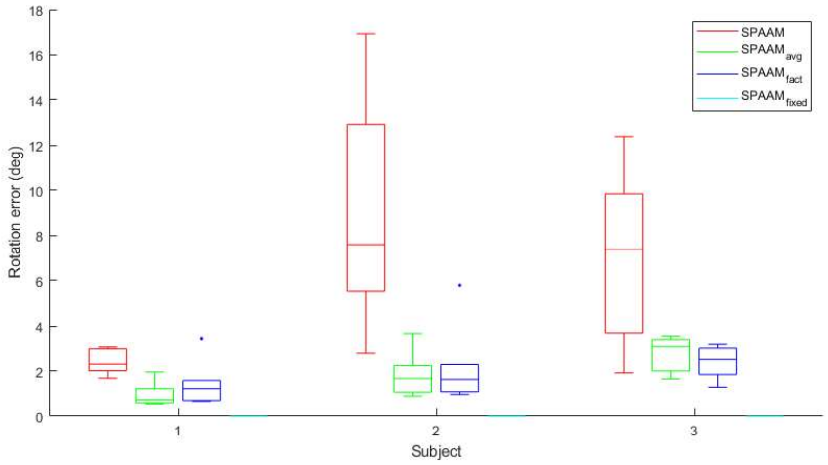


Figure 5.12: Rotation error between the obtained and the theoretical pose between eyes, 6 datasets per method with subject 3 having 5

of parameters are fixed but while in SPAAM$_{avg}$ the intrinsic matrices K_l and K_r vary between calibration attempts, in SPAAM$_{fact}$ they are always the same (equal to K_{fact}). The significantly lower error values observed for SPAAM$_{fact}$ mean that the obtained matrices T_l are much more similar between each other. The observation that letting the intrinsic parameters free leads to significantly different calibration results indicates that the problem is overparametrized and it may not be adequate to estimate 17 DOF (as in standard SPAAM) with the provided amount of data.

One conclusion reported in the literature is that although SPAAM tends to provide good calibration results, it lacks accuracy in estimating the depth component. In order to confirm this fact, another experiment was performed. For each calibration result, the relative pose

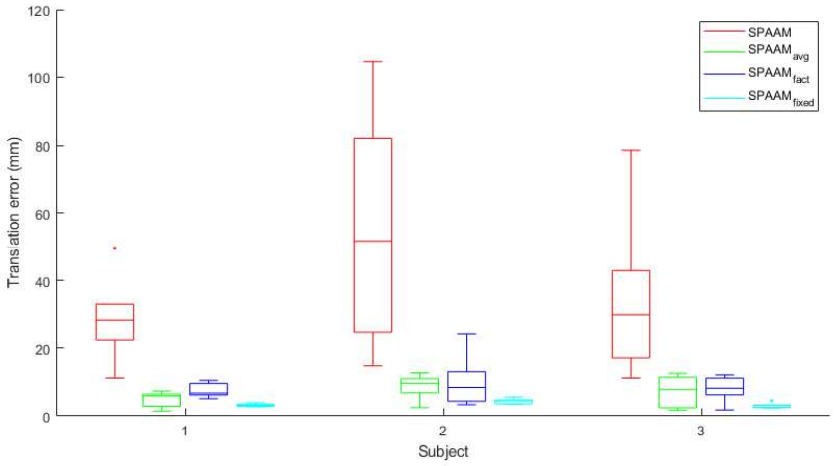


Figure 5.13: Translation error between the obtained and the theoretical pose between eyes, 6 datasets per method with subject 3 having 5

between both eyes T_{lr} was estimated and compared with the theoretical pose

$$\Delta T_{gt} = \begin{bmatrix} 1 & 0 & 0 & IPD \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

by estimating the rotation and translation errors of matrix $T_{error} = \Delta T_{gt}^{-1} * T_{lr}$ as previously described. Results are given in Figs. 5.12 and 5.13. It can be seen that as more parameters are fixed, the relative pose T_{lr} approaches the theoretical pose between eyes. This result, together with the one in Figs. 5.10 and 5.11, shows that the obtained calibration results are plausible, being close to the theoretical ones, and this closeness increases by fixing parameters to acceptable values.

In order to better understand if the depth component is being poorly estimated, the x, y and z components of the translation vector of T_{error} are shown in Figs 5.14, 5.15 and 5.16, respectively. For the case of SPAAM, it is easily noticeable that the z component (that represents depth) presents the highest error. These experimental results are inline with the literature, which refers poor depth estimation as one of the drawbacks of SPAAM when applied to binocular HMD. However, fixing the intrinsic parameters largely solves this problem, which is the main advantage of the time-consuming method StereoSPAAM.

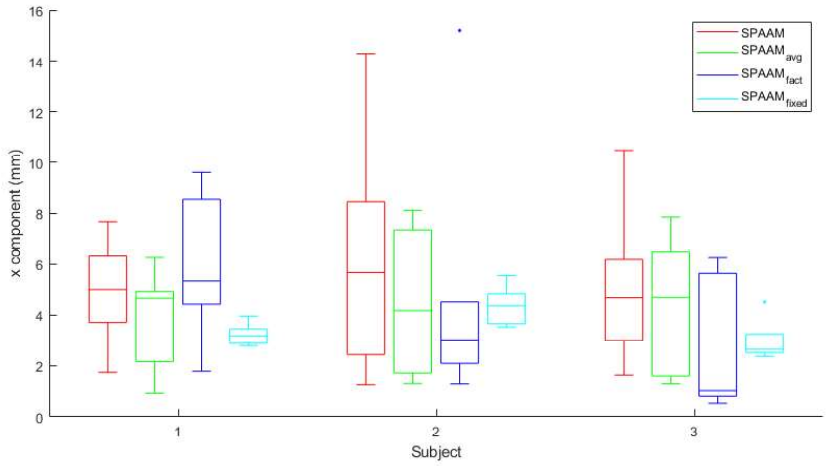


Figure 5.14: Translation error in the x component of T_{error}, 6 datasets per method

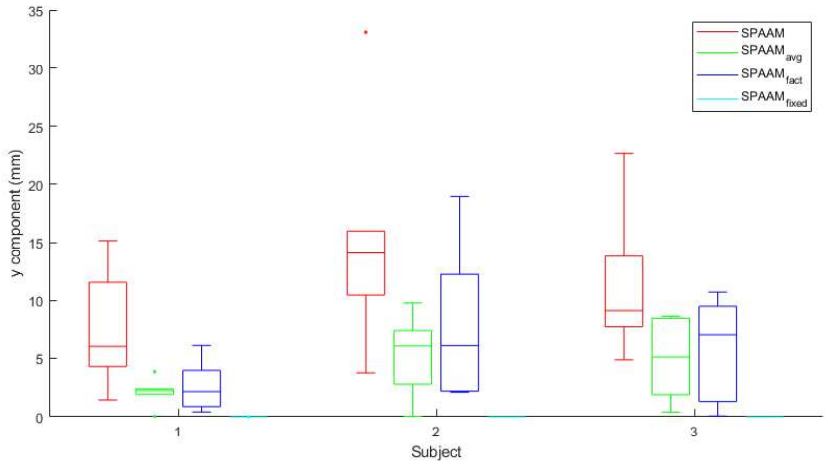


Figure 5.15: Translation error in the y component of T_{error}, 6 datasets per method

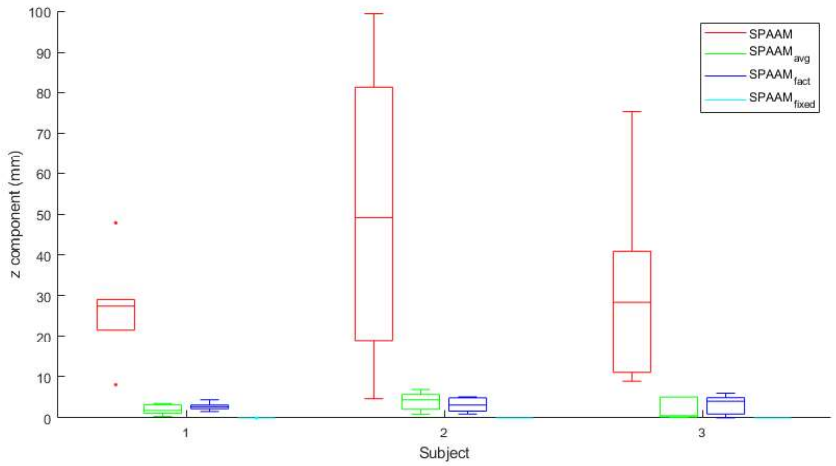


Figure 5.16: Translation error in the z component of T_{error}, 6 datasets per method

Transformation Error in alignment verification

Since subject 1 presented the most consisting results and in order to simplify, 3 datasets where chosen randomly from subject 1 to proceed with the alignment of the virtual cube. The results are presented in Fig. 5.17 and Fig. 5.18, which show slightly better results for $SPAAM_{fixed}$ and $SPAAM_{fact}$.

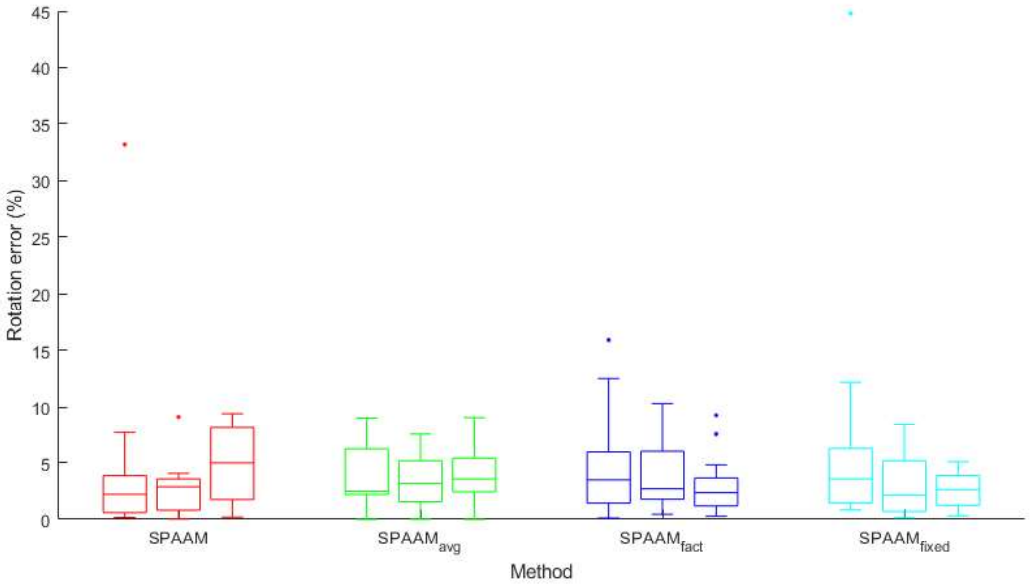


Figure 5.17: Rotation Error in Virtual Cube Alignments, for each method the 3 boxplots correspond to 3 different calibration datasets

Fig. 5.19 shows the error per type of alignment in rotation while in Fig. 5.20 is shown the error for translation. As shown in the results, alignments in the center have the smallest errors while alignments in the periphery get the largest errors. These outcomes are inline with what was expected since SPAAM does not take in account the rotation of the eyes, which results in a degradation of the calibration if the user is looking to the periphery of the HMD. Free movement errors slightly improve from periphery errors due to the user being able to look at the virtual cubes directly, without significantly shifting his/her eyes. Even though these results confirm what was expected, this pattern was not so evident in some of the datasets because the acquisition process is highly error-prone.

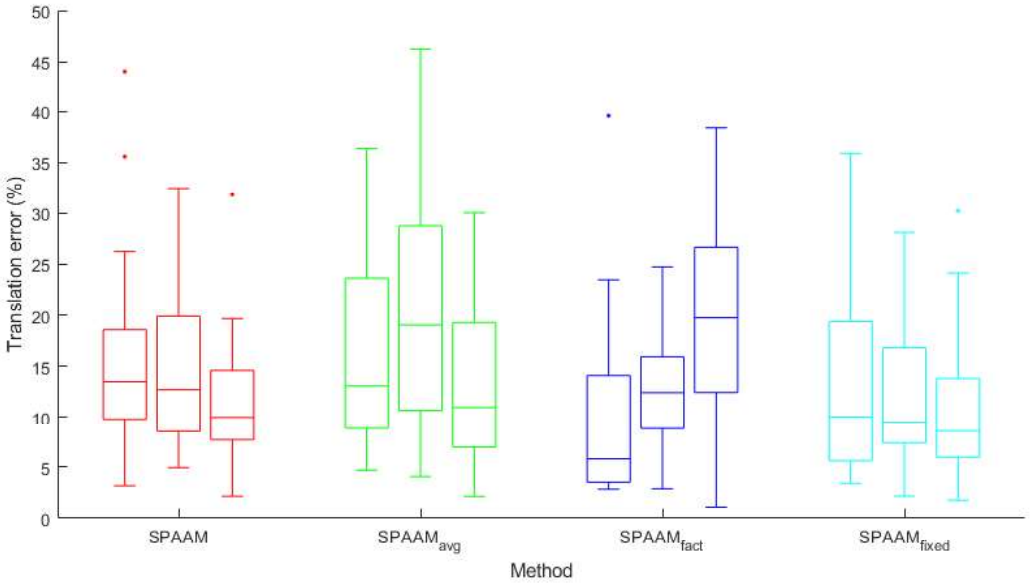


Figure 5.18: Translation Error in Virtual Cube Alignments, for each method the 3 boxplots correspond to 3 different calibration datasets

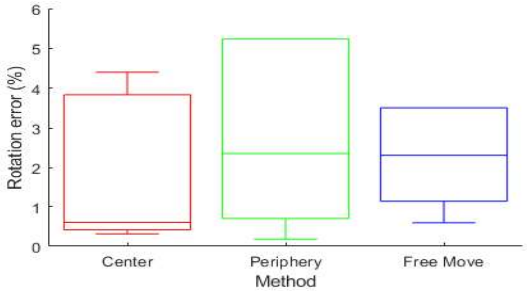


Figure 5.19: Rotation error in the virtual cube alignment with SPAAM for the first dataset, each boxplot corresponds to 6 alignments

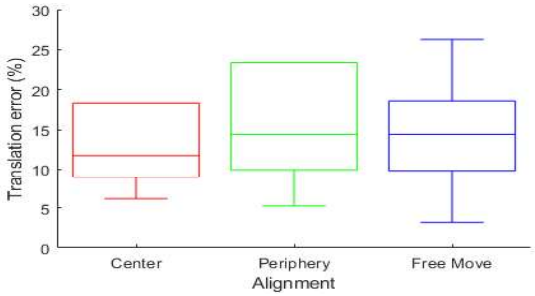


Figure 5.20: Translation error in the virtual cube alignment with SPAAM for the first dataset, each boxplot corresponds to 6 alignments

5.3 Final Results

The final outcome of Mode 2 and its sub-modes are presented in this section.

5.3.1 Display of 3D Model

In Fig. 5.21 shows the 3D virtual model of a shoulder overlaid with the printed model. Although, the virtual model is not totally overlaid correctly with the bone, it is considerably more accurate that the overlaid displayed in Fig. 5.25, where it is used a calibration from Mode 1 (Fixed Display). The other methods are also shown in Fig. 5.22, Fig. 5.23 and Fig. 5.24.

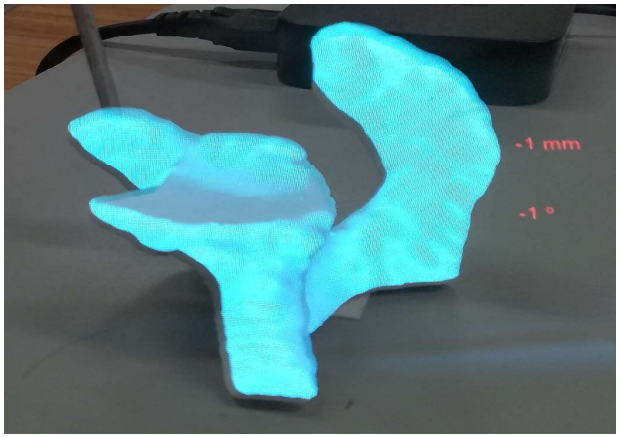


Figure 5.21: 3D Model when SPAAM calibration is used

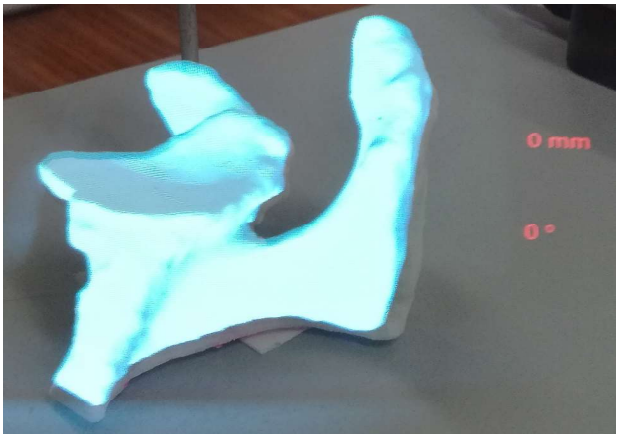


Figure 5.22: 3D Model when $SPAAM_{avg}$ calibration is used

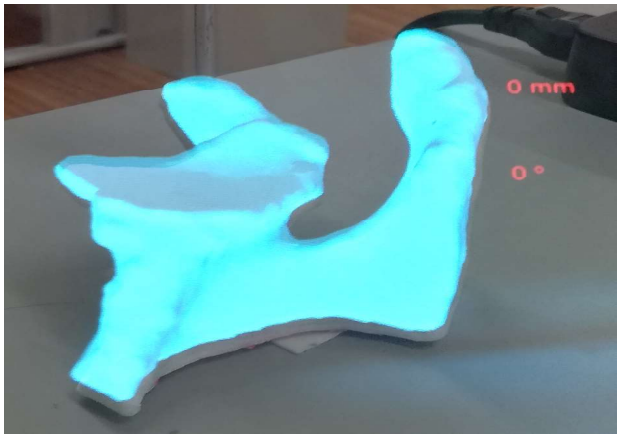


Figure 5.23: 3D Model when $SPAAM_{forced}$ calibration is used

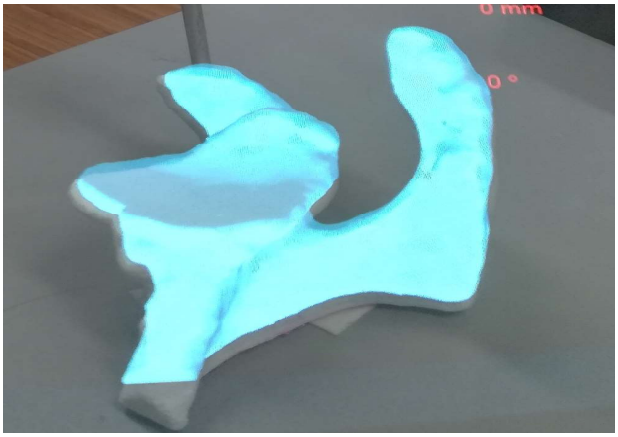


Figure 5.24: 3D Model when $SPAAM_{fixed}$ calibration is used

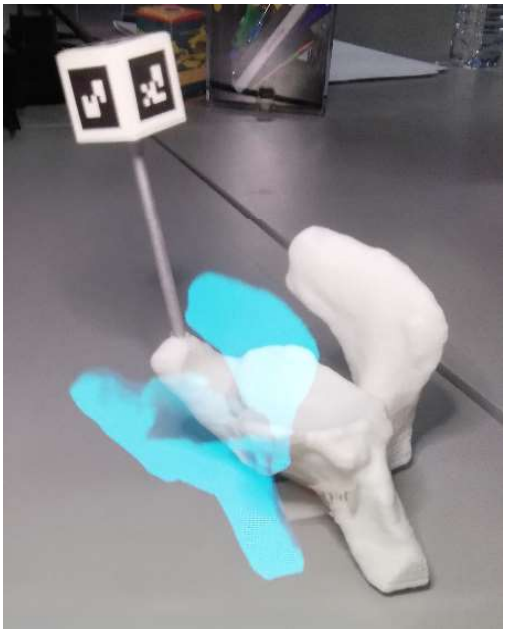


Figure 5.25: Incorrect alignment using calibration from Mode 1

5.3.2 Highlights

This sub-mode consisted on overlaying the highlighted part of the 3D model and its point of intersection with the marker anchor, as shown in Fig. 5.26. A slight deviation of model is seen due to the calibration.

Figure 5.26: Highlights of the 3D model

5.3.3 Guidance

Finally, the sub-mode of guidance allow the user to align a tool marker with a virtual rod, in order to perform the process of perforation. Two measurements are also shown that give the distance from the tip of the touch probe to the point of perforation and the angle between the rod and the probe. In Fig. 5.27 is shown a wrong alignment of the touch probe with its identifying red color. The Fig 5.28 shows the correct positioning of the touch probe with the wrong angle, resulting in the display of a yellow color. Finally, in Fig. 5.29 is shown the correct alignment of the touch probe and the rod, which results on the display of a green color.

An evaluation to the performance of the system is also conducted, where image processing was measured in in.nav. In order to compare the results, two consideration are firstly taken: the external camera has a fixed 30 frames per second and the HMD has a refresh rate of 60 Hz. in.nav without the HMD implementation processes 30 frames in one second, which correspond to the maximum provided by the camera. In a HMD implementation, in.nav speed drops to 16 to 19 frames each second. This performance downgrade is a result of the TCP/IP connection implemented, required for in.nav and Unity to communicate. Since the

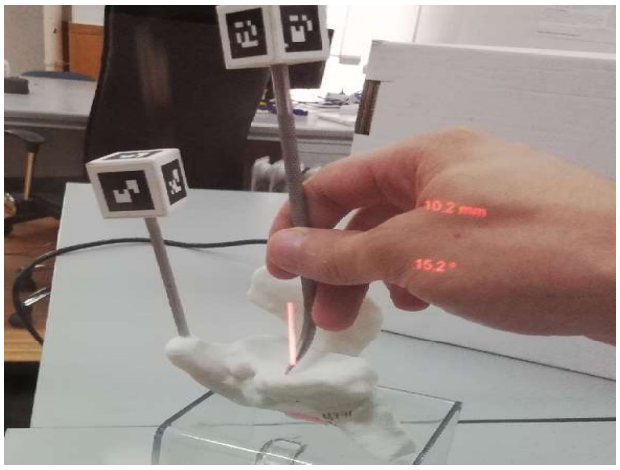


Figure 5.27: Wrong alignment of touch probe

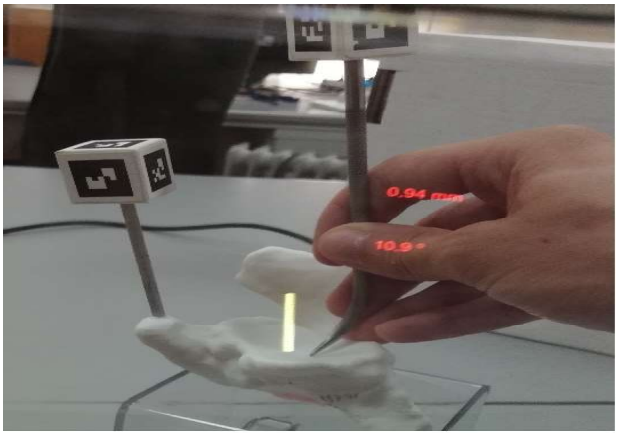


Figure 5.28: Correct position of touch probe but wrong angle of alignment

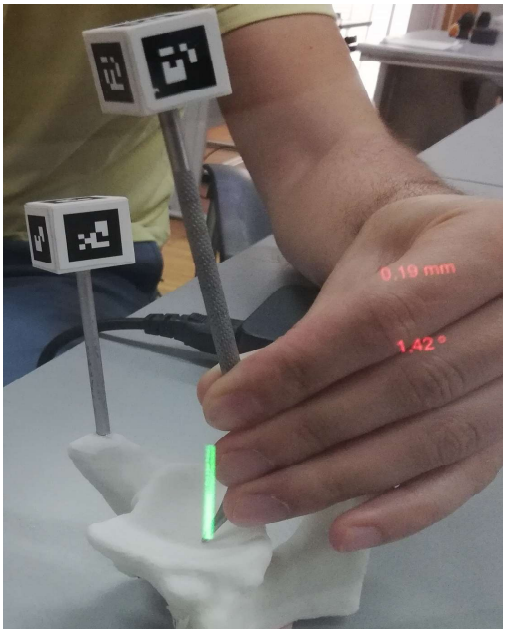


Figure 5.29: Correct alignment of touch probe

HMD has a refresh rate higher than the amount of frames processed, the bottleneck is due to the data communications.

Even though the results presented do not complete the full requirements for the application of surgical navigation in a real surgery, it gives a good approximation to what constitutes a good surgical navigation system and opens the development of future works. Also, it is important to notice that the touch probe showed in Fig. 5.27, Fig. 5.28 and Fig. 5.29 does not correspond to the tool normally used for perforation, it was only used as a dummy example.

6 Conclusions

This dissertation describes the work on the integration of an optical see-through HMD to surgical navigation. This resulted in the development of two operation modes: a first mode with its main focus on user usability by rendering a conventional 2D display and a second mode focused on rendering guidance information over patient's anatomy. Despite the difficulties that appear during this work, the main objectives were achieved, resulting in a interesting development with several conclusion to be taken:

- DreamGlass hardware constitutes a problematic subject since the lack of eye-tracker limits the amount of calibration methods that can be used;
- Performance wise, the bottleneck in the decrease of image processing in in.nav is due to the TCP/IP connection, which reduces the quality of model display;
- The results obtained from calibration of the HMD show that Single Point Active Alignment Method (SPAAM) proposed in the literature [24] has a poor depth estimation. Using SPAAM variants that estimate fewer DOFs lead to a more accurate depth estimation, which indicates that estimation of 17 DOF for normal SPAAM might not be adequate. Additionally, fixing the depth problem allows the improvement of calibration without the need to use StereoSPAAM, which is both time-consuming and tedious;
- Preliminary results obtained based on the use of a calibration cube for computing an alignment metric suggest that the quality of rendering degrades as it moves to the periphery of the screen and so calibration methods that use more complex projection models may need to be used;
- Although not having attained the precision required for surgical navigation, the obtained results demonstrate the advantages and weaknesses of the employed methods and open doors for future works.

6.1 Future Work

The base line of implementation for these systems (in.nav and Dreamglass) is completed in this dissertation. This allows the development of future works that can improve the correct implementation and also other works in different HMDs. Some of the possible improvements are:

- The inclusion of an eye-tracker to the HMD and posterior estimation of the pose of the eyes would definitely improve several of the calibration problems;
- Improvement of frame rate in the in.nav in order to increase quality of display by fully integrating the system in the HMD;
- Unity constitutes a challenging API to develop for an OST-HMD, which results in an increased learning curve to overcome. Some research can be made in order to implement in a different API;
- Improvement of calibration by developing a new method or in case of eye-tracker introduction, the use of semi-automatic or automatic methods;
- Without eye-tracker some improvements can be made to the periphery rendering problem by introducing the rotation of the eyes into the calibration process.

7 Bibliography

[1] Dreamglass project. "https://bitbucket.org/joaofalcaouc/dreamglass/src/master/".

[2] Opengl programming. "https://en.wikibooks.org/wiki/OpenGL_Programming/Mini-Portal_Smooth".

[3] M. Arefin, K. Moser, and J. E. I. Swan. Impact of alignment point distance and posture on spaam calibration of optical see-through head-mounted displays. *IEEE International Symposium on Mixed and Augmented Reality*, Oct 2018.

[4] Arspectra. "https://www.arspectra.com/".

[5] Augmedics. Xvision. "https://www.augmedics.com/".

[6] Ehsan Azimi, Long Qian, Nassir Navab, and Peter Kazanzides. Alignment of the virtual scene to the 3d display space of a mixed reality head-mounted display. Oct 2018.

[7] João Pedro De Almeida Barreto and U. de Coimbra. Methods and systems for computerised surgery using intra-operative video acquired by a free moving camera. Sep 2016.

[8] Jean-Yves Bouguet. Camera calibration toolbox for matlab. "http://www.vision.caltech.edu/bouguetj/calib_doc/".

[9] S. Condino, M. Carbone, R. Piazza, M. Ferrari, and V. Ferrari. Perceptual limits of optical see-through visors for augmented reality guidance of manual tasks. *IEEE Transactions on Biomedical Engineering*, pages 1–1, 2019.

[10] Yakup Genc, Frank Sauer, Fabian Wenzel, Mihran Tuceryan, and Nassir Navab. Optical see-through hmd calibration: a stereo method validated with a video see-through system. pages 165 – 174, 02 2000.

[11] J. Grubert, Y. Itoh, K. Moser, and J. E. I. Swan. A survey of calibration methods for optical see-through head-mounted display. *IEEE Trans Vis Comput Graph*, 24(9):2649–2662, Sept 2018.

[12] Kore. Understanding the different types of ar devices. "`https://uxdesign.cc/augmented-reality-device-types-a7668b15bf7a`", Sept 2018.

[13] G. Kramida. Resolving the vergence-accommodation conflict in head mounted displays. *IEEE Trans Vis Comput Graph*, 22(17):1912–1931, July 2016.

[14] J. E. S. I. E. Kruijff. Perceptual issues in augmented reality revisited. Oct 2010.

[15] Kenneth R. Moser and J. Edward Swan. Improved spaam robustness through stereo calibration. *IEEE International Symposium on Mixed and Augmented Reality*, Oct 2015.

[16] Novarad. Opensight. "`https://www.novarad.net/products/opensight/`".

[17] A. Patney, J. Kim, M. S. A. Lefohn, A. K. D. Luebke, C. Wyman, and N. Benty. Perceptually-based foveated virtual reality. *ACM SIGGRAPH 2016 Emerging Technologies*, (17), 2016.

[18] F. Picard and J. Clarke. Computer assisted knee replacement surgery: Is the movement mainstream? *K. D. O. Muscular*, 2014.

[19] Ricardo Simoes, Carolina Raposo, João Pedro Barreto, Philip Edwards, and Danail Stoyanov. Visual tracking vs optical tracking in computer-assisted intervention. (in press).

[20] J. B. Stiehl. Computer navigation in primary total knee arthroplasty. *The Journal of Knee Surgery*, 20(2):158–164, 2007.

[21] Andrew Straw. Computing the opengl projection matrix from intrinsic camera parameters. "`https://strawlab.org/2011/11/05/augmented-reality-with-OpenGL`".

[22] Insight Medical Systems. Arvis. "`http://insightmedsys.com/`".

[23] A. Todesca, L. Garro, M. Penna, and J. Bejui-Hugues. Conventional versus computer-navigated tka: a prospective randomized study. *Knee Surg Sports Traumatol Arthrosc*, 25(6):1778–1783, 2016.

[24] Tuceryan, M. Genc, and Y. Navab. Single-point active alignment method (spaam) for optical see-through hmd calibration for augmented reality. *Presence: Teleoperators and Virtual Environments*, 11(3):259–276, June 2002.