DESIGN AND DEVELOPMENT OF AUTONOMOUS ROBOTIC MACHINE FOR KNEE ARTHROPLASTY

OMAR SHALASH

Department of Biomedical Engineering

University of Strathclyde

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Biomedical Engineering

September 2018

Copyright Statement

Signed:

Date:

Acknowledgement

I would like to express my sincere gratitude to my advisor Prof. Philip Rowe, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless.

I also like to express gratitude to my wife and my backbone Esraa who spent sleepless nights with me and was always my support in the moments when there was no one to answer my queries.

My sincere thanks also goes to Shailesh and Mathew who provided me with skills and information which made significant contributions to my research.

Special thanks to Prof. Mohamed El-Habrouk for the help and support in my research.

I would like to thank my family for supporting me spiritually throughout writing this thesis and my life in general.

last, I would like to extend thanks to the many people, in many countries, who so generously contributed to the work presented in this thesis.

Abstract

Robotic aided surgery has become mainstream in many fields of medical science. Two systems have been developed to assist unicompartmental knee arthroplasty. One of these systems is the Blue Belt system which is a hand-held robotic tool designed to remove the damaged surfaces of the patient's knee to aid the surgeon during knee replacement surgery. Another more expensive system is the Mako Rio robot, it provides better accuracy and precision and is haptic.

However, both the Blue Belt and the Mako Rio robot are expensive which is an obstacle for widespread adoption in the field. These systems are also manually driven by an operating surgeon for historic medico-legal reasons.

In this research, a Novel system was developed to provide the required accuracy and precision but in a more affordable automatic system. It was guided by an OptiTrack motion capture navigation system. A CNC machine was built to drive the cutting burr across the knee joint. Multiple software applications were developed to enable the operation of the system such as communication with the navigation system, cluster marker identification and tracking, path planning, wireless communication, and CNC control. The system can perform the cutting phase in the surgery autonomously.

The system has been tested on two sets of tibia and femur artificial bones and then the cut shape was analysed. The mean error of the cutting process was 1.9 mm with standard deviation of 0.55 mm for the tibial bones. The femural bones also showed improved surface finish.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Osteoarthritis, also known as degenerative wear-and-tear arthritis, is the most common type of arthritis, especially among the elderly. It is thought to occur when the cartilage in the knee joint is disrupted which increases the friction and contact between knee bones causing pain and discomfort (Hill et al., 2015).

When osteoarthritis is not cured by non-surgical treatments –such as: lifestyle modifications, physical therapy, or medications– doctors recommend undergoing knee arthroplasty surgery. In the Total Knee Arthroplasty (TKA) or Unicompartmental Knee Arthroplasty (UKA), the disrupted cartilage is removed, the affected bones are resurfaced, with new metal on plastic bearing surfaces (Harwin, 2003) or ceramic implants (Koshino et al., 1997).

UKA is less invasive surgery –compared to TKA– as the surgeon only replaces the deteriorated portion of the knee through a relatively small incision. It has advantages over TKA including improved function results, faster recovery, and less blood loss (Bell et al., 2016).

Knee arthroplasty can be either performed solely by the surgeon or with the assistance of a robotic system, such as: Mako Rio and Blue Belt Navio. Robot-assisted surgery reduces the chance of human error. A limited number of studies have shown that robotic-assisted surgeries give better implant alignment, enhanced knee motion, and faster recovery with less post-operative complications and discomfort compared to patients who received a conventional UKA knee surgery (Motesharei, 2014), (Pearle et al., 2010),(Dunbar et al., 2012),(Bell et

al., 2016),(Smith et al., 2014). In this thesis we seek to build on these studies by prototyping a system which was designed to be more affordable than the existing ones, while retaining the necessary accuracy and precision required for the bone cutting. While the proof of concept has been established by this expensive machines, they are beyond the scoop of all but first world countries. The system consists of a navigation system, a set of developed applications runs on a separate computer and a CNC (computer numerical control) robotic cutting machine.

The system was navigated by OptiTrack^{TM} motion capture system. The OptiTrack system is a much cheaper than other motion capture camera systems and has the potential to be as accurate as conventional camera navigation systems. The Mako robot uses a by Polaris camera system for navigation. The Polaris system was two cameras, a Spectra and a Vicra. The Spectra camera costs $23,000, while Vicra camera costs $12,000 (Dockter, 2013). The OptiTrack with twelve camera costs $\$11,346$ (*Build Your Own Motion Capture System*, 2018) as opposed to the $35,000 needed for the Polaris two camera system.

A series of applications were developed to navigate and control the robotic CNC machine. The main application in D-Flow tracked motion capture clusters and planned the cutting path for the burr based on the knee registration process and CNC machine location. The cutting path was stored in a file.

Another application was written to send the cutting file through a wireless network by a transmission control protocol. In the created communication protocol this application represents the client side, with the server side being an Arduino micro-controller which controlled the CNC machine.

The server side Arduino application manages the data transfer from the computer which runs the previously mentioned applications to the Arduino controller through its Wi-Fi shield module. The Arduino controller also control the CNC robotic machine. Once the machine starts receiving data of the cutting path, the burr starts drilling the shape of the knee joint implant on the bone surface.

The thesis explains the design, development, accuracy and precision of the prototype device when cutting sawbones. The thesis sets out the developments undertaken the outcome and possible future development required for the pro-

duction of a full working prototype.

The system was tested on two sets of tibia and femur bones. the tibial cut sawbones were then 3D scanned and compared to same type uncut bones. The cut sawbones were then compared to another bones that have been cut using a different robotic assisted surgical system. The mean error of the cutting process was 1.9 mm with standard deviation of 0.55 mm while the Mako set had 2.87 mm with standard deviation of 0.84 mm and the Blue Belt system cut set had mean of 3.07 mm with standard deviation of 0.94 mm. The femural cut sawbones also showed smoother surface finish when compared to the other two commercial systems cuts.

Chapter 2

Literature

2.1 Robotic Arthroplasty

Knee replacement surgery, either Total Knee Arthroplasty (TKA) or Unicompartmental Knee Arthroplasty (UKA), is recommended for many patients who suffer from osteoarthritis and joint problems. In knee osteoarthritis, the cartilage guarding the bones slowly wears away, this can occur in one or more knee compartments (van der Esch et al., 2013), (Felson, 2006). Surgery is performed in order to reduce pain and restore joint function.

Although the number of re-constructive knee surgeries has grown rapidly, better patient satisfaction still needs to be reached, satisfaction ranges from 82% to 89% after TKA, and 80% to 83% in UKA (Sharkey et al., 2002),(Manaster, 1995). Sharkey et. Al performed a study to determine the main causes of failure of TKAs. It was found that the main failures were: loosening (95%), infection (27.4%), instability(7.5%), periprosthetic fracture(4.7%) and arthrofibrosis (4.5%)(Sharkey et al., 2014),(Parvizi et al., 2011). The efficiency of these surgeries depend on two main factors: the implant design and the surgery technique itself. Many changes have been made to the implant design such as the 3D anatomical design and the materials used. However, these changes have not improved patients satisfaction (Bourne et al., 2010). Thus, recent enhancements in knee replacement surgery have focused on improving the surgical technique itself, including, but not limited to, access to the joint, implant sizing, alignment and fixation, and wound closure.

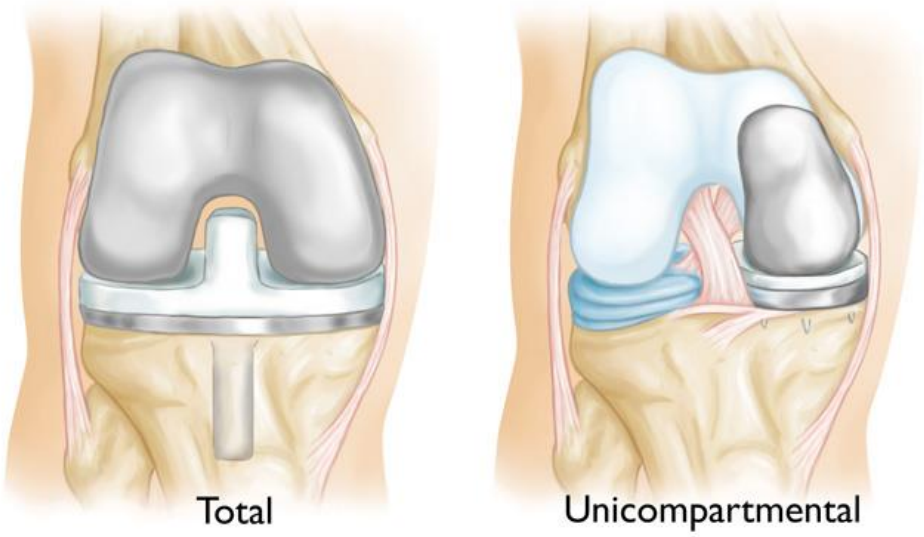


Figure 2.1: TKA vs UKA (Reproduced with permission from OrthoInfo. (c) American Academy of Orthopaedic Surgeons.) http://orthoinfo.aaos.org (Foran, 2016)

Knee replacement surgery -also known as "knee resurfacing"- was first performed in 1968 (Lonner & Kerr, 2012). UKA is a less invasive operation compared to the TKA, which makes it preferred if the patient's history is favourable (Harwin, 2003). During Partial knee resurfacing or UKA, only the damaged compartment is replaced with a partial knee implant while preserving the undamaged bones and tissues (Harwin, 2003). Figure 2.1 shows the difference between UKA and TKA. Each compartment of the knee could be solely damaged by osteoarthritis, so there is an implant designated for each. If a second compartment becomes damaged, it can also be resurfaced and the appropriate uni-condylar implant fitted. In this manner, a total knee replacement can be achieved while preserving the healthy undamaged tissues and bones. Also, UKA needs a smaller incision compared to TKA, it ranges from 3 to 4 inches, therefore patients spend less time in hospital for recovery compared to TKA. Moreover, patients claim that a uni-compartmental knee resurfacing feels more natural with better bending of the knee (Repicci, 2003). Moreover, UKA has resulted in 60% lower postoperative complications, though it increases the risk of revision surgeries(Piva & Klatt, 2017; Purcell et al., 2018; Maxwell et al., 2017).

Knee arthroplasty can be performed using the following techniques: conventionally by the surgeon who orients or places the knee implant manually, or by a robotic system controlled or supervised by the surgeon. In case the surgery is performed manually by the surgeon, the alignment and positioning of the implant are determined visually by the surgeon who inspects/analyse X-ray and sometimes CT scans of the affected joint and adjusts cutting guides accordingly to make the cuts. On the other hand, performing knee arthroplasty using a robotic system makes use of Computed Tomography (CT) scans instead of X-rays. CT scans are preferred in this case as they enable the surgeon to construct a 3D image of the knee joint. These constructed 3D images are uploaded onto the software of the robotic system in order to guide the surgeon who controls the robotic arm during the surgery. This gives more precise results and also less invasive to the knee joint during the surgery (Magee, 2018).

Robot-assisted surgery reduces the chance of human error during the operation, this reduces instances of implant malalignment (Motesharei, 2014). In a paper by Moon et al. (Moon et al., 2012), a comparison was made between robot-assisted arthroplasty versus conventional surgery. Moon et al. used multiparameter quantitative three-dimensional CT assessment of the implant alignment. Results of the comparison proved that robot-assisted TKA yields higher precision and accuracy levels in the sagittal and coronal planes of the 3D CT scan, and also in relation to the femoral rotational alignment.

Another Study by Motesharei and coworkers (Motesharei, 2014) compared between robotic-assisted UKA using the Mako system (illustrated in section 2.1.2) versus conventional Oxford UKA (illustrated in section 2.1.1). This paper shows that robotic-assisted UKA gives better implant alignment, enhanced knee motion and faster recovery. Motion analysis was conducted one year after the surgery for both the robotic-assisted UKA and conventional UKA patients. Those who received a robotic-assisted UKA operation achieved a high range of movements values during the highest flexion portion of the weight bearing stage of the gait cycle. These values were comparable with normal healthy knees. On the other hand, patients who received a conventional UKA, has lower knee excursion angles

less than the normal range. Robot-assisted arthroplasty has proven to show benefits over conventional techniques

Many computer-assisted surgical systems have been developed over the years. These systems are either active, semi-active, or passive, depending on how independently they perform the surgery (Picard et al., 2004). Active robots can perform some surgical tasks without the direct intervention of the surgeon, these tasks can be drilling or milling, etc., while in passive systems no part of the surgery is overtaken by the machine and the surgeon is in full control at all time of the surgery. Finally in semi-active systems, the control of the surgery is divided between the surgeon and the robotic tool, the robotic tool is not autonomous but enhances the surgeon control.

One of the drawbacks to robotic-assisted systems is their relatively high cost. However, the overall cost effectiveness of the robotic-assisted surgery is more important than the purchase price itself. Using a Markov decision analytic model a study was made to assessed how lifetime costs and quality-adjusted life years (QALYs) vary as a function of age at the time of initial treatment (ATIT) of patients with end-stage unicompartmental knee osteoarthritis undergoing TKA, UKA, and nonsurgical treatment (NST). Separate models were estimated for ATITs at 5-year intervals from 40 through 90 years. Direct and Indirect costs were calculated. Cost-effectiveness and incremental cost-effectiveness ratios (ICERs) were calculated for each treatment at each ATIT. Societal savings were estimated. The study showed in order to maximize cost-effectiveness, NST should be used sparingly in patients below the age of 70 years and UKA should be chosen over TKA (Kazarian et al., 2018).

Another constraint to a wider use of robotic-assisted systems is the surgeons themselves. Many surgeons would prefer to rely on their manual surgical skills. Only a small number of surgeons have used these new robotic techniques in surgery. So far only a few sites have adopted the robotic surgery approach despite better surgical alignment achieved when using robotic-assisted systems (Motesharei, 2014).

The main aim of developing computer-based surgery systems is to enhance

the end results of knee-arthroplasty and improve the surgical precision of the operation. However, robots do not replace the role of the surgeon; they assist them by relocating and repositioning the surgical tools during the operation (Specht & Koval, 2001). Robotic arthroplasty assists in taking control of the bone cutting which is a key factor that may affect the final result of knee arthroplasty. It gives the surgeon the ability to perform better implant positioning and fixation(Van der List et al., 2016). However much of the surgery remains in the hands of the surgeon particularly the soft tissue balancing around the knee which is probably the key to successful post operative function. Moreover, as surgical robotic platforms advance in terms of accuracy, precision and accessibility, it is likely that both UKA and TKA knee arthroplasty will benefit and that once they become mainstream they will help decrease healthcare costs.

In the following subsections, examples of surgical UKA systems are demonstrated.

2.1.1 Oxford UKA

The Oxford Partial Knee is an evolution of the original arthroplasty concept, first used in 1976 in Oxford (Goodfellow & O'Connor, 1978). The Oxford UKA was designed by John Goodfellow and John O'Connor in 1982 (Goodfellow et al., 1988). Its design gives a minimal polyethylene wear by offering the advantage of a large area of contact throughout the entire range of movement (Argenson & O'Connor, 1992), (Psychoyios et al., 1998). The current Oxford implant is based on its clinically successful predecessors (Phase I and Phase II) which achieved survival rates of 98% in 10 years (Murray et al., 1998), (Price et al., 1999), with an average wear rate of $0.03mm$ per year (Argenson & O'Connor, 1992), (Psychoyios et al., 1998).

Early Oxford UKA implants had problems with high failure rates, as discussed by Kort et al (Nanne Kort, 2018). Many factors lead to these failures such as the polyethylene wear at the bearing surface and the implants geometry. Therefore, over the years many enhancements have been made to the prosthesis itself which improved the outcomes of the operation.

Figure 2.2: Oxford knee phase three III. (ZimmerBiomet, 2018)

The implant design comprises a femoral component, a tibial component, and a polyethylene bearing in between (Figure 2.2). The femoral component is a unique, spherically designed cast cobalt chromium molybdenum alloy which gives it strength, higher wear resistance and biocompatibility. The design is available in five sizes which are parametric and have corresponding radii of curvature. A suitable size of the femoral component is chosen according to the patient's size which is determined through pre-operative templating of lateral radiographs and intra-operative measurement confirmed with sizing spoons (Goodfellow et al., 1987)

The tibial component is also made of cast cobalt chromium molybdenum alloy. The design is available in seven sizes, both right and left. These provide optimal bone coverage, while avoiding component overhang anteromedially.

The bearing between the femoral and tibial components is manufactured from ArCom Direct Compression Molded Polyethylene for better wear resistance. The design has five sizes in order to match the radii of the five femoral component sizes. For each size, there is a range of seven thicknesses ranging from $3mm$ to $9mm$.

A medium sized femoral component is suitable for most patients. This size was the only size used in Phase I and II components. However, patients who are less than 5 ft and 5 inches require a small sized component, patients who are more than 5 ft and 7 inches require a large sized component. The extra small and

extra-large components are rarely used only in very small women and extra-large men respectively.

Surgery Technique

The main steps of the surgery are listed below (Goodfellow et al., 2011):

- Positioning the limb.
- Skin incision.
- Osteophyte Excision
- Tibial plateau resection
- The femoral drill holes and alignment.
- Femoral saw cut
- First milling of the condyle
- Equalizing the flexion and extension gaps
- Confirming equality of the flexion and extension gaps
- preventing impingement
- Final preparation of the tibial plateau.
- Final trial reduction
- Cementing the components

2.1.2 Mako Rio

Figure 2.3: Mako Rio System

The Stryker/Mako haptic Robotic Interactive Orthopaedic Arm (RIO) was introduced in 2005, see Figure 2.3. Since then, it was used in more than 50,000 UKA operations (Van der List et al., 2016). In the United States, 20% of the UKA operations have been performed using the Mako surgical system (Chawla et al., 2016). The only major drawback for using the Mako system is its cost, it costs nearly 1 million US dollars (Motesharei, 2014). In 2016 Zimmer Orthopaedics purchased the Biomet robotics company and launched the ROSA^{TM} robot for spinal surgery and it is likely this will also be used for knee arthroplasty in the future. In June 2017 Smith and Nephew acquired Blue Belt technologies the makers of the Navio hand-held robotics assisted platform which was given FDA clearance in 2002 for partial knee replacement. Most recently of all, Johnson and Johnson acquired the robotic assisted surgery company called Orthotaxy in 2017 whose system is in early stage development for total and partial knee replacements. Hence as robotic systems in orthopaedic increase in popularity, there will

be more competition which is likely to reduce the cost.

The partial knee arthroscopy system developed by Mako Surgical Inc. (Fort Lauderdale, Florida, USA) consists of a robotic arm which interacts with the surgeon in order to guide him through preparing the patient's knee for the implantation. The system has 3 main components: a robotic arm, optical camera, and controlling computer. It uses CT scans taken for the patient's knee prior to the operation. These CT scans are used to form 3D image of the knee joint which allows the surgeon to accurately plan the navigation throughout the surgery and then implement the bone cutting using the robotic platform (Lonner & Kerr, 2012). During the operation, the robotic system provides a stereotactic interface which constraints the cutting tool during the femoral and tibial cutting stage. In contrast to the Oxford UKA system which makes use of manual instruments such as: pinned cutting blocks, saw, jigs, etc. The Rio system controlled the surgeon and so does not hypothetically require the same surgical skill as the manual technique requires.

One of the significant advantages to the Makoplasty system is that the bone cutting or Osteoplasty is planned before the beginning of the operation. CT scans are imported into the Mako software, which then forms a 3D model specific for the current patient. This is achieved by using the CT scans to form slices at knee joint, and through the hip and ankle. These slices are defined and combined to generate the 3D model while the hip and ankle data give the mechanical alignment of the knee. Using these 3D models, the implants are superimposed on the knee joint to visualize their position and alignment. In this way the system can be used to provide better accuracy for the bone resection (Lonner & Kerr, 2012).

Before the surgery, the motion capture system and the robotic arm are calibrated. When the patient enters the theatre, an incision is made in order to reveal the knee joint. Then the surgeon confirms that the knee joint is ready to undergo the surgery. Multiple markers and pointers are used to pinpoint anatomical landmarks on the patient's knee which are then correlated with others on the knee joint model generated by the CT scans. This is done in order to match the knee model with the patient's knee in theatre. Probes (or pointers) are used in

order to locate the landmarks on the bony surface. Moreover, in order to achieve a constant point of reference, pins are drilled into the proximal tibia and distal femur to become fiducial for registration and matching processes. Tracking arrays are attached to bone pins in the tibia and femur which act as references for the system to locate and position the tibia and femur.

After the registration and matching has been finalised, a soft tissue balancing algorithm is launched. This is achieved by applying a varus/valgus stress assessment and range of movement assessment in order to capture patients kinematic nearly every 10° and then capturing gapping of the knee at different points throughout a passive range of motion. This shows how tight the components will be in different amounts of flexion-extension, when the initial implant position is as planned. The implant position can be adjusted in silicon until a good balance is achieved (Lonner & Kerr, 2012).

After adjusting the final planned implant position, bone cutting is performed with the constraint applied by the robotic arm. The burr is moved into a haptic zone where the system applies boundaries. Only within these boundaries is the arm is allowed to move. Any attempt to move the arm outside the boundaries is met with a force confining the burr to the targeted area of the bone. As a result, only those parts of the bone are cut that are required to allow the implant to be fitted. After the cutting procedure, trial implants and bearings are provisionally implanted. The ones that achieve the best feel, range of motion and stability are chosen. The trial implant is then removed and the area is cleaned and the real implants are cemented into their position in the bone and the bearing inserted. Finally, the incision is stitched up and the surgery is complete.

The University of Strathclyde in association with Glasgow Royal Infirmary has undertaken the first independent randomised controlled trial of the Mako system against the Oxford UKA (Motesharei, 2014).

Below are examples of various studies made in order to assess the efficiency of the Mako RIO system:

- The mechanical alignment of the Mako system was observed in 10 patients who had a medial UKA robotic-assisted surgery, it was reported that the

intra-operative registration step took 7.5 minutes and the burring itself took 34.8 minutes (Pearle et al., 2010). Pearle et al. compared between the planned lower leg alignment and the actual post-operative leg alignment 6 weeks post operation. It was found that all measurements were within 1.6°of the planned alignment.

- Another study was made to assess the accuracy of the implant positioning of the Mako system in 20 patients (Dunbar et al., 2012), in this study they compared 3D CT scans of the knee joint before and after the operation. It was found that the femoral components were within $0.8mm$ and 0.9° in all directions and that the tibial components were within $0.9mm$ and 1.7° in all directions.

- A study by Plate (Plate et al., 2013) assessed the accuracy of soft tissue balancing in Mako system in 52 patients, they found that at all flexion angles the ligament balancing was within $0.53mm$ of the original plan. Moreover, it was found that in 83% of the patients accuracy was within $1mm$ at all flexion angles.

- In a study by Bell (Bell et al., 2016) a comparison was made between one hundred and twenty patients who were randomly treated using either the Mako RIO system or the conventional surgery using the Oxford Phase-3 UKA. Three months after the operation, patients' tomographic scans were used to examine the accuracy of the axial, coronal, and sagittal implant positioning. Results showed that the percentage of patients whose implants were within 2° of the target position was significantly higher (Figure 2.4)among those who were treated using the Mako RIO system.

In conclusion, Makoplasty achieves enhanced surgical results. It increases the accuracy of the surgery and reduces post-operative complications and discomfort compared to the conventional UKA with Oxford implants (Blyth et al., 2013).

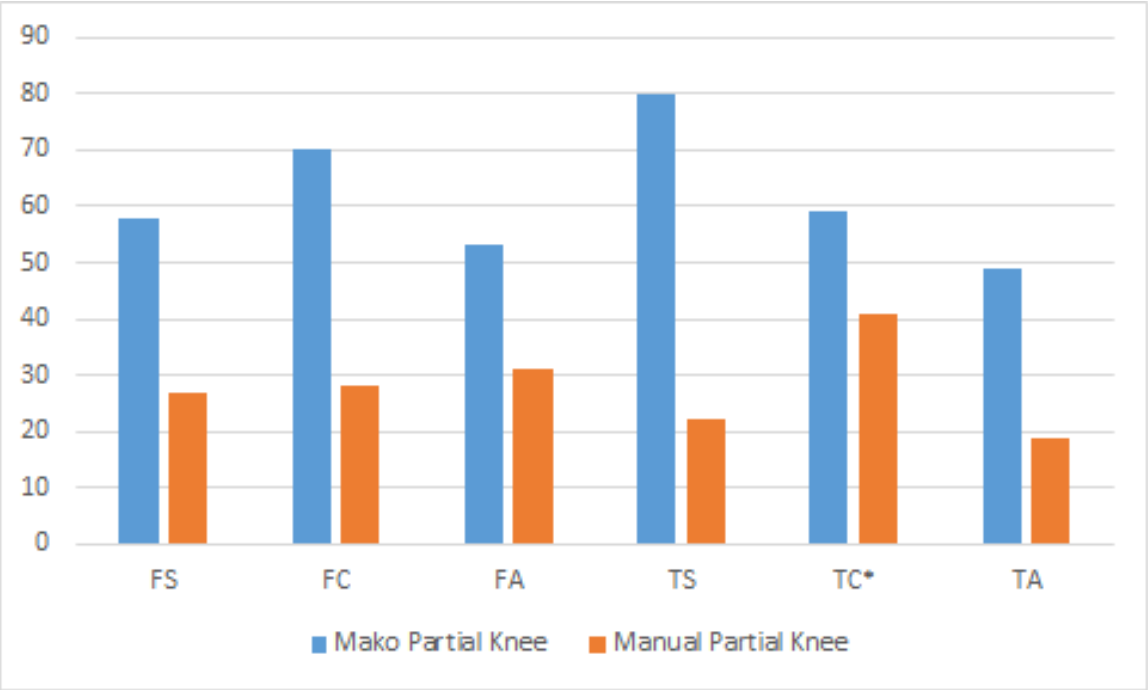


Figure 2.4: Percentage of knees with components positioned within 2° of the target position. FS= Femoral Sagittal, FC= Femoral Coronal, FA= Femoral Axial, TS= Tibial Sagittal, TC*= Tibial Coronal, TA= Tibial Axial, *= Non-significant parameter (Stryker.com, 2018)

2.1.3 NAVIO Blue Belt

Figure 2.5: NAVIO Blue Belt System

Blue Belt Technologies developed the NAVIO PFS surgical system, see Figure 2.5 above. This is a hand-controlled cutting tool which is controlled via an optical tracking system and the rotating burr retracts when the constraint boundary is approached in order to allow depth control of the cutting tool (Payne, 2015). Therefore, unlike the Mako system, it is not a haptic system. It is more general in application when compared to the Mako system as it enables the surgeon to use any implant. Moreover, it costs less than the Mako system at $300,000. Further clinical trials and investments need to be undertaken by the Blue Belt company if the system is to become widespread. (Motesharei, 2014).

This system is an image-free semi-active robotic system and has the same characteristics as the Mako system (Lonner, 2016).

A study was made to assess the accuracy of implant positioning in 20 synthetic

femurs and tibia. It was found that the maximum rotational error was 3.2°, the angular error was 1.46° in all orientations, and the maximum translational error was 1.18 mm for both the tibial and femur components. Another study tested the accuracy of implant positioning in 25 cadavers. The results were similar to these found in the study of Smith et al (Smith et al., 2014) and similar to the results of other robotic-assisted systems.

Smith and colleagues assessed the accuracy of component positioning using 20 synthetic femurs and tibia (Smith et al., 2014). They reported a maximum rotational error of 3.2°, an angular error of 1.46° in all orientations and a maximum translational error of 1.18 mm for both the tibial and femoral implants. Lonner and colleagues (Lonner et al., 2015) assessed the accuracy of component positioning in 25 cadaveric specimens. They found similar results as were found in the study of Smith and colleagues and concluded that these results were similar to other semi-active robotic systems designed for UKA (Dunbar et al., 2012; Cobb et al., 2006).

In summary then the Mako Rio gives the most accurate cuts, closely followed by the Blue Belt system and both are significantly more accurate than manual cutting of bones.

2.2 Navigation System

2.2.1 Introduction and History

Motion Capture (MoCap) is the process of sampling and recording motion of subjects as 3D data. It is also defined as "The creation of a 3D representation of a live performance" by Alberto Menache (Menache, 2000). It has been used since 1872 when Edward Muybridge used it to perform an experiment on a horse to confirm that a horse raises all its four feet off the ground while trotting (Sharma et al., 2013). He placed cameras in a row with hip wires to capture multiple pictures of the horse's movements and proved his statement. Afterwards, Etienne-Jules Marey analysed human and animal motion with cine film (Braun, 1994). Since then, motion capture systems have been widely used in many fields such as enter-

tainment, sports, and medicine. Since the 1950's MoCap has played an important part in Biomechanical research and clinical practice. (Gage, 1993),(Lofterød et al., 2007).

More recently MoCap technology was spread and improved due to its use by the animation industry. Game development is now the largest market for motion capture technology. Many games are based on capturing real world movements of subjects and translating them into data to be inserted in a 3D model of the world in a virtual environment (Sharma et al., 2013).

2.2.2 Motion Capture Methodology

MoCap technology is implemented using as two main methods:

2.2.2.1 Marker-less Motion Capture

Due to the increasing use of computer vision applications, marker-less motion capture systems have been developed. These systems do not need any type of markers or special suits to track the subject's movements. Another advantage of marker-less motion capture systems is that the subject is not hindered by any wiring constraints, therefore it becomes easier to be captured and tracked with less time and cost (Nogueira, 2011). In this method video cameras capture multiple video stream of the subject's movements which are then analysed by computer vision algorithms to turn the captured data into a 3D character. Thus, the motion capturing process is performed via software without the need for any physical limitations. However, the data requires extensive post capture computing resources and could not be real time unless major simplification actions are undertaken.

One of the most famous example of a simple marker-less motion capture systems that is able to process in real-time is the Microsoft KINECT. It makes use of 3D depth cameras, colour cameras and a four-microphone array to enable users to interact with the software with their body movements in a natural way and also provide voice recognition capabilities (Zhang, 2012). KINECT usage has extended beyond the gaming industry to include robotics, electronic and computer

engineering.

However, while it is real-time and fits a human skeleton to the data, the process is less than perfect. The KINECT is unable to detect long bone rotations (rotations around the long axis of a bone) and hence the KINECT data lacks biomechanical validity.

2.2.2.2 Marker-Based Motion Capture

Marker-based motion capture systems are the mainstay of biomechanical research and applications. In marker-based motion capture systems, multiple markers and placed on the subject and then their movements are tracked. There are many types of marker-based systems according to the type of markers used:

- **Acoustical systems**: They make use of sound transmitters and receptors. A set of sound transmitters are fitted on the subject's main articulations while sound receptors are placed in the capturing area. In order to determine the 3D position of each transmitter, they are sequentially activated and produce a characteristic frequency that the receptors will pick up. The position of each transmitter is calculated by using the time taken for the sound waves to navigate from the transmitter to the receptor (Nogueira, 2011). One of the disadvantages of this type of system is the restrictions on the movement of subjects because of the cables of the sound transmitters.

- **Mechanical systems**: They make use of potentiometers and sliders and are often referred to as an exoskeleton. The subjects puts on a set of metal strips and on each joint there is a sensor giving its position. Mechanical MoCap system can include a basic skeleton, gloves, arms, or articulated models, etc. This method of direct measurement has many advantages as it is not effected by light or magnetic fields, it is inexpensive and does not need a long recalibration process (Menache, 2000). It has been used in robotic suits designed for heavy lifting. However, it does not preserve normal human movements.

- **Magnetic systems**: These make use of magnetic transmitters and receivers. A set of magnetic receivers are placed on the subject's main body segments so that the transmitters would trace and calculate the subject's movements. This method is widely used in simple movement capture as it is not very expensive compared to other motion capture systems. Its limitation is the huge number of cables which hinders the movement of the subject and would not be suitable for a surgical area, specially orthopaedic tables are full of metals (Guerra-Filho, 2005a), (Yabukami et al., 2000).

- **Optical Systems**: They make use of reflectors and high-resolution cameras in order to triangulate the 3D position of the subject. In animation applications the actor wears an especially designed suit with reflectors fitted on the main body segments. Cameras are placed in a certain manner in order to track the reflectors during the subject's movements. Each camera determines the 2D coordinates of each reflector, then a proprietary software is used to calculate the 3D coordinates of each reflector in order to generate a digital model for the subject. Due to this system's high sampling rate, it is recommended while capturing fast movements. Moreover, this system does not require cables, therefore the subject is not constrained. Occasionally, markers cannot be tracked if an obstacle gets in the way between the subject and the sensor, and this kind of system is prone to light interference. These problems are overcome using multi camera systems, careful marker placement, and blackout curtains.

2.2.3 Optical Motion Capture

Optical motion capture is one of the most important fields of computer vision and is responsible for many advances in a variety of research areas (Guerra-Filho, 2005a). Optical motion capture depends on using multiple cameras placed at different angles from the scene. Using two or more cameras to capture the same object enables us of reconstruction of the subject in 3D.

Optical motion capture can be performed in one of two ways: Reflective

(Passive) and Pulsed-LED (Active). Passive markers are coated with a retro-reflective material that reflects light emitted by Infra-red (IR) Light Emitting Diodes (LEDs) mounted around the camera lens. Light is then received back by the cameras. The Cameras are first calibrated so that only the passive markers are tracked. In order to calibrate the cameras, some markers with known positions are used; usually a wand having a group of reflectors is waved across each cameras' field of view. Active markers emit their own IR light instead of reflecting light emitted by the diodes around the cameras.

In order to build an optical motion capture operating theatre, four resources are required: A capture volume, markers, multiple cameras, and an image acquisition system (Guerra-Filho, 2005a).

- The capture volume should be large enough to allow the capturing of multiple viewpoints. IR Lighting must be uniform in order to minimize the presence of shadows and unwanted reflections in the capturing scene. This is achieved by using fixed light sources, non-reflective surfaces and blackouts.

- Markers are needed to track the moving subject, they can be fitted on a dedicated suit in case of tracking people or animals movements or in surgery attached to bone pins and surgical tools. As mentioned before, markers could be passive or active. Also, markers are distinguished using their position, colour, and shape.

- Progressive scan cameras are used in order to eliminate the saw pattern in captured videos. Cameras are synchronized with each other to correctly capture multiple viewpoints of the markers at the same time. Synchronization is achieved using an external trigger pulse from the computer received by each camera. Moreover, tracking accuracy is directly related to image resolution. Many factors need to be adjusted in order to minimize the blurring in the video, such as exposure and focus of the cameras. The cameras communicate with the video acquisition system using a USB cable or a FireWire.

- Video acquisition system which is responsible for receiving real-time synchronous video data, provide bandwidth for recording and storage of multiple video streams. A video grabber system moves the captured images to memory and then writes them onto disk. The acquisition system process the data to provide Cartesian coordinates in 3D for each marker.

2.2.3.1 Triangulation theory

Each camera detects the 2D position of each marker. For a specific marker, there are multiple sets of 2D coordinates; one set for each camera. When these sets are matched together, triangulation is performed to construct 3D coordinates (Faugeras & Robert, 1996).

Triangulation can be defined as the process of determining the 3D location of a point by forming multiple triangles to it from known points. The 3D location of a certain point can be computed by triangulation using the projections of this point onto the centres of two or more non-parallel image planes (Rahimian & Kearney, 2017).

2.2.3.2 Optical Motion Capture Products

The key hardware items in any optical motion capture system are the cameras. They must work at a suitable frequency and resolution for the target application. Moreover, they must provide high infra-red sensitivity Below are examples of some optical motion capture and tracking products:

- OptiTrack Slim 13E: It has multiple lens and filter options, 1.3 MP (mega pixel) resolution, 240 FPS (Frame per second) frame rate and GigE I/O to provide high-speed and high precision computer vision. Its price starts from $1499 (NaturalPoint, 2018d).

- OptiTrack Flex 3: It has a 0.3MP resolution (640×480), 100 FPS frame rate, it has a relatively high time latency of 10ms. It is supplemented with 26 LEDs. Its price starts from $599 (NaturalPoint, 2018a).

- OptiTrack Prime 13W: It has a 1.3MP resolution (1280 × 1024), 240 FPS frame rate, and a latency of 4.2ms. It is supplemented with 10 Ultra High Power (UHP) LEDs . Its price starts from $2,499 (NaturalPoint, 2018c).

- OptiTrack Prime 17W: It has a 1.7MP resolution (1664 × 1088), 360FPS frame rate, and 2.8ms latency. It is supplemented with 20 UHP LEDs.Its price starts from $3,499 (NaturalPoint, 2018b) .

- Polaris: The system consists of different hardware components that can be configured for different medical simulators. The system components are: Polaris Spectra, Polaris Vicra, System Control Unit and a host USB converter (NorthernDigital, 2018) . Polaris Spectra is used for simulation environments that require large measurement volume and costs around $23,000, while Polaris Vicra is used for simulation environments that require a small, targeted measurements volume, it costs around $12,000 (Dockter, 2013).

Figure 2.6: Polaris Camera System. (NorthernDigital, 2018)

- Vicon : For thirty years Vicon stayed the leading developer of motion capture systems. (Vicon, 2016). The Vicon Vantage camera series can allow the connection of up to 244 cameras to a single computer, Figure 2.7a. They have a 2000 Hz maximum frame rate, and resolution 5MP, 8MP, and 16MP for the V5, V8, and V16 cameras respectively. While the Vicon Vero family has a maximum frame rate of 250Hz in the v1.3 and v1.3x, and 330 Hz in the v2.2, Figure 2.7b. The resolution is 2.2MP for the v2.2 and 1.3MP for the v1.3, v1.3x, and vertex cameras. Another edition the Vicon Bonita, Figure 2.7c. The Bonita system costed £40,000, Bonita B3 240 Hz and B10 camera captures at 250 fps with 1MP of resolution.

(a) Vantage camera

(b) Vero camera

(c) Bonita camera

Figure 2.7: Vicon Vantage and Vero cameras

2.2.4 Measuring Performance

Optical motion capture systems are widely used in high-quality applications such as clinical Biomechanics. These types of applications require very high levels of accuracy and precision (Mündermann et al., 2006), (Harris-Love et al., 2004), (Hodt-Billington et al., 2008). Many factors influence the efficiency of the capturing process (Unal et al., 2007), such as: quality of the camera equipment, temporal-spatial resolution, illumination conditions, accuracy of the calibration process (Leardini et al., 2005), and marker specifications including shape, size, and inter-marker distances (Diaz Novo et al., 2014).

- Diaz et al. performed an evaluation of three different motion capture systems: two different Vicon motion capture systems and a low cost customized motion capture system installed in the Santiago de Cuba Hospital (SCH) using common video cameras. The standard deviation in measurement was much higher in the case of the SCH system than the two Vicon systems (Diaz Novo et al., 2014).

- Windolf et al. developed a method in order to assess accuracy and precision of motion capture systems regarding different system parameters, such as:

camera arrangement, calibration area, marker diameters, and using lens filters. The developed method was tested on a Vicon-460 system using four cameras. The calibration area was $180 \times 180 \times 150mm^3$. With the most favourable parameters the overall accuracy was $63\pm 5\,\mu m$, and the overall precision was $15\,\mu m$ (Windolf et al., 2008).

- Another study was performed to compare the linear accuracy from an OptiTrack system (low cost) and a Vicon system (high cost). The Vicon system had 12 Vicon MX cameras. The OptiTrack system had twelve OptiTrack Flex:V100R2 cameras. The capturing volume was approximately $2.5m \times 1.5m \times 1.5m$, Both systems sampled their data at 100Hz. The linear accuracy was tested using a reference frame which was measured using a Faro scanning arm. The OptiTrack system produced slightly higher error levels than the Vicon system, the maximum absolute percentage error of the OptiTrack system was 0.84%. moreover, for both systems no percentage absolute error was more than 1% (Thewlis et al., 2011).

2.2.5 Applications-Uses

Motion Capture systems started as a tool that analyses subject's movements in the biomechanical field. Afterwards, it became a very important tool in the animation industry. Below are some examples of the most wide-spread motion capture applications.

- Video games: In order to animate in-game characters. Sega Model 2 arcade game Virtua Fighter 2 was the first video game to use motion capture systems (Wawro, 2016).

- Movies CG effects: Replaces traditional animation by generating creatures and movie characters. One of the most famous examples is Avatar.

- Gait Analysis: Used in conjunction with an analytical software by physiotherapists, orthopedists, and neurologists in order to evaluate patients' status and rehabilitation by measuring several human biometric factors (Pfister

et al., 2014; Rojas-Lertxundi et al., 2017; McPherson et al., 2017).

- Sports: Studying all the actions of players. It improves players techniques for better results in different sports activities.

2.2.6 Summary

Optical motion capture is the most appropriate method for surgical navigation. Current systems use two cameras and have limited coverage. Vicon is the leading biomechanical optical system but is costly. OptiTrack may provide a cheap but accurate alternative. A current optical tracking camera system such as the Polaris costs $35000. For one third of this price a 12 camera OptiTrack system can be purchased and would provide much greater coverage of the surgical field without markers being obscured or the need to reorder the theatre when changing from left to right legs. The cameras could be housed in a cleanable fixation device and attached to the inside of the laminar flow hood. Such a system would be cheaper but have more utility than the current two camera stands and would leave the operating theatre less cluttered with wires and stands.

2.3 CNC

CNC stands for Computer Numerical Control. CNC machines typically form an object by carving it out from a solid block of material; such as wood and aluminium (Hood-Daniel & Kelly, 2009). As the name implies, CNC uses a computer as a means to control the machine itself throughout the carving process. Using computers allows for designing the target product before the carving process itself, and it also allows for specifying the way by which the machine would do the cutting.

First, in order to design the product, a Computer Aided Design (CAD) is produced. Then, the user specifies how the machine should do the cutting (the cutting path) by generating a Computer-Aided Manufacturing (CAM) file. At this step, the computer's role is to interpret this CAM file into signals sent to the

CNC machine to follow.

2.3.1 Types of CNC Machines

CNC machines can be classified into the following categories:

- Routers: They are the most widely used CNC machines, they cut through relatively soft materials such as: plastic, wood, and aluminium. Routers have a spindle that moves in an XYZ configuration. They operate at high speeds, around 18,000$RPMs$ (Heisel & Krondorfer, 1997).

- Milling Machines: Their operation is similar to CNC routers. However, their purpose is to cut through harder materials such as metals, also they are much slower than CNC routers, at around 1,000$RPMs$. Another difference is that milling machines make use of a cutting table that generally moves in an XY configuration while the machine's spindle moves on a linear axis (Z) above the piece (Altintas, 2012).

- Lathes: In contrast to routers and milling machines, in Lathes the work piece is rotated, while the cutting tool only controls the depth of cut. Therefore, Lathes are mostly used to form cylindrical or spherical surfaces and creating symmetrical pieces (Altintas, 2012).

- Plasma cutters: They use hot plasma to cut through electrically conductive materials. They work by sending an ionized, high powered stream of gas through a nozzle creating an electric arc which heats the gas and converts it into plasma. This plasma melts the metal and clears away the metal debris (Keraita & Kim, 2007), (Iosub et al., 2008), (Kolarevic, 2001).

- Laser cutters: They project a laser beam on the work piece which burns through the material. They can cut through a wide range of materials such as wood and plastic. Compared to plasma cutters, they consume less energy and result in better precision (Kolarevic, 2001).

- Waterjet cutters: A jet of highly pressurized water mixed with solid abrasive particles is streamed through a tiny nozzle eroding the material creating

very accurate cuts. They can cut up to 15-inch-thick titanium (Kolarevic, 2001)

- Spark machining: Cutting is performed using a series of electrical sparks generated across two electrodes with dielectric fluid in between them and an electric voltage applied across them. Spark machining is generally targeted to cut through hard metals, therefore it is widely used in aerospace, automobile, and electronics industries (Jain et al., 1999).

Most CNC machines have a local controller or some kind of a micro-controller which communicates with a central computer. As long as the right code is written to program the CNC machine, any complicated cuts can be achieved.

2.3.2 Cutting Burrs

Burrs are rotary cutting tools that can be used with hand-held tools or CNC machines. They consist of two parts: stationary elongated outer tube and rotating elongated inner tube. These tubes are made of stainless steel.

Many factors are considered when planning a bone drilling or milling operation (Dillon et al., 2016), including:

- Cartesian path: A 3D cutting path covering the targeted area is generated avoiding any un-targeted areas.

- Cutting angle: Cutting efficiency increases when surgeons use the side of the burr whenever it is possible instead of its distal tip, this is due to its spherical shape. Cutting with the distal tip generate greater force spikes (Dillon et al., 2013).

- Cutting force: When cutting near vital anatomical structures, such as nerves, it is recommended to reduce the cutting force to prevent any possibility for the burr to deviate from its planned path and also lowering the force reduces the amount of heat generated hence protecting the vital structures from any heat damage.

- Cutting velocity: The velocity by which the burr cuts depends on two main factors: the density and volume of the bone area being cut, and orientation of the shaft. Lower velocity level is recommended when cutting through bones of higher density and volume. Also, lower velocity is recommended when the burr is not well-oriented for efficient and accurate cutting.

2.3.2.1 Burr Categories

Surgical burrs are available in a variety of different shapes, sizes, and material. Many companies provide various types of burrs such as Delta (Delta Surgical, 2018), Brasselers (*Brasseler USA - Medical*, 2018), and MERICAN (*Surgical Burs and Drill System*, 2016).

The head of the burr can be in many shapes, such as: spherical, cylindrical, tapered with flute in cutting edges, flutes (Chen et al., 2017). Below is a list of burr categories according to their material:

- Stainless steel burrs
- Diamond burrs
- Carbide burrs
- Coarse diamond burrs
- Conical carbide burrs
- Diamond burr for Osseostap

2.3.3 Alternatives to Burrs

One of the drawbacks of the burr is that it can take sometime to remove the bone. This is particularly problematic in TKR when a large volume of bone must be removed. Stryker when they acquired Mako introduced a stiff thick blade to remove the bone rapidly in their TKA application. This blade could also be used in the future for UKA although currently it is not implemented in the UKA application and would have the draw back of not been able to cut a curved

surface which is one of the advantages of the current Restoris implant in that it is maximumly bone conserving and this will not be the case should a saw be used.

2.4 Three-Dimensional Scanning

3D surface scanning is the process of obtaining digital 3D surface data of objects. It can be used in many fields such as medical implants and devices, aerospace, industrial, and automotive applications. Healthcare services have greatly benefited from the advancements in 3D surface scanning as 3D scanners have been able to create 3D internal images of patients body (Weyrich et al., 2004),(Treleaven & Wells, 2007).

3D scanning can be categorized according to the technique by which it captures data. They can be categorized into four main categories (Allard & Lavoie, 2014):

- Measuring arms, portable Coordinate Measuring Machines (CMM) and measuring arms are equipped with a probe wither fixed or touch-triggered. Measurements are taken when the machine senses the contact of the probe tip with a surface. Their advantage is that different tools can be mounted on the arm. However, the limitations are that they need to be fixed on a surface, and also not all objects shapes can be scanned.

- Optically Tracked 3D scanners: This type of scanner utilizes an external optical tracking system to establish positioning. They provide very accurate and precise measurements. Their portability makes them favourable for many applications. On the other hand, a clear and direct line of sight is needed during the whole scanning process.

- Structured-light 3D scanners: These scanners project a pattern of light onto the measured object and then processes how the projected light pattern is distorted. their advantage is the very high-quality scanned data which provides scanning of the smallest details on the measured object. However, in

order to achieve this high resolution, a single scan acquires large quantities of data slowing down the processing step.

- Portable 3D scanners are now the most available 3D scanners on the market, either projecting laser or white light. Portability makes them easier to use compared to other types. They can scan more than half a million points per second. However, self-positioning errors can stack up as the scanning volume increases

3D scanning systems can work either by using white light or by using structured light. Also, they can be classified according to their portability (Allard & Lavoie, 2014).

The scanned data are represented as a point cloud, afterwards they are brought into a common reference system in order to be merged into a complete model. Different computer software programs (such as: Geomagic (3D Systems, 2018)) can be used to process the scanned data, for example by filling in the missing points, and correcting scanning errors.

In (Eder et al., 2013), six scanners (Minolta Vivid 910, Polhemus FastSCAN, GFM PRIMOS, GFM TopoCAM, Steinbichler Comet Vario Zoom 250, 3dMD DSP 400) based on different scanning principles were used to measure five different sized sheep skulls. Figure 2.1 shows the results of scanning using the six different 3D scanners. It can be seen that sub-millimetric accuracy can be achieved using these scanners.

Table 2.1: Manufacturers technical data of the tested scanners

3D scanner	**Accuracy** (mm)	**Scanning interval** ($second$)
Minolta Vivid 910	0.068	2.5-0.3
Polhemus FastSCAN	± 1	30
GFM PRIMROS body	≥ 0.03	1.5
GFM TopoCAM	0.025	≥ 20
Steinbichler Comet Vario Zoom 250	± 0.04	Not specified
3dMD DSP 400	<0.5	0.0008

2.4.1 Examples of 3D scanners

2.4.1.1 FARO

The company FARO has designed and developed the most trusted portable 3D measuring arms for more than 35 years. They have become the global standard for arm technology that meets the diverse consumers requirements. The most recent product line is FARO Quantum which is the first arm on the marker that can be verified against the international certification standard (FARO Technologies, 2018a).

The system can be integrated with FAROBlu laser line probe to scan 5 times faster and provide non-contact scanning capabilities. At full field of view, accuracy of the FARO laser line probe is $\pm 25\,\mu m$, with a scan rate of 300 frames/second, 2,000 points/line, and therefore 600,000 points/second (FARO Technologies, 2018b)

2.4.1.2 SMARTTECH

SMARTTECH is a well-known company expert in 3D measurement products. It has two product lines. SMARTTECH 3D portable combines the best features of both portable and stationary 3D scanners as it is an accurate ultra-fast 3D scanner that can also perform colour measurements. The duration of a single measurement does not exceed 0.2 seconds (SMARTTECH, 2016b).

Scan3Dmed uses white LED structured light, so it needs no laser or physical contact during the scanning. The point cloud scanned using this device has an accuracy up to 0.01 mm (SMARTTECH_Co.Ltd., 2018). One of its main advantages is that very quick as it is able to scan over a million points representing the scanned surface within 0.7 seconds, also because of the SMARTTECH software, the scanned data can be calculated and analysed without extra delay. Moreover, the scanning head can be integrated with up to 5 other 3D scanning units enabling scanning from different angles. (SMARTTECH, 2016a).

2.4.2 Summary

Due to the complexity of the human body, getting a virtual image of any anatomical structure is a hard task. Whatever the medical application is, when a measurement of a patient's anatomical structure is needed, a safe and contactless 3D scanner is needed to produce an accurate image of the target anatomical structures. 3D scanners are used to either generate anatomical 3D models of the human body or create implants to simulate surgical operations. There are different categories of 3D scanner, the best category is chosen according to the scanning environment requirements such as portability, scanning speed, and accuracy.

Chapter 3

Aims and Objectives

The aims of this project were to develop a prototype remote controlled surgical robotic system with sufficient accuracy (be able to cut shape as close as possible to the original shape of the implant) and to perform unicompartmental knee arthroplasty and more affordable than other existing robotic systems.

In order to achieve this aim an inexpensive motion capture camera system will be used to create a multi-camera in-theatre navigation system, once established this system will be tested for the required accuracy (Chapter 4). The next step will be designing an accurate robotic CNC machine which will be responsible for the cutting phase in the surgical process (Chapter 5). Then a series of applications will be written to perform the navigation and control of the CNC robotic machine using the selected motion action system, the machine will also be controlled remotely in the theatre (Chapters 6 and 7).

The system should perform the bone cutting phase in the surgery. As the surgeon plans the position and orientation of the implant on both femoral and tibial condyle. The operator/surgeon then starts the procedure by registering the knee (to inform the system where should it cut the implant shape). The system acquires some data input from working cluster by the motion capture system, then the system will orientate the implant shape, then plan a path for the CNC machine to burr the bones. After the system completes the process, the burr will be retrieved, the surgeon can then start the implant cementing and fixation phase.

After the building stage, the system will be tested on a set of artificial bones, so that the result can be compared to those produced by other robotic systems (Chapters 8 and 9).

Chapter 4

Methods: Validation of The Navigation System

4.1 Introduction

Motion capture systems track real-time movements of key points on a target object and then translate that data into sequences of Cartesian coordinates in three-dimensional space. Motion capture systems were initially developed to measure human movement but their progression has been greatly enhanced by the involvement of the animation industry (*What is motion capture*, 2017).

Nowadays motion capture serves a wide range of purposes such as computer games, animations films, validation of computer vision, robotics, and evaluating the functional performance of athletes, pilots, drivers, injured people, etc.
As detailed in Section 2.2, motion capture systems can be classified into two types: Marker-based and marker-less motion capture systems (Perrott et al., 2017). Marker based motion capture systems are sub classified into four types: acoustic, mechanical, electro-magnetic, and optical systems. Each of these types has an optimal deployment. However, in the proposed application of surgical navigation, optical systems are the preferred technique due to their high sampling rates and ability to give immediate feedback. Moreover, theoretically an unlimited number of reflectors can be used (Shiratori et al., 2011). Optical motion capture systems yield highly accurate tracking when compared to the alternatives.

The method depends on finding spatial correspondence for a detected marker in more than one image captured from different viewpoints simultaneously, such that each image must correspond to the projections of the same key point from each camera perspective. Thus, Triangulation of the various camera views is applied to recover the 3D positions of the markers, markers are tracked from one frame to another and these 3D marker positions are used to fit a model to the marked movement (Guerra-Filho, 2005b).

4.2 Aims and Objectives

The main objective of this section of the thesis was to determine if the OptiTrack (optical motion capture system) was a reliable navigation system and could be used to capture accurate locations of objects when they are stationary or moving as they would be during surgery. The proposed motion capture system (OptiTrack) offers a cheaper alternative to existing systems such as Vicon and a multi-camera option with better field of view when compared to existing surgical tracking systems like Polaris.

An accurate motion capture system was needed to provide tracking of the surgical tools and the thigh and shank segments involved in the surgical operation. Various modes of use of the motion capture system were required depending on the proposed task of the navigation system. An example was developing a three dimensional scanner. This application will be covered in Section 6.3.2. The experiments in this chapter tested the ability of the OptiTrack system to deliver the motion capture data with sufficient quality to achieve these tasks.

4.3 Methods

4.3.1 Introduction

Figure 4.1: OptiTrack Flex V100:R2 camera

Figure 4.2: Twelve Camera System Mounted

The motion capture system used consisted of twelve OptiTrack Flex V100:R2 cameras as shown in Figure 4.1. The cameras were linked together via OptiTrack's

software platform Motive (*Motive:Tracker - Motion capture and 6 DOF object tracking*, 2018a). Motive software sent the relative positions of each marker to the created tracking application in 3D. In the developed system twelve cameras were used. In typical surgical applications fewer cameras are used, usually 2 or 3. In these 2 or 3 camera systems, the view of the cameras can be blocked by objects in the field of view and the software will then lose track of the markers. The twelve OptiTrack camera system hardware costed $\$11,346$ (*Build Your Own Motion Capture System*, 2018). So, the OptiTrack system is cheaper, gives much greater coverage, with less obscuring of markers and can be wall mounted leaving the operating theatre uncluttered. The cameras were placed on the walls surrounding the operating table and configured in a "u" shape around the operating table head end, (Figure 4.2). In this way it is hoped that normal obstacles found in the operating theatre such as the operating table, anaesthetic stack and any other tools while blocking one or two cameras would not limit the available field of view of the system as a whole. Also, the "u" shaped arrangement provided a bigger capture volume with better avoidance of marker occlusion when the surgeon or the surgical assistant where in the field of view. While the setup in theatres varies all orthopaedic theatres will have a laminar flow hood on which the cameras can be mounted.

4.3.2 Motive Software and OptiTrack Calibration

Motive is a software which can reconstruct three-dimensional objects by processing multiple two dimensional images(*Motive Documentation*, 2016).First, Motive software had to be initialized in order to be able to perform the camera calibration, then network streaming settings had to be defined. The version of Motive used was V1.5.0 64-bit.

Figure 4.3: Camera Settings

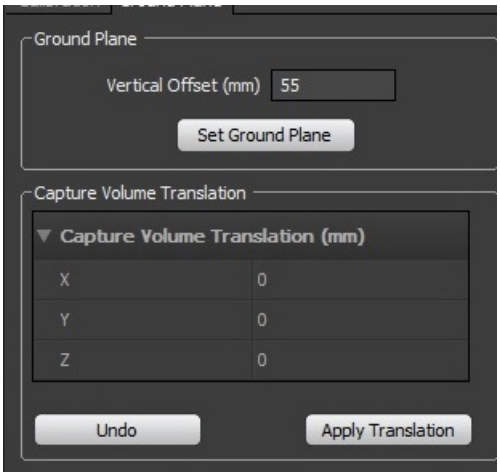
Initially the cameras' settings were set to accommodate to the environment of the theatre. The cameras' settings would be different from theatre to theatre depending on variables such as: the light intensity and the objects in the theatre. Figure 4.3 shows the cameras' settings for the work and experiments conducted in our operating theatre room.

The exposure setting is responsible for the amount of time the camera will acquire light per frame, this feature controls how bright the object (marker) would be. However, raising the camera's exposure to a high level makes the system pick up shiny objects as false markers. In the developed system, the exposure was set to 21 out of 480.

The threshold setting is the level of brightness below which the camera would discard any pixels. It is a very important feature, it could be levelled down to pick up missing markers that don't reflect the required amount of light due to different markers size or different light conditions in same theatre. However, too much levelling down of the threshold may cause the cameras to pick up shiny unwanted objects (like metal hand watches or reflective metal surfaces from working environment). In the developed system, the threshold was set at 200 out of 255.

The LED Illumination setting is responsible for adjusting the IR LED's; turning up the amount of IR illumination from the camera in the room may help in marker detection especially if the markers were far away from the cameras, but this may also cause the cameras to pick up shiny objects. In the developed systems, this feature was set to 15 out of 15 in which the camera only picked working clusters. The Frame Rate per Second was set to the maximum 100 to negate the motion blur effect and also enabled the running of the application in real-time (NaturalPoint, 2016). Potentially a fast capture rate could overload the visualisation software but this didn't occur at 100 frames per second.

(a) Wand Setting



(b) Ground Plane Settings

(c) Wand 500mm

(d) 55mm Ground Plane

Figure 4.4: Calibration Settings

The camera calibration process was simple and fast. It took a maximum of 15 minutes to complete the calibration process of the cameras in the developed system, this step can be performed by the system operator. Providing the cameras are not knocked or moved, then calibration is only required to be checked daily and this can be done in a matter of seconds, if the calibration check fails then the system can be recalibrated in 15 minutes maximum. It would be perfectly feasible to mount the cameras on the hood in such a way that they would not be

knocked or interfered with and hence wouldn't require recalibration frequently. To start the calibration process a new project was created in Motive software, then the Camera Calibration button was pressed. A calibration window opened on the right side of Motive. As shown in Figure 4.4a, at the OptiWand option, "large 500mm" was selected because for the experiment we used a large 500mm calibrated wand (see Figure 4.4c), the 500mm wand is a standard size, as there is three markers on top of the wand T-shape with known distinct distances in between, so that when the wand is in the wanding process (moving the wand to make oval shapes front of the cameras), the oval shapes are compared in perspective to each camera. The "Block Visible" option was used to block out all shiny objects that existed in the theatre prior to the calibration process.

Next, the wanding process was initiated by pressing the "Start Wanding" button, and then the wand was introduced to the field of view of the cameras and moved in a u-shape in the theatre using circular wand-like moves around the capture volume.

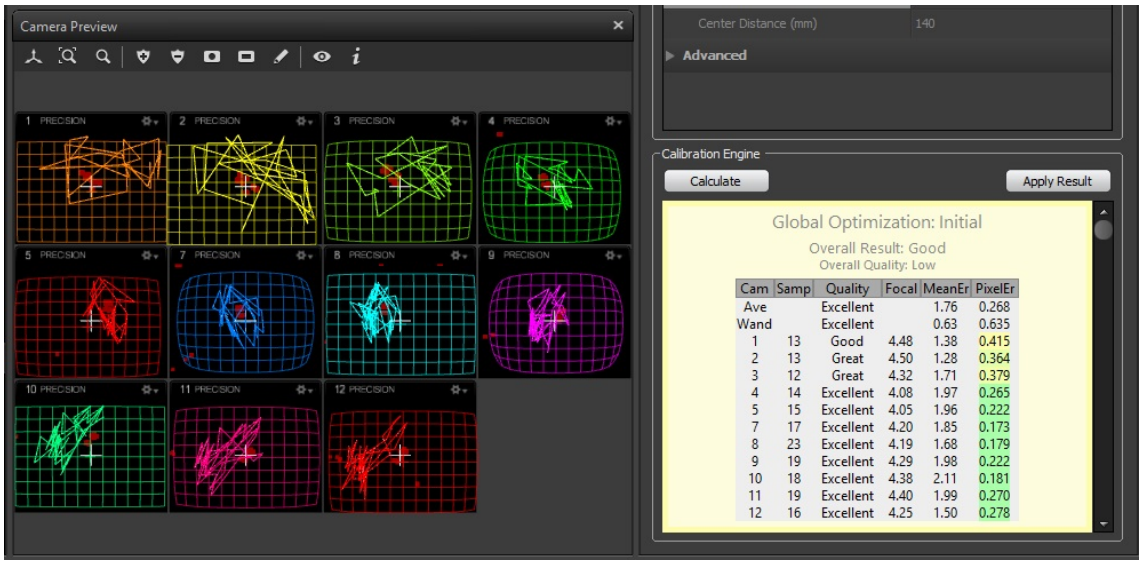


Figure 4.5: Wanding Input Screen

In order to achieve the best results, the wanding process was continued until all cameras had at least 2000 sample, if not the operator should focus the wanding process to face cameras with lower samples count. Then"Calculate" button was pressed, As is shown in Figure 4.5, the calculations values were displayed on

the right side of the Motive screen. In this example, cameras four to twelve have excellent quality, as the calculations continues the overall quality will changes, as the calculations was finished the overall quality was excellent and ready to apply the result message was shown on the screen. Then "Apply Result" button was pressed to finish. The motion capture system was now calibrated. A second process was to assign the system an origin (0,0,0 coordinates) for which the Ground Plane was used, (Figure 4.4d), the system recognized these two axes also because of the distinct distance between the three markers. The origin of the system was set to the intersection of the two arms of the ground plane with the short thick side (vertical in the picture) as the x-axis and the longer thin side. (horizontal in the picture) being the z-axis, with the y-axis vertical and orthogonal.

4.3.3 Network Settings

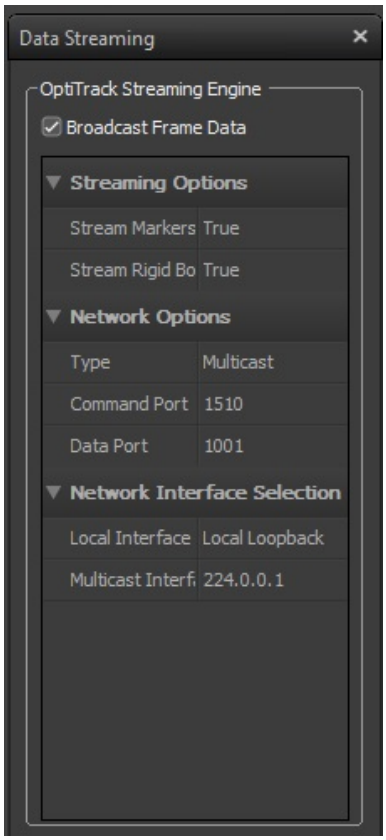


Figure 4.6: Data Streaming Setting

Once the initialization procedures had finished, Motive was ready to stream the markers coordinates to D-Flow, D-Flow is a software that provides great support for motion capture system as it provides visualisation to tracked markers and can pass their locations to programmable script modules. In Motive, the "Streaming Pane" button was pressed and the Data Streaming panel opened. The settings were set as the D-Flow application required (see Figure 4.6). First of all, the "Broadcast Frame Data" check box was checked, this tells the software to stream out its data. Then in the network options, the command port was set to 1510 and the data port to 1001. In the network interface selection, the local interface was set to "local loopback" and multicast interface was set to "224.0.0.1", this tells the Motive software where to stream the data so that other software can find it, which in this case this IP points to same computer.

4.3.4 Experimental Methods

Three experiments were undertaken to validate the OptiTrack navigation system. The first experiment was to capture a cluster of markers and compare the recorded data to the dimensions of the cluster, this experiment was to test the system accuracy.

The second experiment was designed to validate the accuracy of the system across the field of view of the OptiTrack in theatre. This was accomplished by recording the cluster dimensions in multiple positions around the theatre of operation.

Finally, The third experiment was to test the system's precision of capturing moving markers. This was the system's real-time performance and was tested by capturing multiple clusters moving simultaneously in eight separate scenarios.

4.3.4.1 Experiment 1: Testing Accuracy using the Known Shape of the cluster of markers

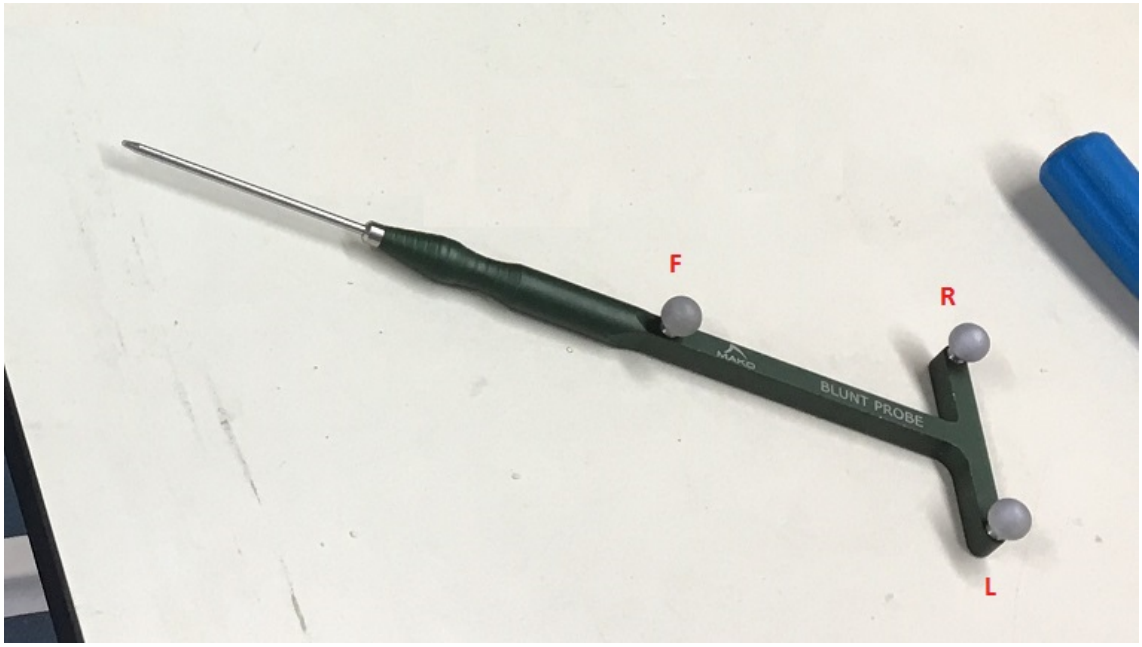


Figure 4.7: Blunt Probe

In the first experiment, data were recorded using the installed OptiTrack multi-camera system and Motive software and a Blunt Probe (see Figure 4.7 above) with a cluster of 3 markers on it. These data were then compared to the known dimensions of the probe. The marker XYZ positions were calculated by Motive software and then sent via the network streaming interface to D-Flow. A D-Flow LUA script module (LUA is a multi-paradigm dynamic language, where LUA means moon in Portuguese) was written to capture the incoming data and use it to calculate the Euclidean distances between the cluster's markers in real time, see Appendix A.1 for more details about the code. These calculated distances were then compared to another set of distances in which the marker positions had been carefully recorded by a Vicon Bonita 8 camera motion capture system. The result of the comparison can be viewed in Table 4.1, So accurate to within 0.3mm. This data showed concurrent validity with Vicon when the pointer is static and in middle of capture volume.

4.3.4.2 Experiment 2: Testing Range by Marked Distances in Theatre

Figure 4.8: Testing Field of Camera View in Theatre

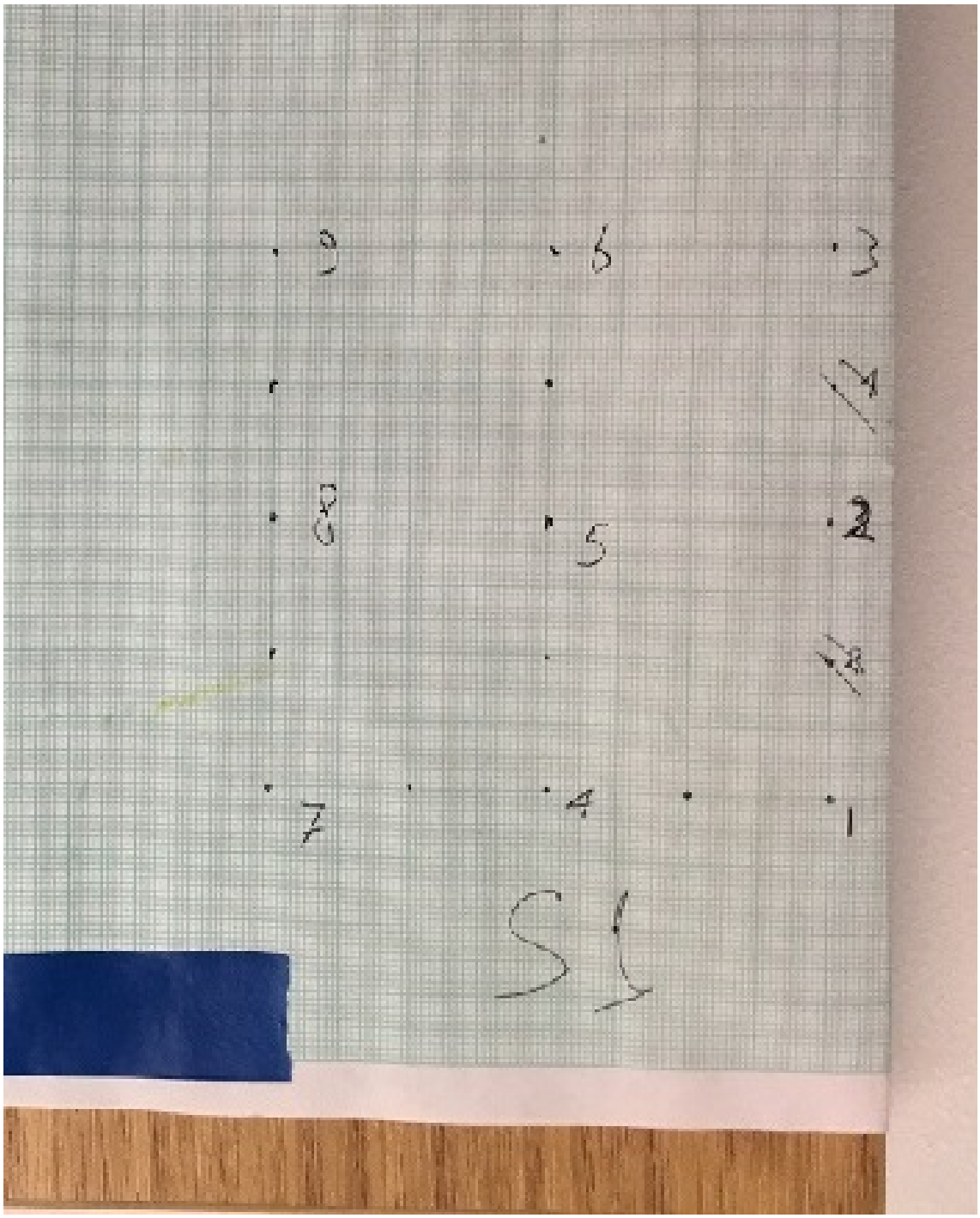


Figure 4.9: Testing Range by Graph Paper

The motion capture system accuracy may be dependent on the position within the field of view of the system, thus when the markers reach the borders of the OptiTrack system, they may become less accurate. Experiment two was performed to insure sufficient accuracy of the navigation system across the field

of view (active range) in the theatre of operation. This experiment was performed in two steps.

The first step was to frame the space at which markers needed to be observed. This was achieved by building a rectangular frame of suitable dimensions, see Figure 4.8. The space inside the frame represented the required area for the clusters to function; whether stationary or moving. The dimensions of the cuboid was 1.5m x 1m x 0.6m.

The second step was to test the system's accuracy in the space inside the frame. This was accomplished by placing the blunt probe in known positions within the field of view of the system. Graph paper was marked by an array of nine points in three columns and three rows, each had a $40mm$ separating space (Figure 4.9). The graph paper was then mounted on a flat wooden surface, this surface was placed on the base/ground side of the cuboid. The Blunt Probe was then used to locate each point in the points array in a specific order, the probe ws pointed in each location by hand.

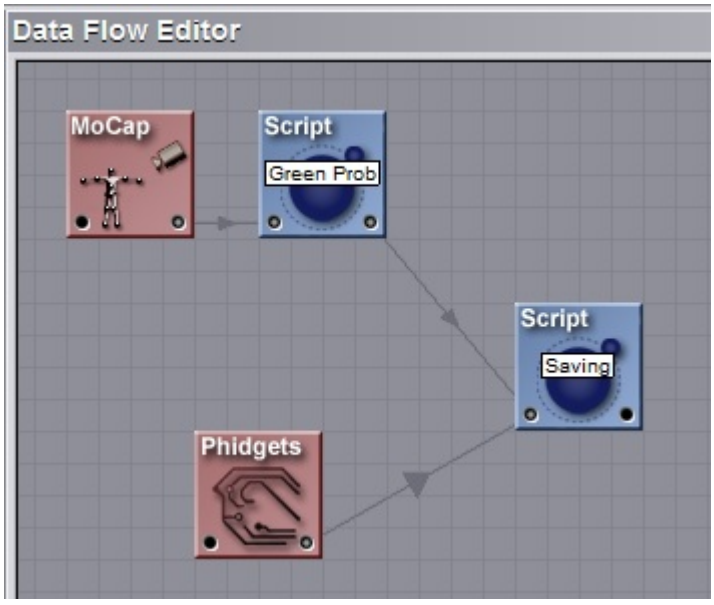


Figure 4.10: Testing Application in D-Flow Editor

A special application was written in D-Flow using a LUA script to map the points from the graph and store them in order in a file, see Figure 4.10 for the flow editor view in D-Flow studio. Two scripts were used to perform these tasks, the

first script was the Green Probe script which retrieves the pointer's tip location this was linked to the second script (Saving) which was used to store each pointer markers and calculate the location in order when the foot switch (phidget) was pressed. A voice assisting feature was built in to the code to assist the recording operation. The first script (Green Probe) will be discussed in details in Section 6.3.1. For the other script (Saving) see Appendix B.5 for more details.

The location of the tip of the probe relative to the three fixed markers was recorded prior to operation. This data was then used to reconstruct the location of the probe tip from the locations of the three markers during operation. This is known as a virtual marker.

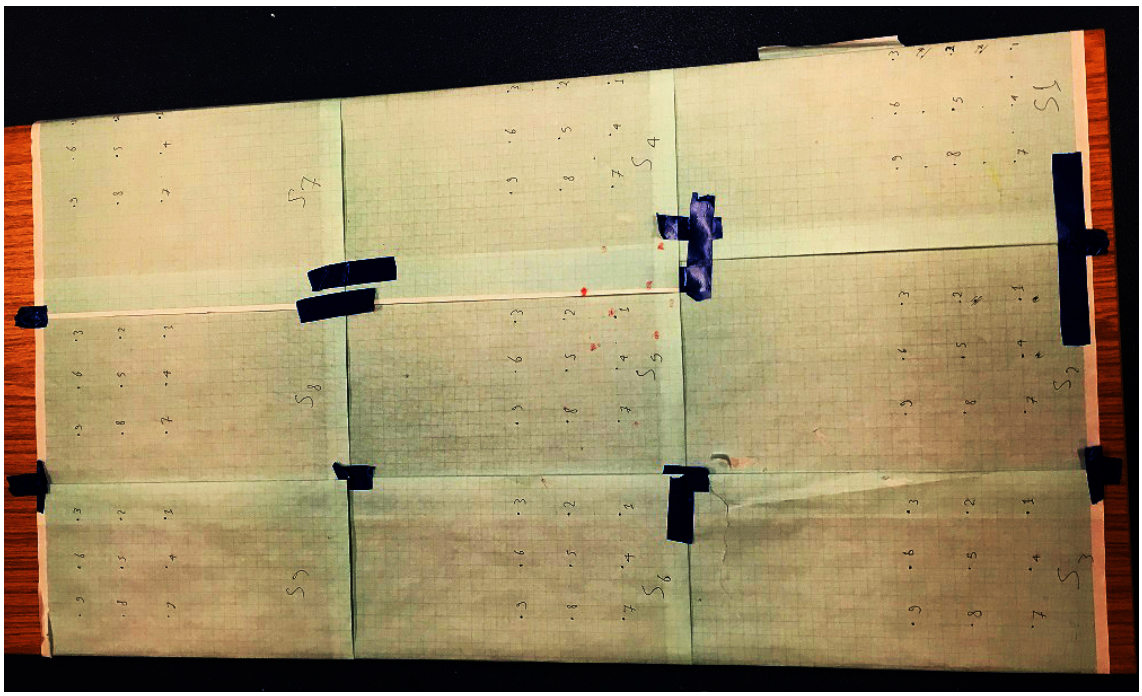


Figure 4.11: Testing Application in D-Flow Editor

(Figure 4.11) nine arrays of nine points were attached using tape to a wooden board. The nine points were numbered in order and each set of points on each paper was named a segment, so the figure shows one used segment. The data retrieved from measuring the nine segments on the cuboid floor is shown in Table 4.2. The Data for the whole level 1 (at Minimum height/ground level) is shown in Table 4.3 and level 2 (at maximum height/the wooden surface was placed at the top side(roof) of the cuboid) is shown in Table 4.4.

4.3.4.3 Experiment 3: Testing Precision and Real-Time Performance by Identifying Multiple Moving Clusters in eight different scenarios

In the next experiment six of Mako's clusters were used, four of them were moved in field of view (the Femur, the Tibia, the Blunt probe and the Sharp probe clusters), all these four clusters were required for the surgical procedure. The remaining two remained stationary all the time (Base cluster and the Endeffector cluster), all these clusters were first fixed on a table then picked up and moved within the camera system field of view. The Tibia and Femur clusters have four markers on them so that there is some redundancy for a missing marker and the Blunt and Sharpe probes have only the minimum required to track an object in 3D space of three non collinear markers. As Mako made the markers positions with unique distances between each other's, the tracking process was based on finding these Euclidean distance between the markers. The clusters with four markers have six distances between them and the clusters with three markers have only three distances between the markers. To test the tracking ability of the navigation system with moving clusters, the Euclidean distances between markers for each cluster were saved from a static recording and then again while the cluster was moving and then the two sets of data were compared. For example in Figure 4.12 below the cluster's markers locations was saved while the cluster was static. The cluster was then moved and its motion and markers coordinates were recorded by the Motive software while it was moved. The captured data was played-back and paused to capture marker coordinates at various locations for the cluster's four markers, see Figure 4.13. The Euclidean distance was then calculated and compared, (Table 4.5 and Table 4.6). Finally a single cluster's motion was tested inside the cameras field of view experiment scenarios(1 - 4) . Next, one cluster's motion was tested while there was another cluster/s in the cameras field of view scenarios(5 - 7), in scenario 7 all Mako's clusters were used in field of view, five were stationary. In scenario 8, two clusters motions were tested at the same time.

4.4 Results

4.4.0.1 Testing Accuracy using the Known Shape of the cluster of markers Results

Table 4.1: OptiTrack vs Vicon

Side	OptiTrack	Vicon	Unit
LR	58.3	58.0	*mm*
RF	96.7	97.0	*mm*
LF	105.7	106.0	*mm*

The naming of the sides was according to the markers naming abbreviation according to where they were placed relative to their base, so R is the right marker, L is left and F was the far from base marker.

4.4.0.2 Testing Range by Marked Distances in Theatre Results

Table 4.2: Segment 1 Scanned Points in *mm*

Point	**X**	**Y**	**Z**	**Euclidean**	**Expected**	**Error**
Point 1	357.3	21.8	102.4	57.4	56.5	0.9
Point 2	353.9	21.9	62.6	40.9	40.0	0.9
Point 3	351.5	21.6	22.9	57.5	56.5	0.9
Point 4	316.4	22.1	106.1	40.5	40.0	0.5
Point 6	311.3	21.8	25.7	40.0	40.0	0.0
Point 7	277.6	23	108.4	55.4	56.5	1.0
Point 8	274.3	22.4	67.8	38.8	40.0	1.1
Point 9	271.6	21.8	27.7	56.2	56.5	0.2
Average						0.7

The data in Table 4.2 above shows the data from segment 1 scan in the first level of the scanning process. The first column is the points order. The second, third and fourth column represents the XYZ coordinate value of each point. The

fifth column is the Euclidean distance between each point and point number 5 which is the centre point of each segment. The sixth column is the correct value of the Euclidean distance between each point and point number 5 based on the graph paper distance, for example point 1 has the distance of $56.5mm$ to point 5 which is the base of a two equal sided triangle of $40mm$, also points as 2, 4, 6, 8 have a direct distance of $40mm$. The seventh and last column was the absolute error column which is the absolute difference between the calculated Euclidean distance (fifth column) and the expected distance (sixth column) values. Average absolute error in the segment above is $0.6mm$. it should be noted that the points on the graph paper were hand drawn. 4.3 shows the average absolute error for all nine segments in a level for all nine points with the overall average absolute error in the bottom right hand corner.

Table 4.3: Level 1 Average Absolute Error in mm

Segment	**P1**	**P2**	**P3**	**P4**	**P6**	**P7**	**P8**	**P9**	**Average**
Segment 1	0.9	0.9	0.9	0.5	0.0	1.0	1.1	0.2	0.7
Segment 2	0.6	0.5	1.1	0.4	0.0	0.0	0.1	0.3	0.4
Segment 3	1.3	0.2	0.6	1.4	0.2	0.0	0.2	0.3	0.5
Segment 4	0.1	0.0	0.4	1.8	0.1	1.1	1.5	0.4	0.7
Segment 5	0.5	0.5	0.0	0.5	0.7	0.2	0.1	0.5	0.4
Segment 6	1.0	3.5	0.5	0.2	0.3	1.0	2.8	2.3	1.3
Segment 7	1.9	1.4	1.5	0.1	1.0	0.8	1.5	1.5	1.1
Segment 8	1.4	0.7	1.5	0.0	0.5	0.0	0.2	0.5	0.6
Segment 9	1.1	0.1	1.8	0.1	1.4	0.5	0.4	2.6	0.9
Error Range									**3.5-0.0**
Overall Average Absolute Error									**0.6(0.3)**

The Average error for the whole level was $0.6mm$ and the Standard Deviation was $0.3mm$. The wooden board was then moved the maximum height inside the cuboid frame and the same process repeated. Table 4.4 presents the data in the second level scan:

Table 4.4: Level 2 Average Absolute Error in mm

Segment	P1	P2	P3	P4	P6	P7	P8	P9	Average
Segment 1	0.4	2.9	3.1	0.8	0.1	2.7	3.8	2.7	1.9
Segment 2	0.3	1.0	2.3	1.3	0.4	1.4	2.0	1.3	1.2
Segment 3	0.3	1.8	2.6	0.3	3.3	3.5	5.4	4.2	2.5
Segment 4	0.2	0.8	2.1	0.2	0.8	1.5	0.9	0.3	0.9
Segment 5	0.2	0.0	0.4	0.0	0.7	0.4	0.7	0.8	0.5
Segment 6	1.9	7.5	7.7	0.9	1.0	3.5	9.0	7.3	4.4
Segment 7	0.2	1.9	0.8	0.6	0.1	0.9	0.4	0.0	0.7
Segment 8	0.2	0.0	0.2	0.8	0.3	0.1	0.3	1.1	0.4
Segment 9	1.9	1.2	0.8	0.6	0.1	1.8	2.2	1.3	1.2
Error Range									**9.0-0.0**
Overall Average Absolute Error									**1.5(1.2)**

Average absolute error for the whole level was $1.4mm$ and the Standard Deviation was $1.2mm$. Data from intermediate levels shows intermediate sizes of overall average absolute error ranging from 0.6 - $1.4mm$ (SD ranging from $0.3 - 1.2$). Hence at the level of the surgery the level of the origin the system was accurate to $\pm 1.2mm$ and at extreme of the field of view $\pm 2.8mm$.

4.4.0.3 Testing Precision and Real-Time Performance by Identifying Multiple Moving Clusters Results

Scenario1: Femur Cluster

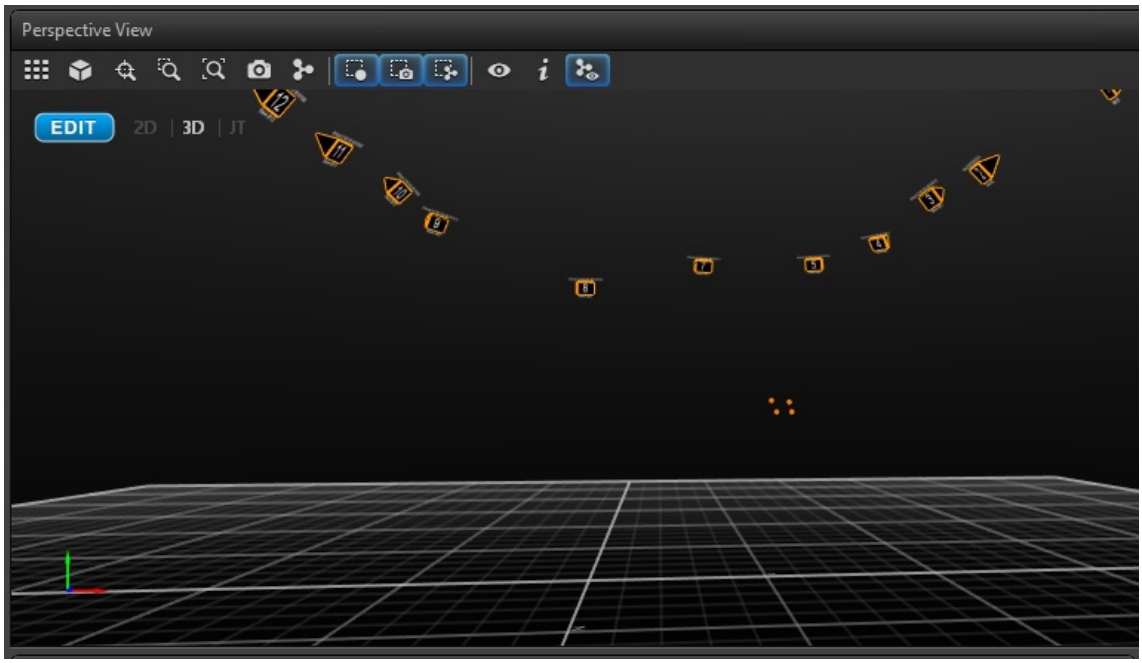


Figure 4.12: Femur Cluster Static

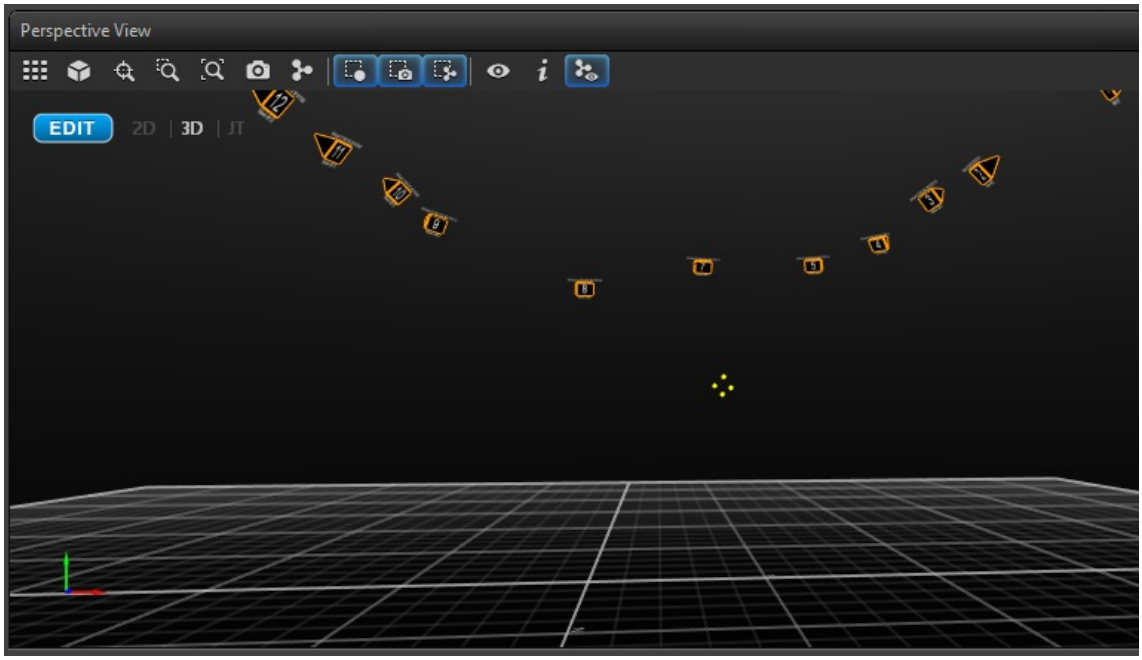


Figure 4.13: Femur Cluster Moving

Table 4.5: Femur Cluster in Motion Data

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	55.1	55.2	0.1
2	65.1	65.3	0.2
3	75.5	74.9	0.6
4	86.5	85.4	1.1
5	95.1	94.5	0.6
6	106.0	105.3	0.7

Table 4.6: Femur Cluster in Motion Extra Data

Measured	**Value**	**Unit**
Distance	0.511	m
Time	0.91	s
Velocity	0.562	ms^{-1}

The average absolute error in this experiment was 0.55 mm.

Scenario2: Tibia Cluster

Table 4.7: Tibia Cluster in Motion Data

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	53.8	52.7	1.1
2	60.7	60.8	0.1
3	69.0	68.3	0.7
4	74.2	73.2	1.0
5	96.1	95.9	0.2
6	126.7	125.7	1.0

Table 4.8: Tibia Cluster in Motion Extra Data

Measured	**Value**	**Unit**
Distance	1.226	m
Time	3.0	s
Average Velocity	0.408	ms^{-1}

The average absolute error in this experiment was 0.68 mm.

Scenario3: Blunt Probe

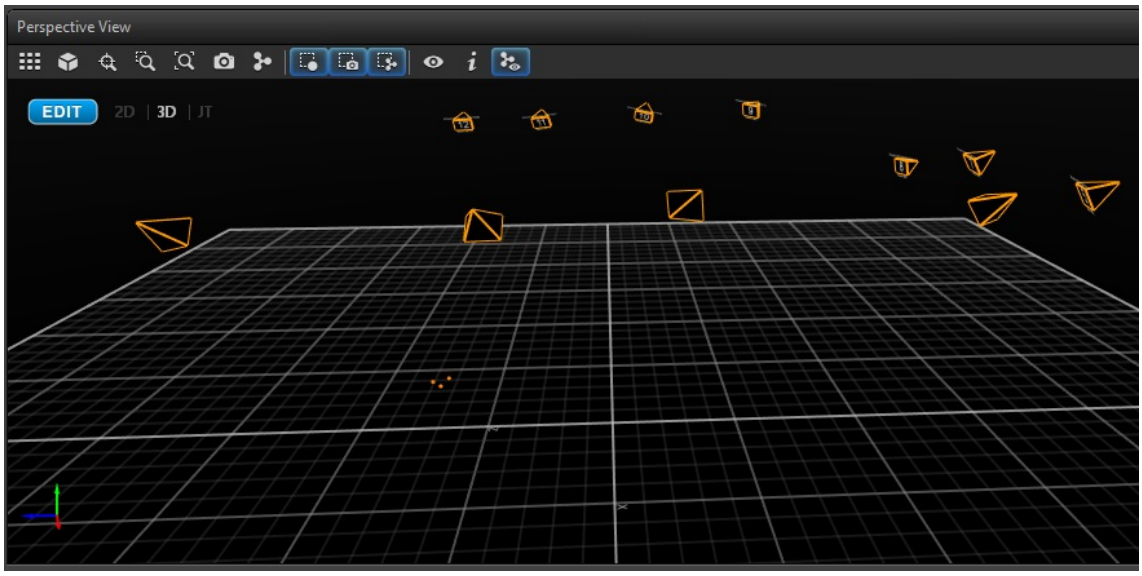


Figure 4.14: Blunt Probe Static

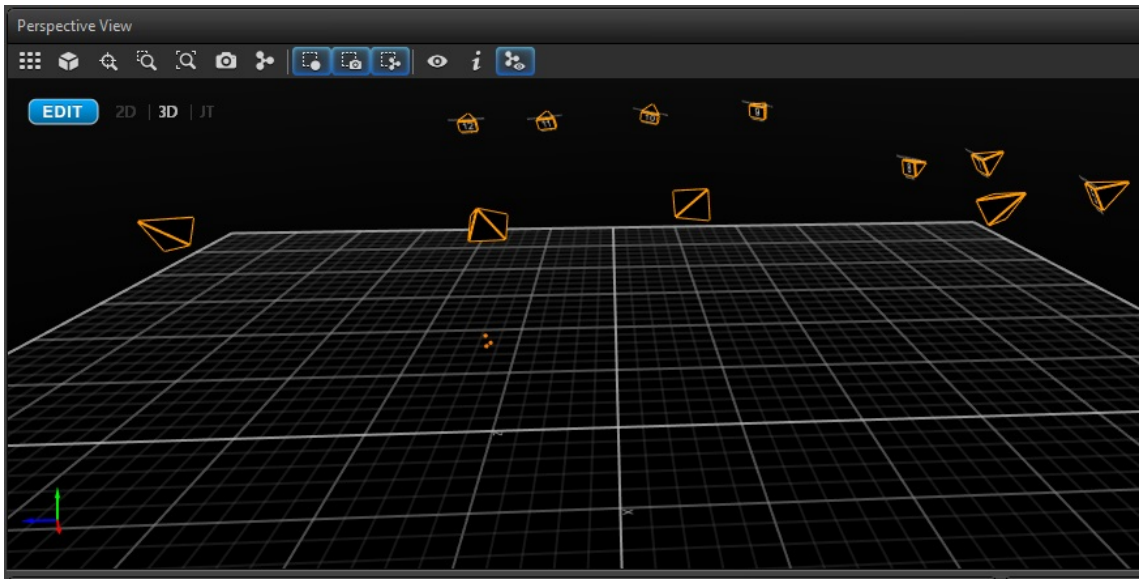


Figure 4.15: Blunt Probe Moving

Table 4.9: Blunt Probe in Motion Data

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	58.4	58.7	0.3
2	96.7	96.6	0.1
3	105.8	105.7	0.1

Table 4.10: Blunt Probe in Motion Extra Data

Measured	**Value**	**Unit**
Distance	0.475	m
Time	1.0	s
Average Velocity	0.475	ms^{-1}

The average absolute error in this experiment was 0.16 mm.

Scenario4: Sharp Probe Cluster

Table 4.11: Sharp Probe in Motion Data

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	61.5	61.8	0.3
2	107.4	18.6	1.2
3	126.6	127.7	1.1

Table 4.12: Sharp Probe in Motion Extra Data

Measured	**Value**	**Unit**
Distance	1.330	m
Time	3.2	s
Average Velocity	0.415	ms^{-1}

The average absolute error in this experiment was 0.86 mm.

Now the next four experiments include tracking of a cluster or more with another cluster/s in the cameras field of view.

Scenario5: Tibia Cluster with Femur Cluster in Field of View

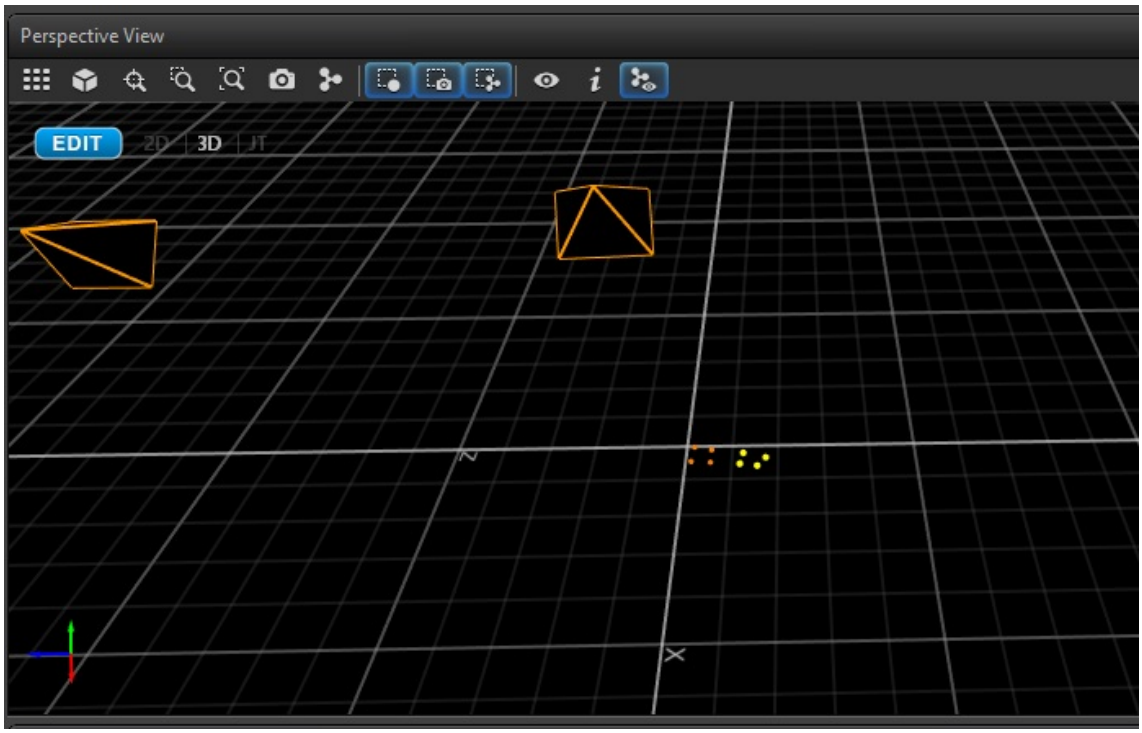


Figure 4.16: Tibia Cluster Static While Femur Cluster in Field of View

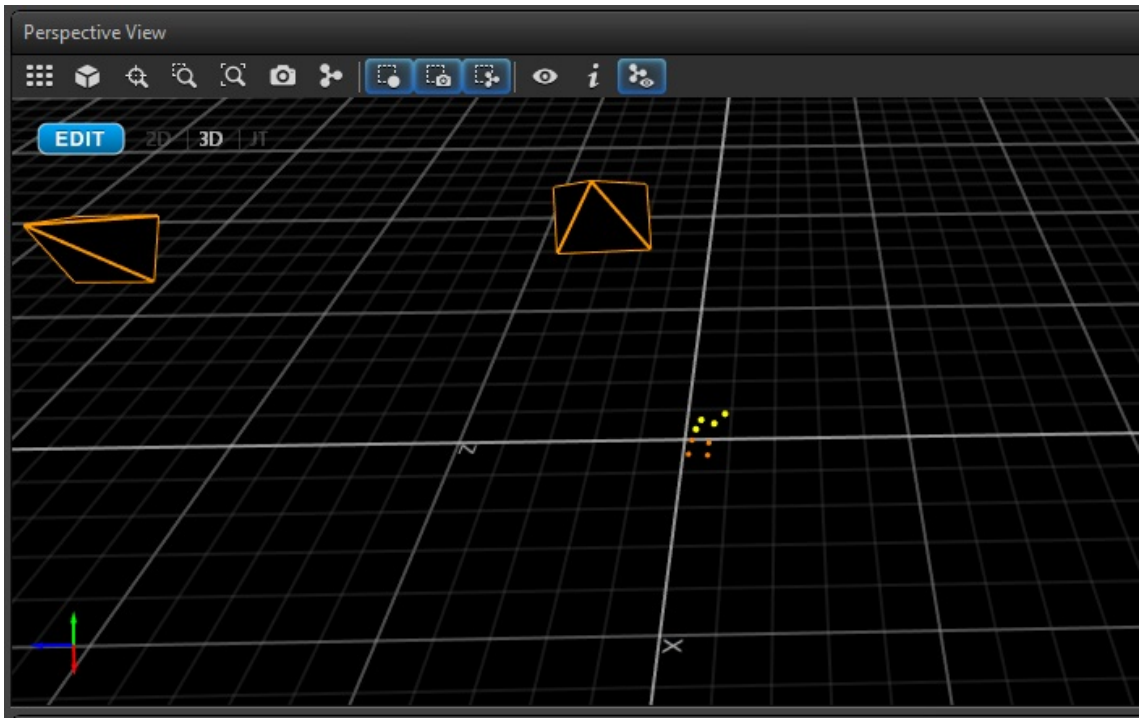


Figure 4.17: Tibia Cluster Moving While Femur Cluster in Field of View

Table 4.13: Tibia Cluster in Motion Data with Other Cluster in the Field of View

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	53.1	52.1	1.0
2	61.1	60.4	0.7
3	68.2	67.8	0.4
4	73.1	73.3	0.2
5	95.8	95.4	0.4
6	125.9	125.4	0.5

Table 4.14: Tibia Cluster in Motion Extra Data While Other Cluster in the Field of View

Measured	**Value**	**Unit**
Distance	0.223	m
Time	1.1	s
Average Velocity	0.202	ms^{-1}

The average absolute error in this experiment was 0.53 mm.

Scenario6: Blunt Probe with Femur and Tibia Clusters in Field of View

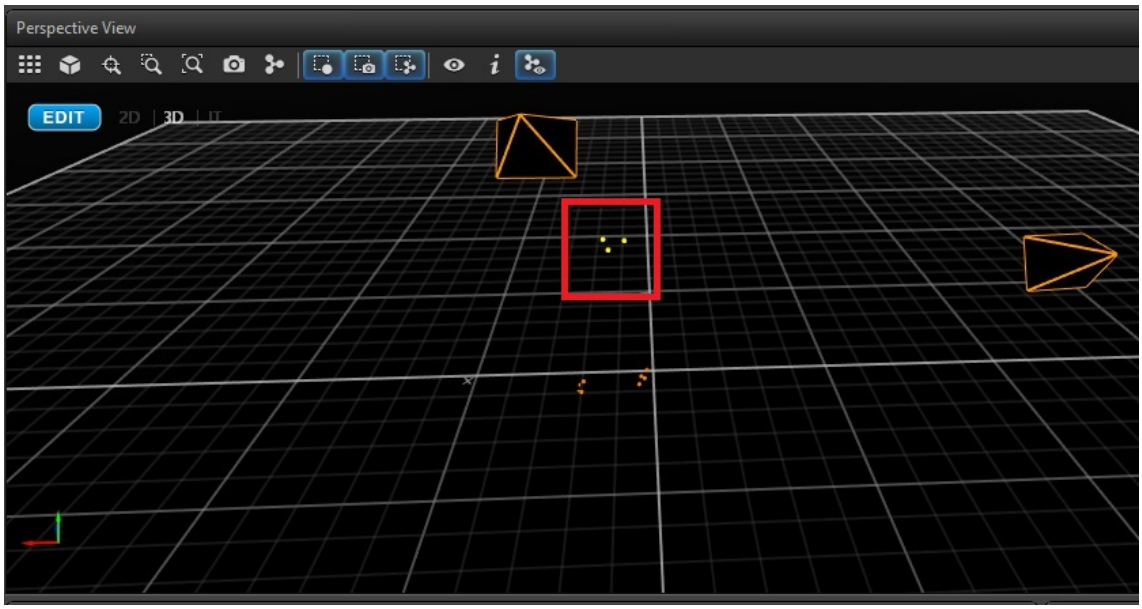


Figure 4.18: Blunt Probe Static While Femur and Tibia Clusters in the Field of View

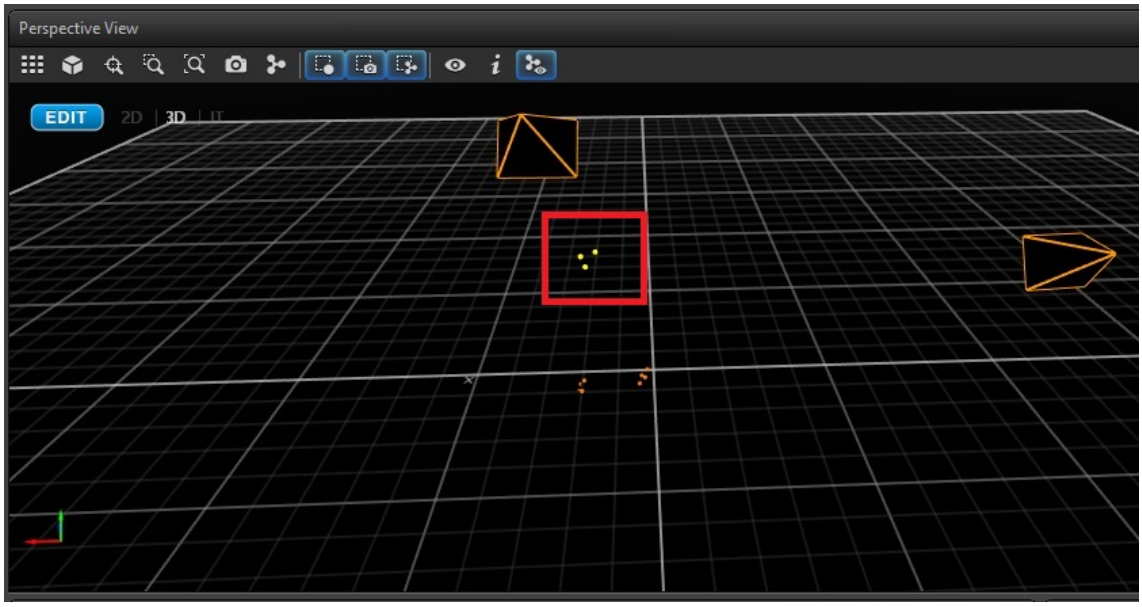


Figure 4.19: Blunt Probe Moving While Femur and Tibia Clusters in the Field of View

Table 4.15: Blunt Probe Blunt Probe in Motion Data with Other Clusters in the Field of View

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	58.5	58.9	0.4
2	96.0	96.0	0
3	105.3	105.2	0.1

Table 4.16: Blunt Probe Blunt Probe in Motion Extra Data While Other Clusters in the Field of View

Measured	**Value**	**Unit**
Distance	0.261	m
Time	0.6	s
Average Velocity	0.435	ms^{-1}

The average absolute error in this experiment was 0.16 mm.

Scenario7: Blunt Probe with all Mako Clusters in Field of View

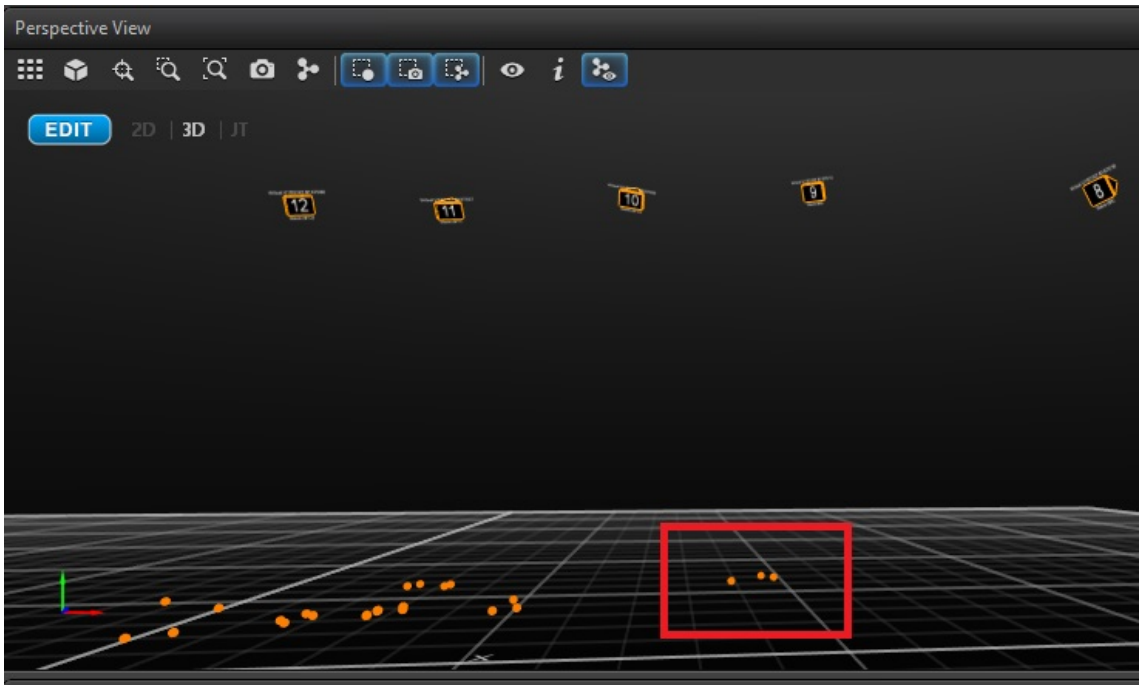


Figure 4.20: Blunt Probe Static while all Mako Clusters in the Field of View

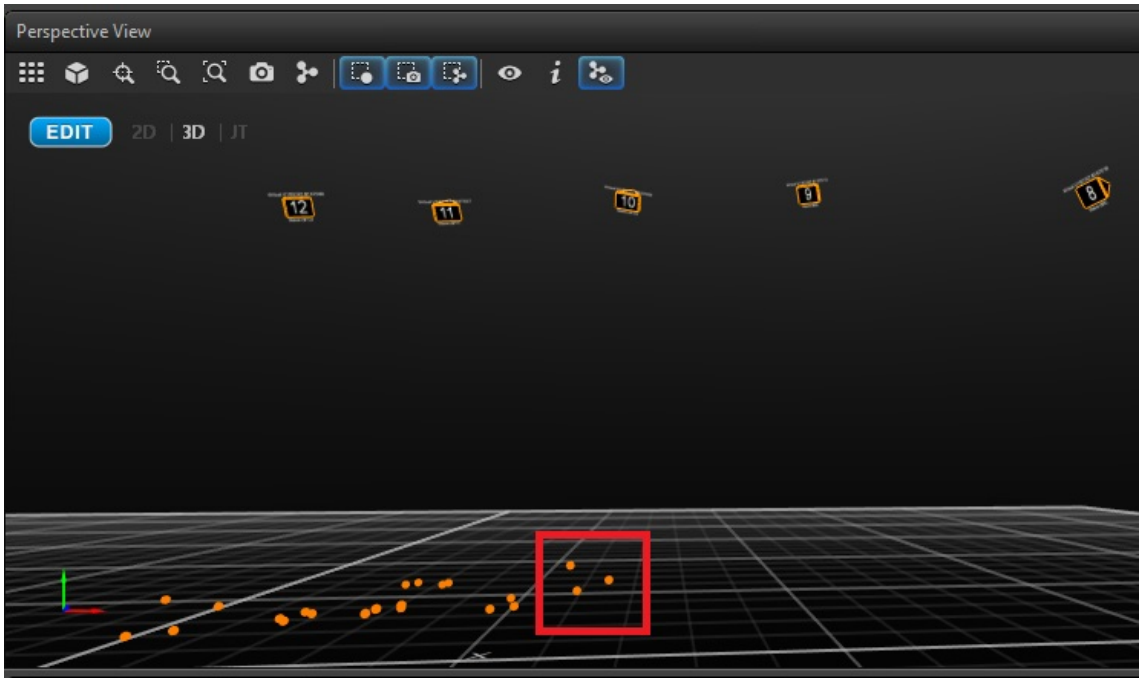


Figure 4.21: Blunt Probe Moving while all Mako Clusters in the Field of View

Table 4.17: Blunt Probe Blunt Probe in Motion Data with all Mako Clusters in the Field of View

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	73.1	73.9	0.8
2	120.1	120.8	0.7
3	131.3	130.6	0.7

Table 4.18: Blunt Probe Blunt Probe in Motion Extra Data While all Mako Clusters in the Field of View

Measured	**Value**	**Unit**
Distance	0.534	m
Time	0.85	s
Average Velocity	0.629	ms^{-1}

The average absolute error in this experiment was 0.73 mm.

Scenario8: Tibia and Femur Clusters Moving in Field of View

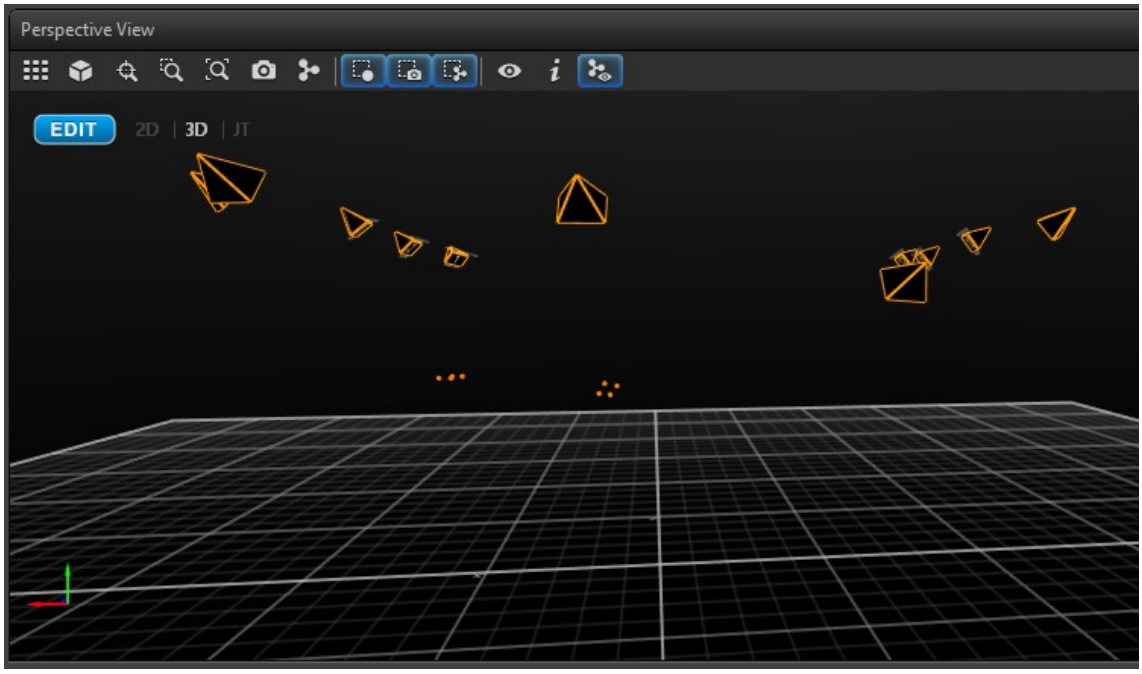


Figure 4.22: Tibia and Femur Clusters Static

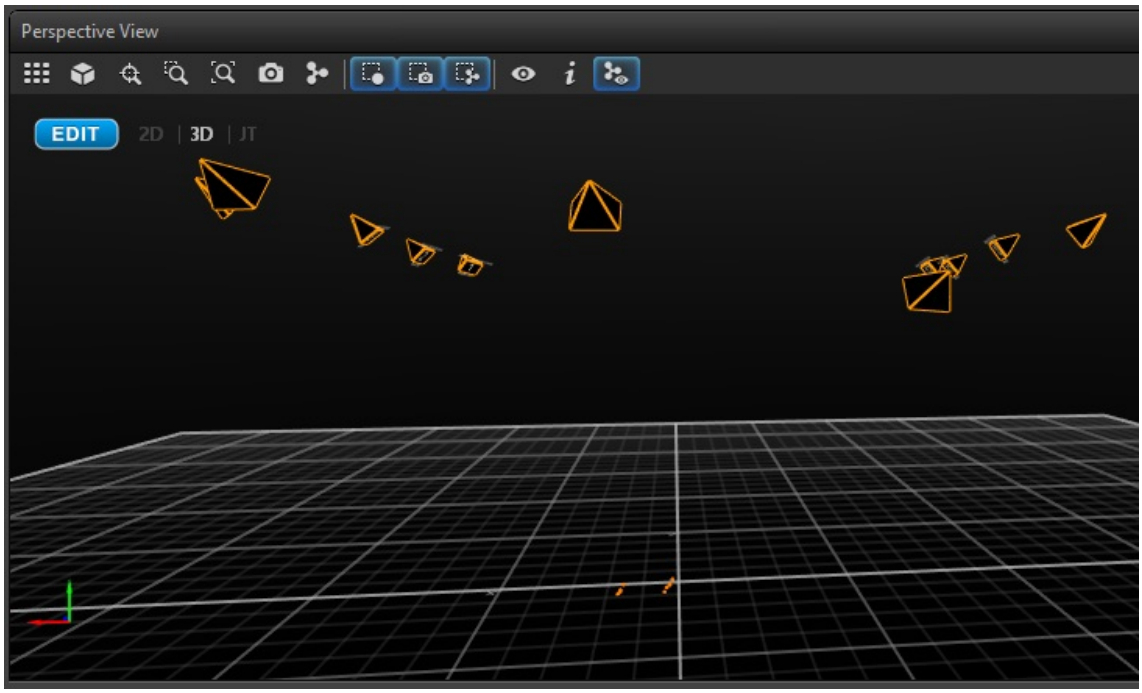


Figure 4.23: Tibia and Femur Clusters Moving

Table 4.19: Tibia Cluster Moving in Field of View Data With Femur Cluster Moving Around

1	**Static**(mm)	**Moving**(mm)	**Difference**(mm)
1	53.1	52.4	0.7
2	61.5	59.7	1.8
3	70.2	67.1	3.1
4	73.7	73.6	0.1
5	96.5	96.0	0.5
6	126.0	126.1	0.1

Table 4.20: Tibia Cluster Moving in Field of View Extra Data While Femur Cluster Moving Around

Measured	**Value**	**Unit**
Distance	1.484	m
Time	0.93	s
Average Velocity	1.596	ms^{-1}

Table 4.21: Femur Clusters Moving in Field of View Data With Tibia Cluster Moving Around

8B	**Before**	**After**	**Difference**
1	54.7	54.5	0.2
2	65.4	65.2	0.2
3	76.0	75.2	0.8
4	85.4	86.2	0.8
5	95.6	94.8	0.8
6	105.1	105.4	0.3

Table 4.22: Femur Clusters Moving in Field of View Extra Data While Tibia Cluster Moving Around

Measured	**Value**	**Unit**
Distance	1.429	m
Time	0.95	s
Average Velocity	1.505	ms^{-1}

The average absolute error in this experiment was 1.05 mm for the first cluster and 0.51 mm for the second.

4.5 Discussion

The Navigation system was a substantial element in this project hence testing the navigation system limitations. The three experiments that were performed finalized the testing phase. The first experiment showed that the navigation system accuracy was 0.3 mm different than the Vicon system, which is acceptable within the functionality of the full system as the cutting process was set to perform a 1 mm cut. The second experiment's accuracy at the origin of the system was ± 1 mm , while at the extreme ends of the field of view ± 2.5 mm. The third experiment showed the robustness of tracking the objects while moving in field of the cameras view. These data have made the OptiTrack valid as a cheap camera system that can perform as the system navigator.

4.6 Conclusion

Navigation systems are widely used in many applications. Motion capture systems are becoming essential in working with many robotic and biomedical practises. The accuracy provided by motion capture systems can prefect surgical operations. Choosing a cheap motion camera system and validating the system was a necessary step for this project. After testing, OptiTrack camera system was proven to be best fit for this project. The OptiTrack system when setup in a U-shape in the laboratory as if it was mounted on a laminar flow hood produce an accuracy of less than 1mm in the centre of the field of view and 2.5mm at the extremes which were nearly a meter from the centre of the field of view. We can conclude that for knee arthroplasty which will require a field of view of approximately 0.3m cubed and would generate an accuracy of $\pm 1mm$ and therefore the OptiTrack in this configuration would be suitable for knee arthroplasty operations.

Chapter 5

Methods: Overview of Design of CNC Machine

5.1 Introduction

The process of CNC machining makes use of computers to control cutting machinery and is widely used in the manufacturing sector in machine in machines such as mills, machining centres, lathes, turning centres, drilling machines, etc. In other words, CNC machining involves having machine tools that function through numerical control in which a computer program is written specifically to produce a certain object and in order to control the motion of the CNC machine while producing that object (*About CNC Machining*, 2017). CNC machines have revolutionized manufacturing turning products from handmade, individual one of pieces, to mass produced, identical, high tolerance components.

Modern CNC systems are highly automated in terms of their mechanical design and their software programming. Developing the mechanical design of CNC machines is usually undertaken using Computer Aided Design (CAD) software, followed by Computer Aided Manufacturing (CAM) software. The movements of the resulting CNC machine are then reconstructed into commands that are needed by the machine to produce the target output object. In traditional CNC manufacture the work piece is fixed and immovable.

In this project, the CNC machine also had to interface with the navigation

system as the object being cut could move and this movement was recorded using the navigation system. This highly complex CNC machine was built and then automated using special customized software programs and some of these programs could communicate with the navigation system and other programs could communicate with the CNC machine over a network. Some of these programs were implemented on the controlling computer and some on a micro-controller in the CNC machine itself. CNC machine movements needed to be controlled to move along at least two axes, supplemented by a tool spindle giving the depth of the movement. Very accurate sub-millimetric movements of the machine were needed, so, a direct-drive stepper motors (servo motors) were needed.

Nowadays, CNC is preferred over manual machining due to various important advantages. First, its operation is highly precise and repeatable which enables it to produce complex shapes that would be impossible to produce using manual machining. Second, CNC machines are fully automated therefore they can work without human interference during their entire machining cycle, eliminating the mistakes that could be caused by human error, so producing a consistent and predictable output object. Third, attempting to produce a different output object ,for example removing the bone for a different sized implant, only requires loading a different software program which makes using CNC machines highly flexible with easy set-up and running steps (Lynch, 2017).

5.2 Aims and Objectives

The objective of this section of the thesis was to build a fully functioning CNC machine with three degrees of freedom. The machine was designed to perform a knee joint burring procedure while also driving by the navigation system. The accuracy of the machine had to be very high to make best use of the working navigation system that was discussed in Chapter 4. Another Objective was to create a driver to control the machine remotely which supports autonomous control for the cutting phase. The development of a mobile and "reactive" CNC machine capable of burring bone during a knee Arthroplasty on a potentially

moving leg was recognized as ambitious but also as providing a step change in orthopaedic-robotics if it could be achieved.

5.3 Methods

The main target was to design a very accurate machine.

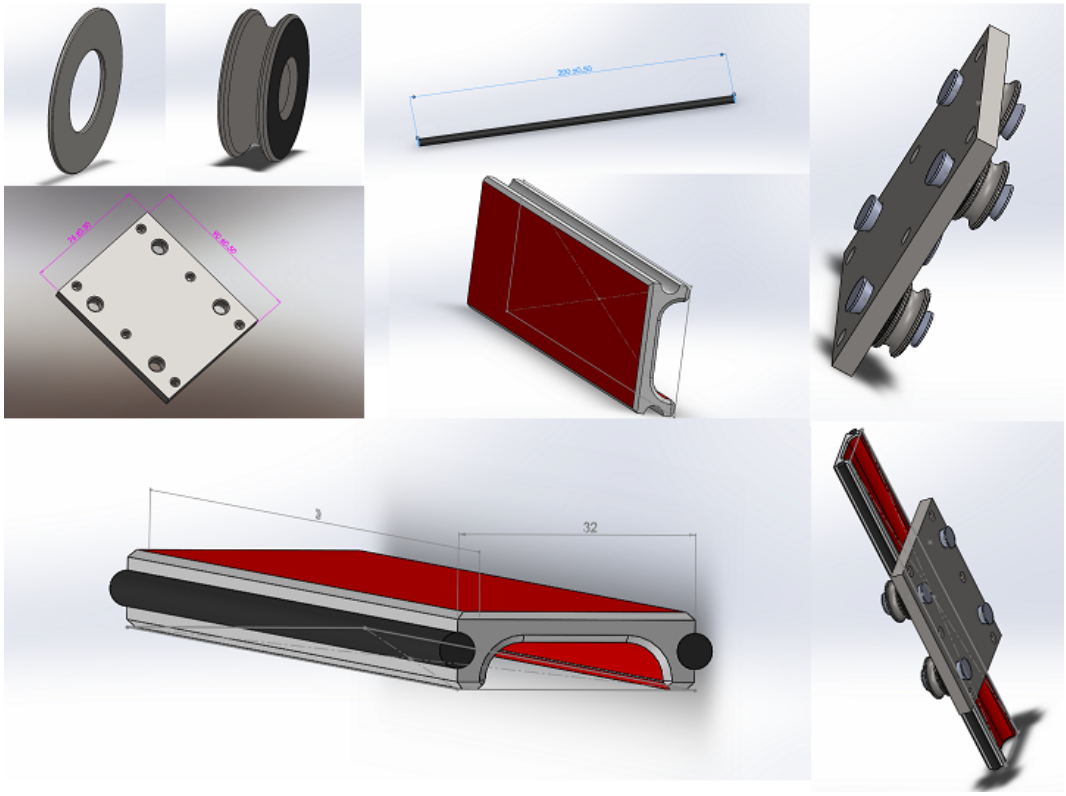


Figure 5.1: CNC Parts Through Assembly

As shown in the previous chapter, the OptiTrack System accuracy was 0.3mm. This value was set to be the machine's required accuracy. In order to reach such accuracy in a CNC machine, the first thing needing was to install highly-accurate motors with highly stable linear motion carriages which held and moved the cutting burr or other machine tools with this accuracy.

5.3.1 Designing The CNC Machine

In the simulated prototype, a rail system was used to carry the carriage which provided a high accurate and stable linear motion. The design was made in Solidworks 2015 software, it was also made from scratch by creating the pivots, washers, rods and carriage as shown in Figure 5.1. By assembling the parts together a full rail carriage system was created and by combining five of them, a fully functioning CNC machine prototype was achieved. Two motors controlled forward backward motion together. A further two motors controlled up and down and then one motor controlled left and right. See Figure 5.2 below.

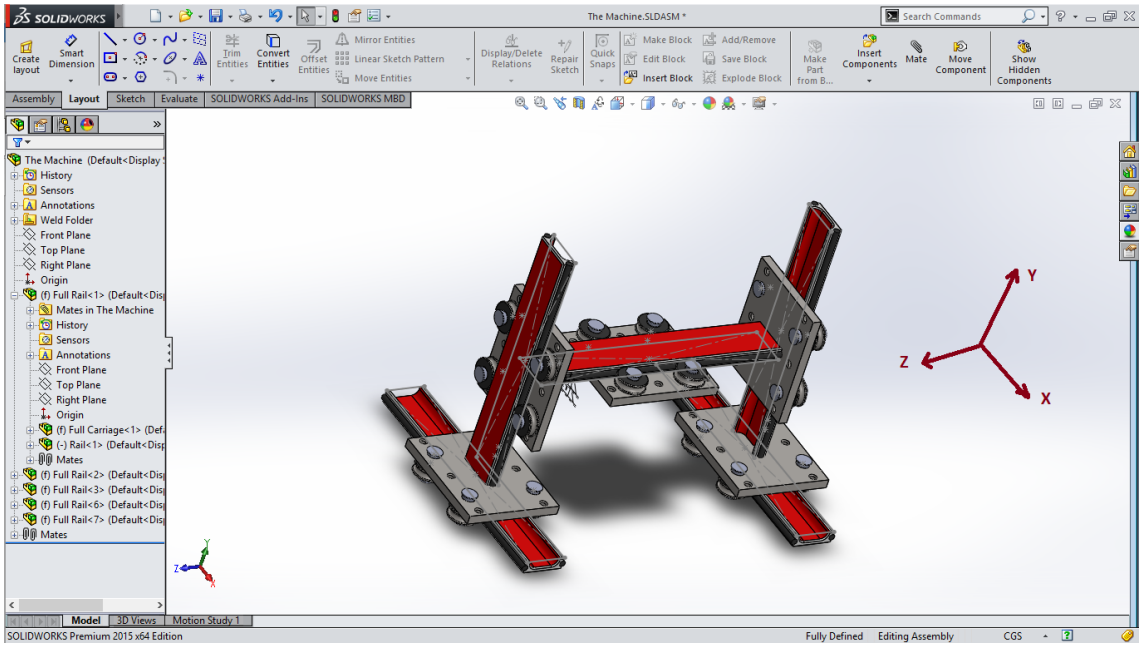


Figure 5.2: CNC Prototype

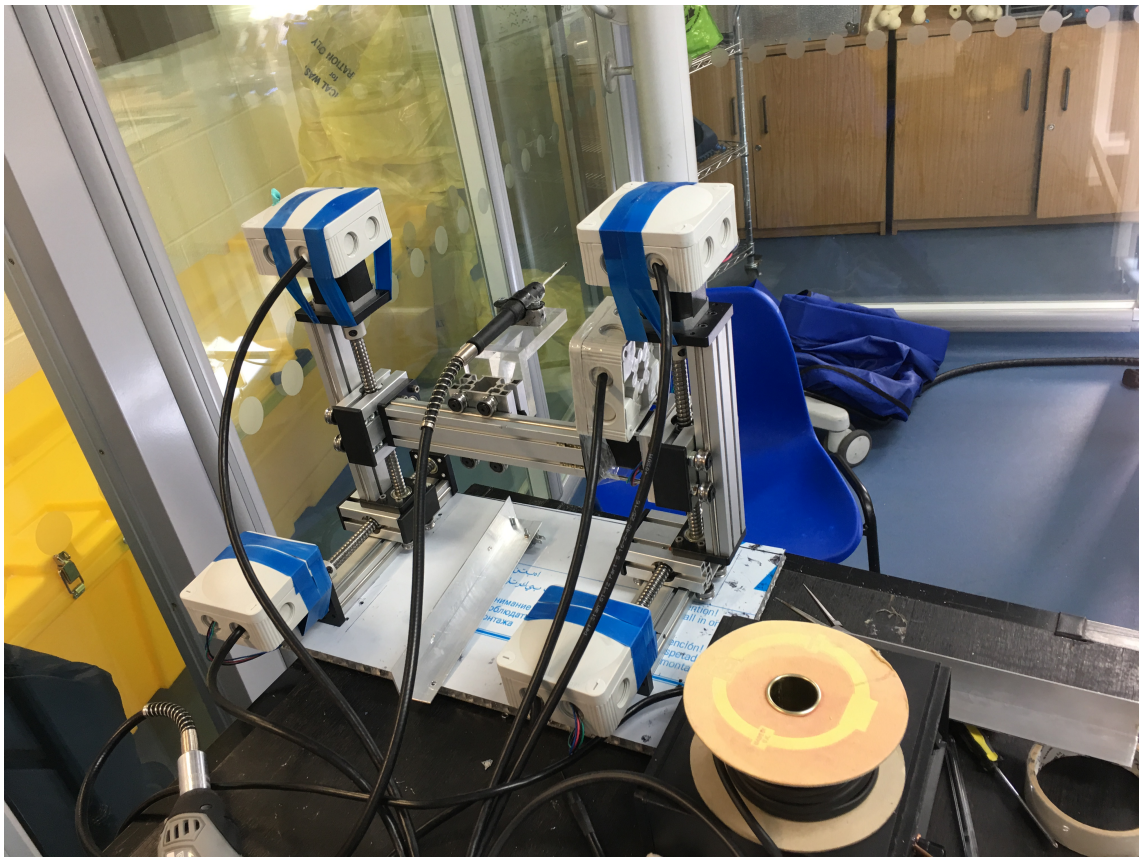


Figure 5.3: CNC Final Design

As a working stepper motor, Nema 23 motors (USA, 2017) were the best fit as they provided 200 steps/revolution and a 64 $N.cm$ holding torque. As for the production process, some of the parts needed to be made and others were purchased. Five rails for linear motion with five Nema motors were purchased and when assembled together they formed the final CNC machine working design shown in Figure 5.3 above. Each carriage had a $100mm$ length of movable distance.

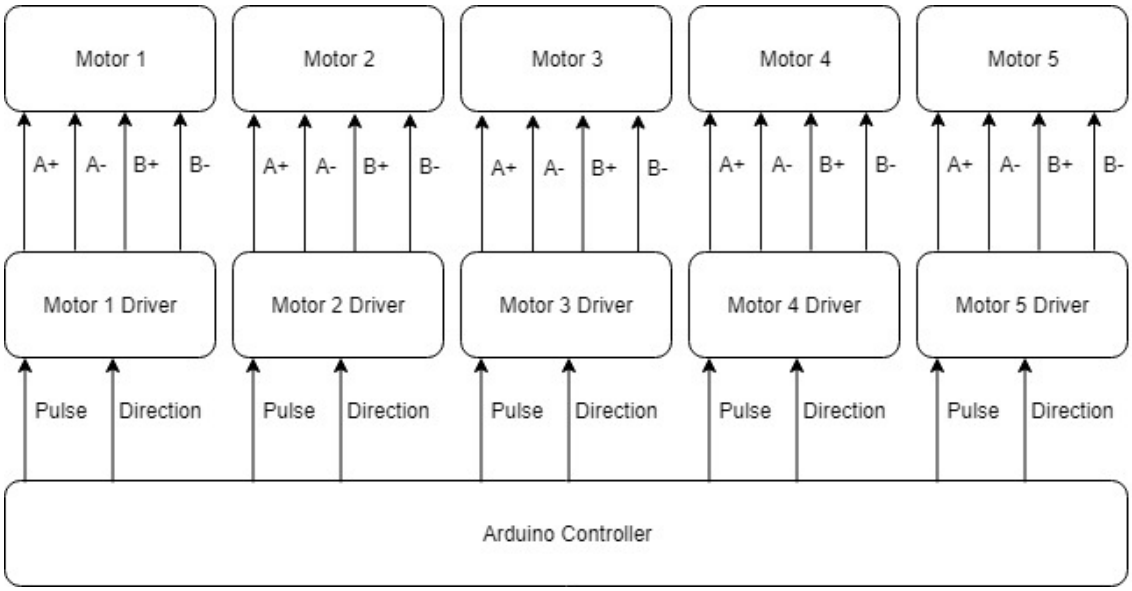


Figure 5.4: CNC Controller Schematic

A micro-step driver TB6600 was used to separate the motor from the controller circuit as each motor took a 3A input current (USA, 2017). Each Driver had four inputs (VCC, Ground, pulse and direction) which it received from a central Controller and output four phases (A+, A-, B+, B-) directly to the motors, see Figure 5.4 above for more details.

The middle carriage - which moved horizontally - was defined as the machine's local Z-axis. This carriage held the cutting burr, see Figure 5.5. The burr was fixed inside a flexible tube which was connected to a Maplin mini grinder 170W variable speed rotary tool. The Z-axis rail was fixed at both ends to another pair of carriages. These two carriages moved vertically in order to act as the machine's local Y-axis. Each of the Y-axis rails was fixed from the bottom to another carriage. These two carriages moved on rails in a horizontal direction (fore and aft) to form the machine's local X-axis, this x-axis was parallel to the cutting burr.

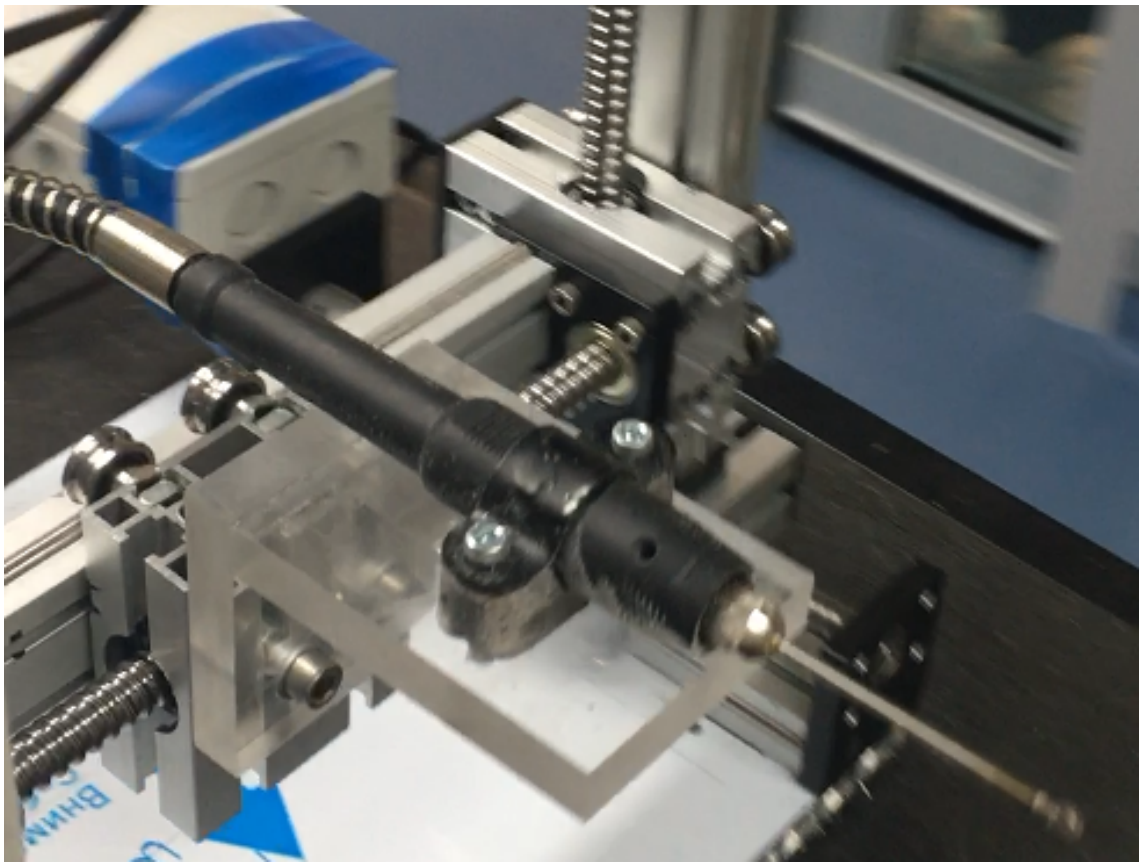


Figure 5.5: Cutting Burr Fixed inside flexible tube and on also carriage

At this point the mechanical design was complete and attention switched to the Controller. Arduino Duo provided all the required local control for the CNC machine and an Arduino Wi-Fi Shield was also used to allow wireless communication from the computer to the Arduino Duo. The Arduino controlled the machine movements through the motor drivers as discussed previously. The use of the Wi-Fi connection eliminated control wires from the surgical computer to the CNC machine.

5.3.2 Hard-wired Circuit

A small basic Control circuit was used to regulate voltage and ampage for the Arduino board,along with two joystick switches, an LCD display and the motor drivers, see Figure 5.6. The joysticks allowed local manual control of the CNC machine prior to implementing of the automatic cutting path and the LED dis-

play allowed process monitoring. The circuit schematic shows the implemented supply circuit for the Arduino board, LCD display, the two Joysticks and all five motor drivers. The circuit was responsible for down regulating the voltage as the power supply output used for the motors was 13.8 VDC, but the Arduino board and all other parts wired in this circuit took 5V as an input, so the circuit took the 13.8V and delivered a 5V output.

The regulator circuit also reduced the noise in the supply to the controller board and other components as the 13.8V supply was noisy due to the working motors, and drivers. The regulator used was the LM7805. Two capacitors were used to reduce the noise on the input voltage. A 10 µF capacitor was connected between the input and the ground, another 1 µF was connected from the regulator output to the ground. Both capacitors negative side was on the ground, see Figure 5.6. The circuit was first created and tested on a bread board and then implemented on a Veroboard, see Figure 5.7.

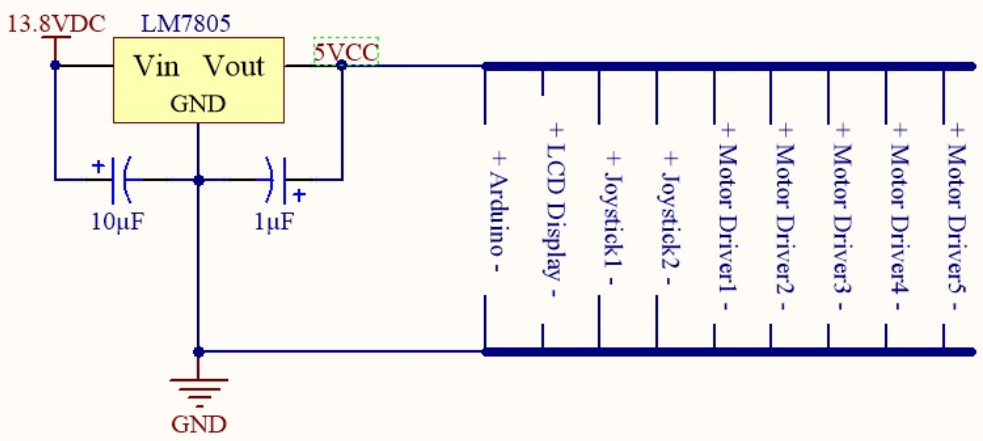


Figure 5.6: Circuit Schematic Diagram

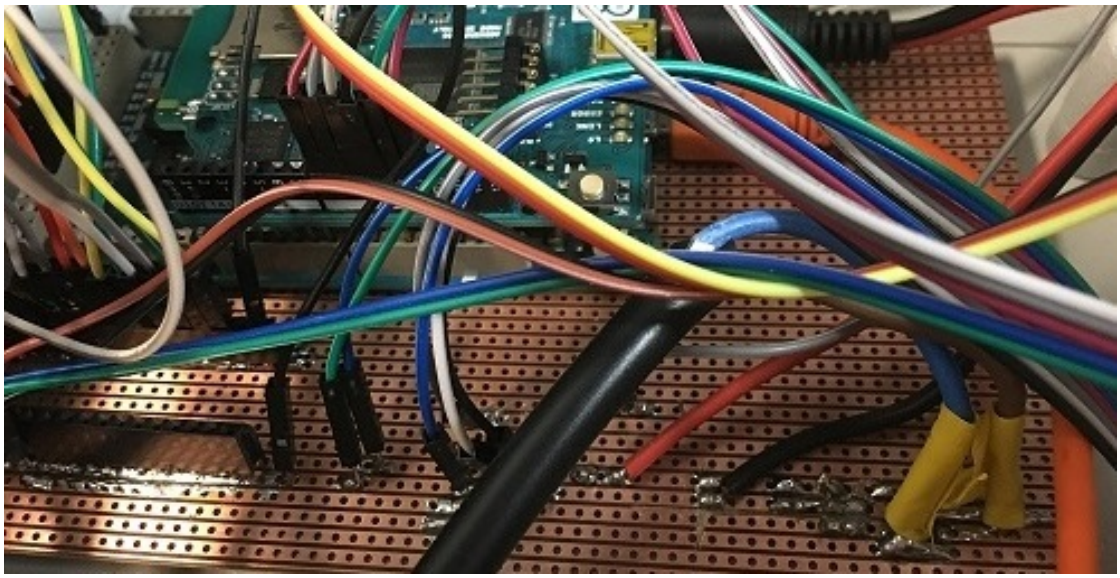


Figure 5.7: Circuit Implemented on Veroboard

5.3.3 Joysticks and LCD Display

The joystick module provided two axes manual control of the CNC machine. When the stick was pressed it also worked as a push button giving two switch controls, see Figure 5.8. The joysticks were used to manually control the CNC machine. As each joystick could control two set axis, one was used to control the machine local X and Y axes, and the other was used to control the machine local Z axis.

The joysticks switches provided selection of the Arduino control modes 1) manual 2) network control. This was achieved by pressing the corresponding switch. The joysticks were placed and fixed to the left and right of the LCD display see Figure 5.8. The joysticks outputs went straight to the Arduino board for location control or selection and then the Arduino board output the location update or selection result on the LCD display for observation.

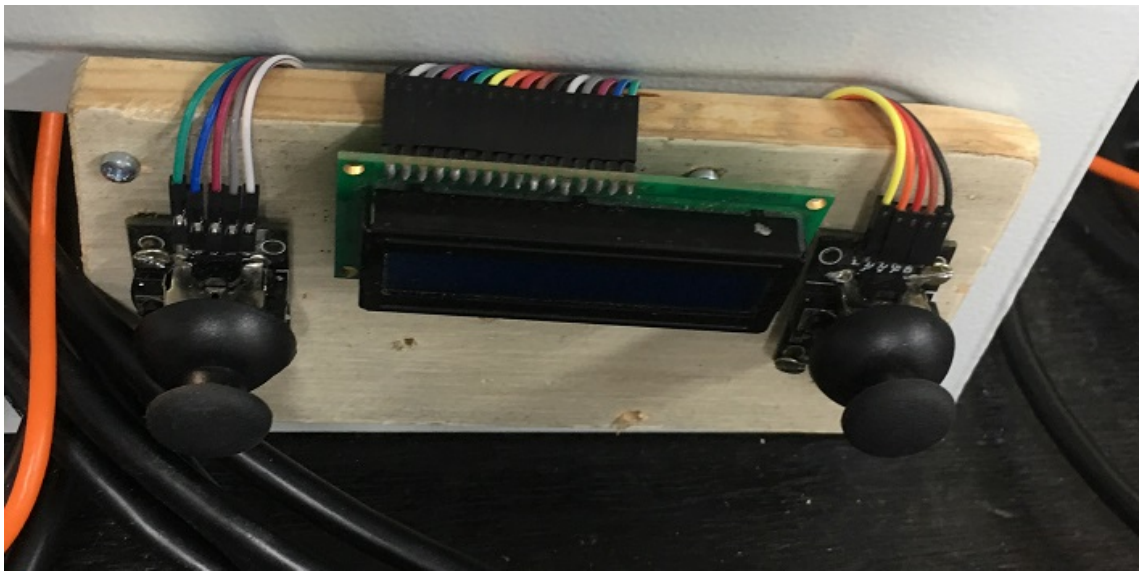


Figure 5.8: Joysticks and LCD Display

Each joystick module board had five pins: the VCC and Ground pins were wired to the Veroboard for power and the other three pins were for the communication between the module and the Arduino board. These three pins were labelled on the module as VRx, VRy and SW. The VRx pin out gave an analog value to the Arduino board, this value was 755 if the stick was in neutral, if the stick was moved towards the positive direction of the X-axis this value increased, and if moved towards the other direction this value decreased. the same concept applies for the VRy pin. The SW pin output was zero when the stick was unpressed but when the stick was pressed the output value switched to 1. To view the control code for the Arduino please refer to Section 7.3.2

The LCD display was a 16x2 display, see Figure 5.8. Some wiring and components were applied on the Veroboard to make the correct setting for the LCD display. LCD pin(0) was wired to ground, pin(1) to VCC(regulator output), pin(3) was wired to a 330 Ω resistance then to Ground, pin(5) was wired to the Ground, pin(15) was wired to 330 Ω resistance then to VCC to enable the +LED, pin(16) was wired to 330 Ω resistance then to the Ground to enable the -LED and pins(4,6,11,12,13,14) were wired to the Arduino board for receiving the output.

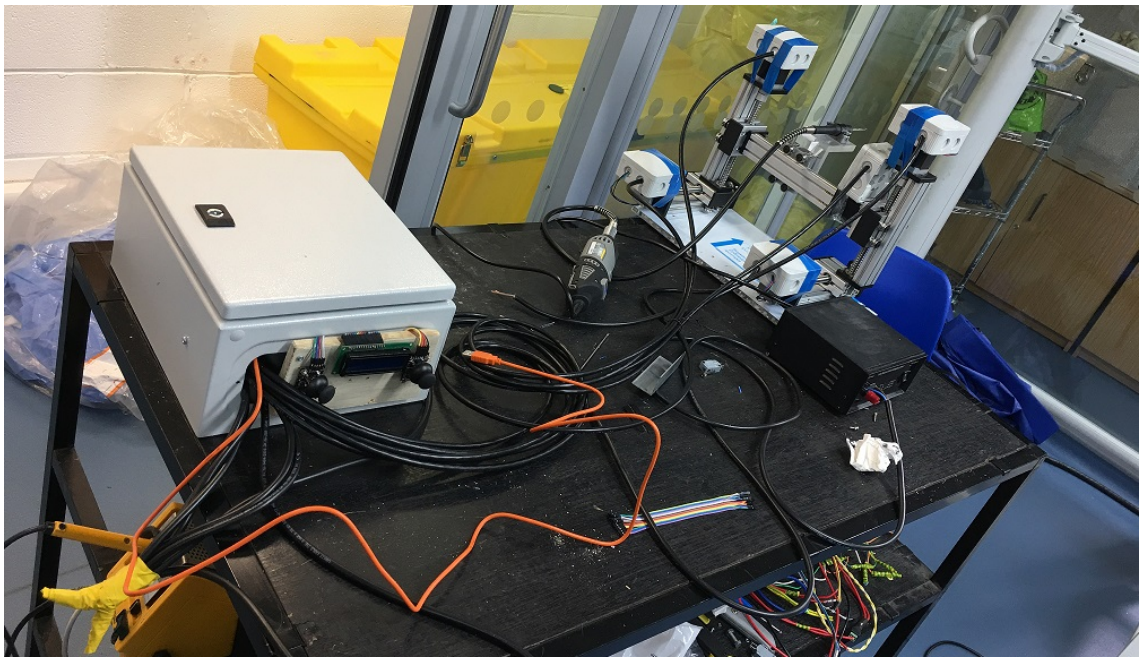


Figure 5.9: CNC Final Design and Controller

A 15A power supply unit was used to supply all of the operated hardware. (Figure 5.9). A safety kill switch was wired on the power supply cord. In case of emergencies the kill switch was pressed and it shut off the power supply and therefore all hardware including the machine movements and the Maplin Rotary Tool would be disabled. There was also a reset press button to reset the machine carriage position to the local origin (0,0,0).

The five motor drivers, Arduino control board, Wi-Fi Shield, and the regulation circuit were all fixed inside an electric box to isolate the parts and provide portability. The joysticks, LCD display and reset button were fixed on the outside of the box to provide access.(Figure 5.9) .

The cost of the CNC unit and the controller box main components as (five motors, five motor drivers, five rails, Arduino and Arduino Wi-Fi shield, small control board, 15A power supply) were: The cost of one rail (one motor and one motor driver) costed $135.5 multiply it by five hence the CNC costed $677.5, the Arduino Duo costed $35.5, the Wi-Fi shield costed $81 and the mercury power

supply costed $48, most of these components were bought from Amazon and RS online store.

5.4 Experimental Validation of response

Two experiments were conducted to measure the machine accuracy. The first experiment was performed to calibrate the system using the navigation system. A marker was fixed on the moving carriage at the tip of the burr and the location was acquired by the navigation system. The Arduino was programmed to move 15000 steps. After the program execution the location of the marker was acquired again. The Euclidean distance between the two locations was calculated and the distance was $75mm$. So if the motors took 15000 steps to move 75mm then $15000 \div 75 = 200$, as the motors move 200 step/revolution then each carriage moves 200 step/mm or $5\,\mu m/step$. This was as expected.

The second experiment was made to verify the previous result by reversing the process. Each carriage had a 100mm length to move on its rail. The Arduino was programmed to move 20000 steps and the carriage was set on the rail starting edge. After the execution, the carriage reached the end of the rail by moving exactly the 100mm.

5.5 Discussion

A precise and accurate CNC machine was needed for this project. The machine was built with three degrees of freedom in order to be able to perform knee joint surface burring. Highly accurate motors were used to move with at least the same accuracy as the navigation system.

The Machine's accuracy after testing was 200 steps/mm for each axis which means that each carriage can move $0.005mm/step$ or $5\,\mu m/step$. A surgery with that accuracy should have good bone cutting results. However, it remains to be

seen if this potential accuracy can be replicated in practice. Considerable further work will be required before such technology could be shown to be ergonomically suitable for the operating theatre.

5.6 Conclusion

CNC machines have become widely used in the world of robotics. This type of machine has provided unique accuracy in many procedures and industries. The pilot CNC machine developed in this project provided accurate burring suitable for the partial knee resurfacing operation.

Chapter 6

Methods: D-Flow Applications

6.1 Introduction

The following chapter presents, a visual programming tool needed to provide operator control of the cutting process and to link the navigation and CNC systems. D-flow software by Motek Medical was used to allow the definition of the operation using visual programming. D-flow was developed as a programming tool which considers the subject as an intrinsic part of the real-time feedback loop and in which the behaviour of the subject is measured using multi-sensory input devices, then motor-sensory, visual, and auditory feedback is returned to the system via output devices. For example, input devices can be motion capture systems and force plates, output devices can be motion platforms, treadmills, audio devices and displays. The feedback strategies are flexible and can be defined by the operator. D-flow has some particular characteristics, it consists of a top layer responsible for the communication between hardware components, a multi-display rendering system, and a modular application development framework based on visual programming. D-Flow combines these components into one system that focuses on rehabilitation techniques.

Although developed for visual feedback of motion capture data during rehabilitation, D-flow has all the elements necessary to control the CNC cutting system while monitoring the position of the limb using the navigation system.

D-flow depends on the creation of modules, each module has a specific func-

tionality. Modules can be incorporated together in order to create complex applications with interactive virtual reality. D-flow offers different types of modules. For example, some modules are responsible for directly controlling hardware devices, other modules are responsible for reading real-time data streams through input devices while other modules manipulate virtual objects to enable the interaction between the subject and the virtual environment by controlling the playback in the virtual environment or disclose collisions between objects. Lastly, low-level modules are necessary to act as the building blocks for high-level function. Moreover, D-flow offers a general-purpose scripting LUA module and an expression scripting module.

In order to provide communication between modules, each module contains a set of input and output channels. Output channels from one module can be connected to input channels from other modules, enabling data to navigate from one module to another one(Geijtenbeek et al., 2011)(*D-Flow - Motekforce Link*, 2018).

6.2 Aims and Objectives

This chapter covers the D-Flow main application for creating the cutting procedure transition file. The methods section includes the application flow, the created algorithms and mathematical formulas for the main application.

The objective was to cover all the used D-Flow modules, provide a detailed illustration of the clusters tracking technique along with flow charts and code portions of the used algorithm, and also present a complete guide for the creation of the registration process using pointer probes with the required codes and mathematical representation.

This section also demonstrates the details regarding implants shape acquisition and path planning for the machine that enables it to move and perform the cutting.

6.3 Methods

With the hardware developed, an overall control program was needed. This chapter is in two parts: detailed illustration of the main application followed by the process of creating a three-dimensional scanner application in D-Flow needed to capture the bone surface shape.

6.3.1 Main Application

For this application, a full task by task flow had to be provided for the surgery. Figure 6.1 below shows the flow of the written programs to perform the surgery. In order to accomplish the whole chart, three programming languages were used: LUA, Java and C/C++(Arduino code). First, Motive software was used to transfer the markers' positions to the D-Flow main application. Second, the markers' positions were received by the D-Flow main application and used to generate the cutting file; which is the focus of this chapter. Thirdly, the cutting file was sent over the network by the Java application (sekseka), which will be discussed in the next chapter. Finally the cutting file was received at the CNC machine and the burr perform the surgery, also detailed in the next chapter.

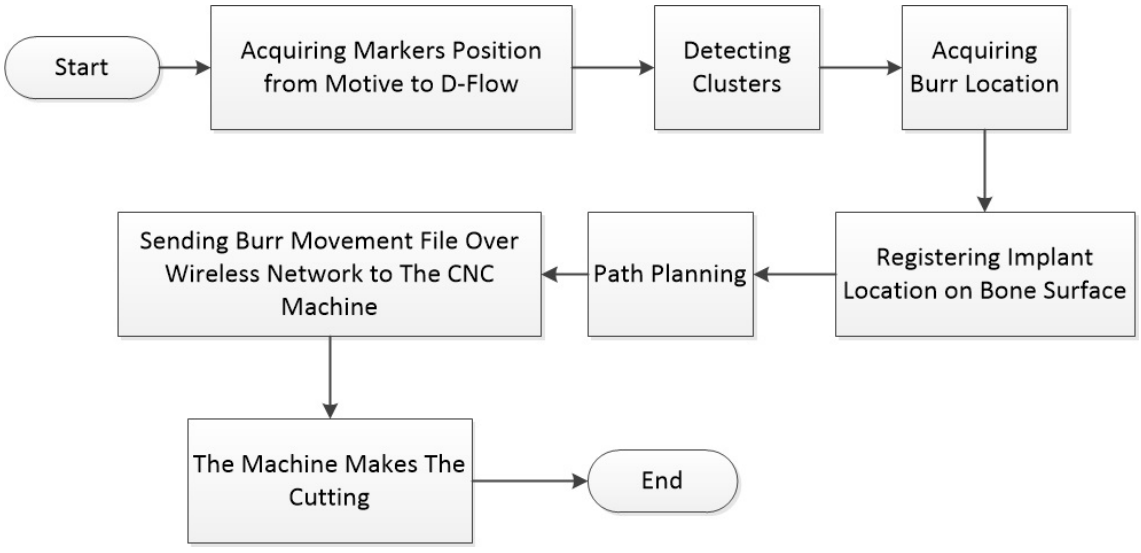


Figure 6.1: Application Diagram

The main application task was to take markers input from all used clusters for

tracking objects in the surgery, the application also uses some of these clusters for the registration process which was mandatory for the surgery. The last task was to create the cutting path for the burr based on the acquired data and store the result in a file (the cutting file).

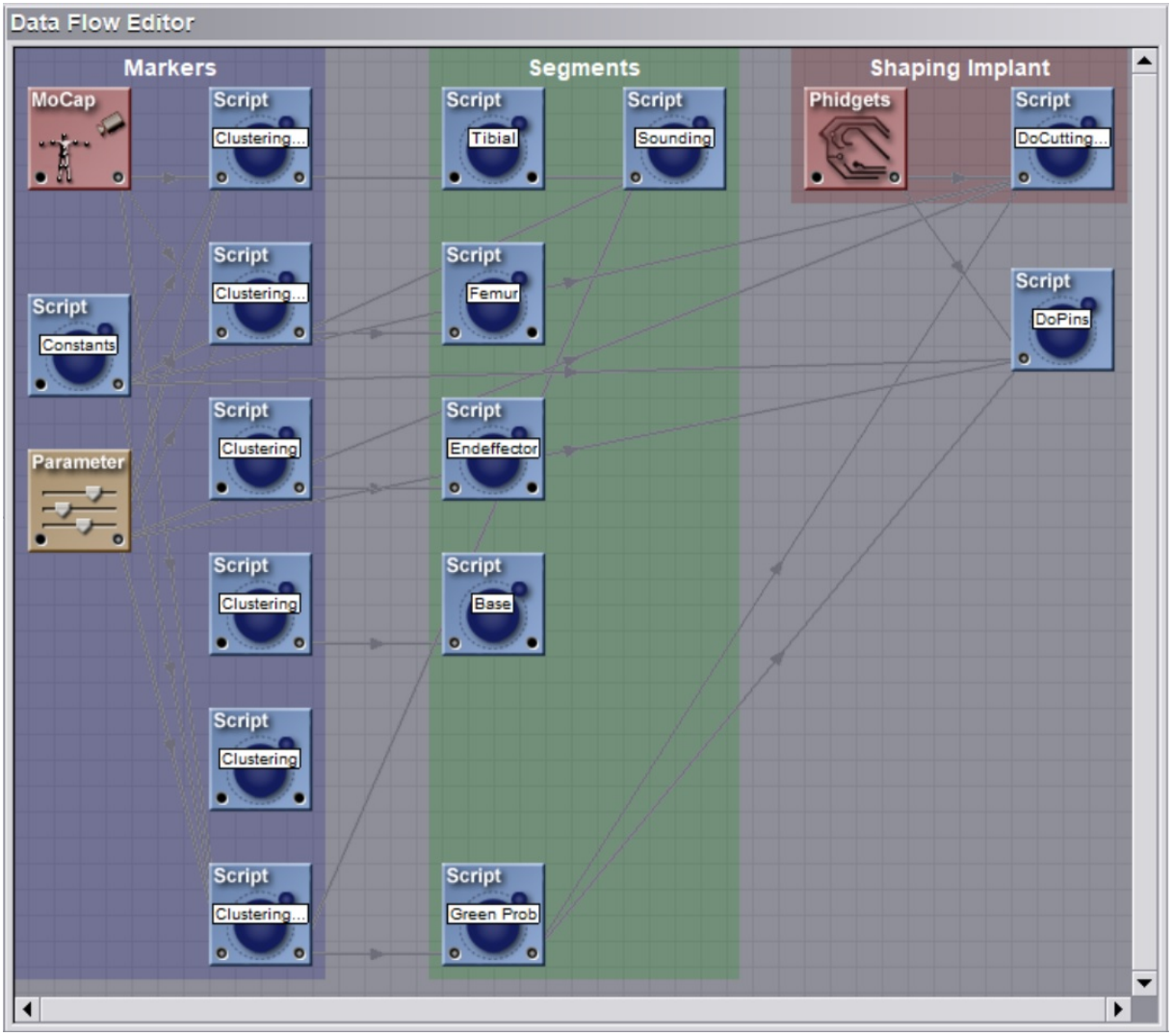


Figure 6.2: D-Flow Data Flow Editor View

D-Flow software has many modules, some of these are task specific modules like the MoCap and Phidgets modules, others are general purpose modules like the Script module which allows the developer to write scripts in LUA programming language, see Figure 6.2 for the previous modules icons. The MoCap module's function was to manage communication between various motion capture software programs and D-Flow software programs. For this application, the MoCap mod-

ule's task was to take markers' positions from the Motive software and pass it forward to other D-Flow modules. Further modules and details will be discussed through this application.

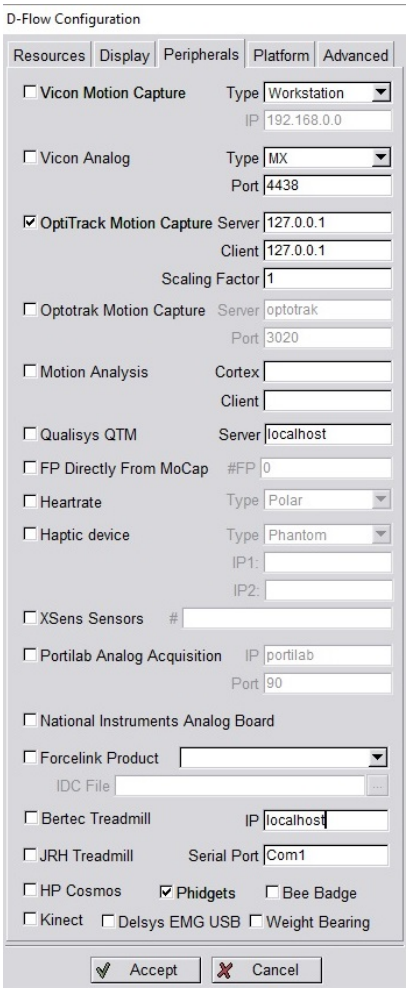


Figure 6.3: Configuration Window

The function of the main application could be divided into a flow of procedures as shown in the application diagram in Figure 6.1. In the beginning, the marker positions were received by D-Flow using the MoCap module. In order to be able to receive the markers from Motive over the network of the computer some D-flow settings needed to be adjusted. As shown in Figure 6.3, at the Peripherals tab in the D-Flow Configuration window, the OptiTrack Motion Capture check box was checked to enable the communication. Since Motive software and D-Flow software were on the same computer, the IP addresses of OptiTrack server and

client were set to "127.0.0.1", but if Motive had been on a different computer then the server IP would've changed. Also, the scaling factor was set to one; if the value changed, it would change the distance between the received markers. At the bottom of the window the phidgets check box was also checked to enable communication with Phidgets hardware which will be discussed later.

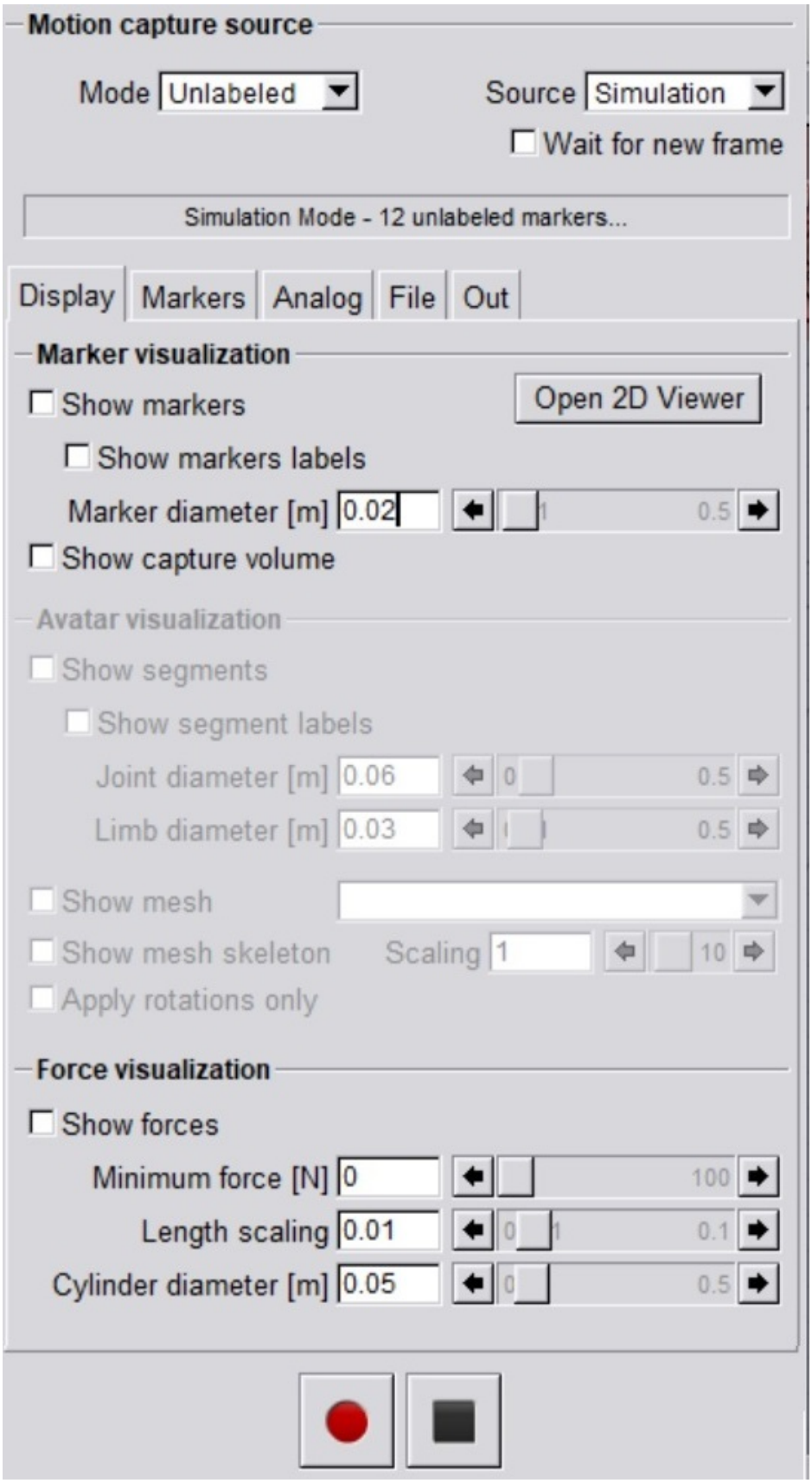


Figure 6.4: MoCap Module - Display Tab

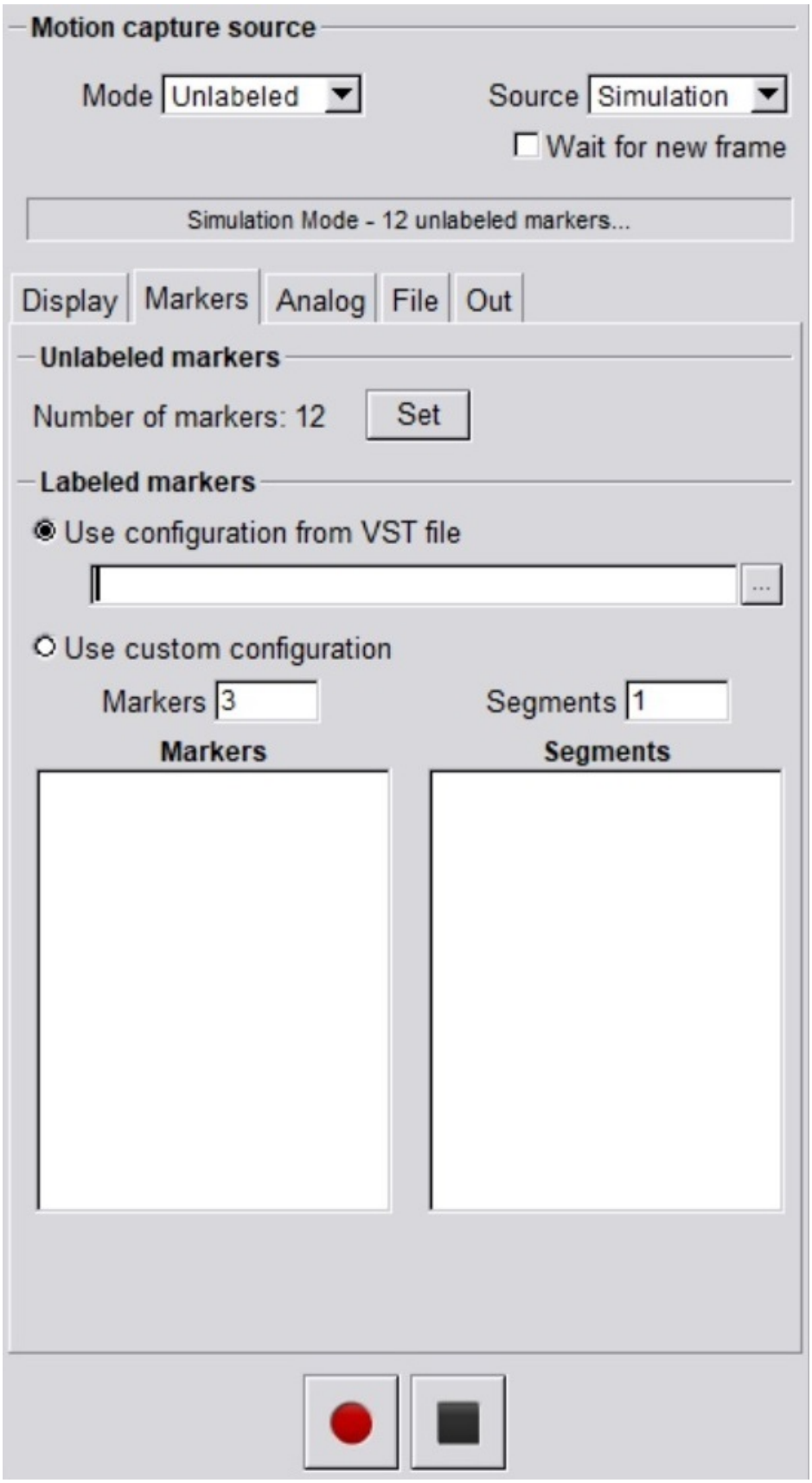


Figure 6.5: MoCap Module - Markers Tab

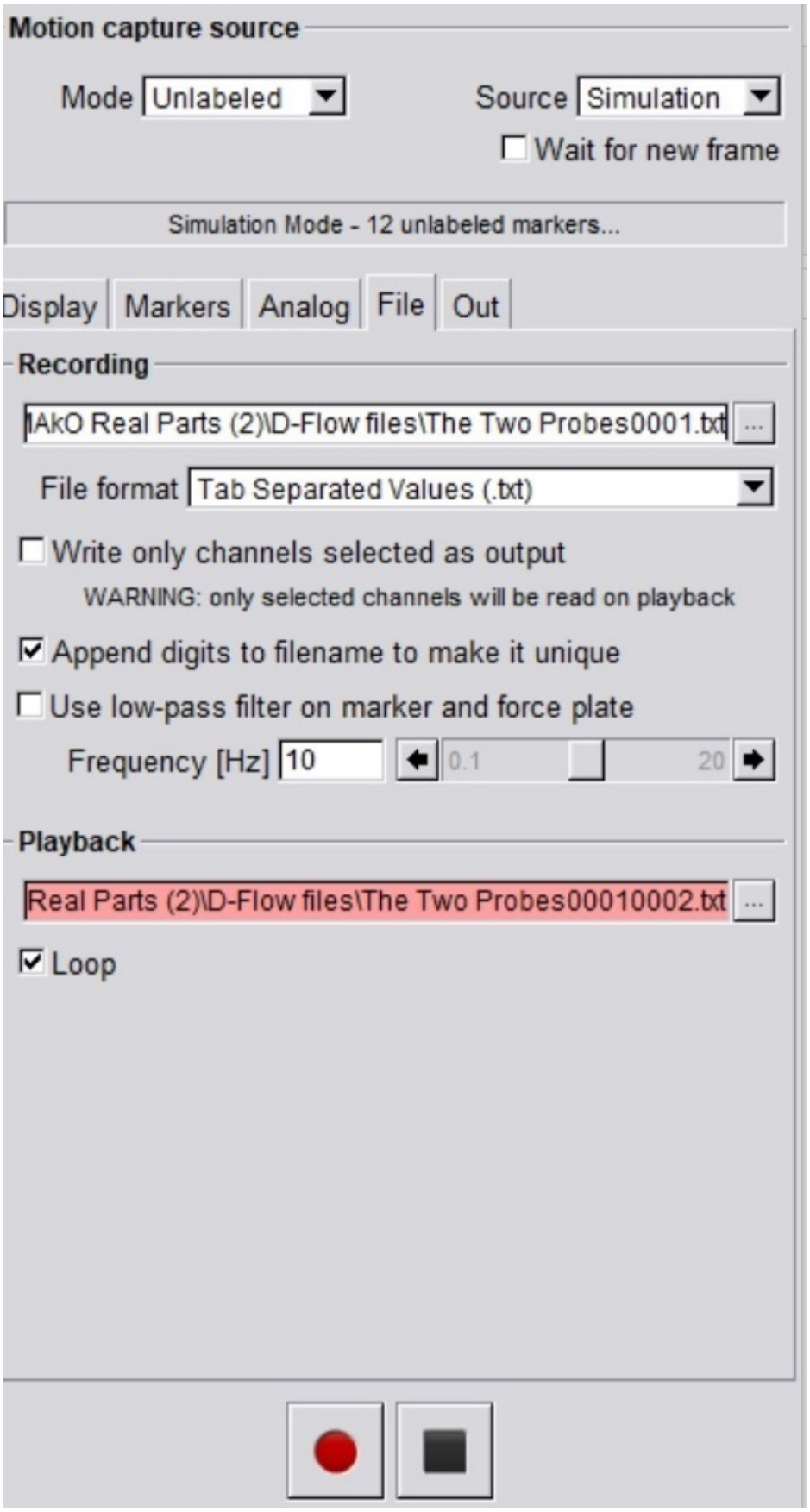


Figure 6.6: MoCap Module - File Tab

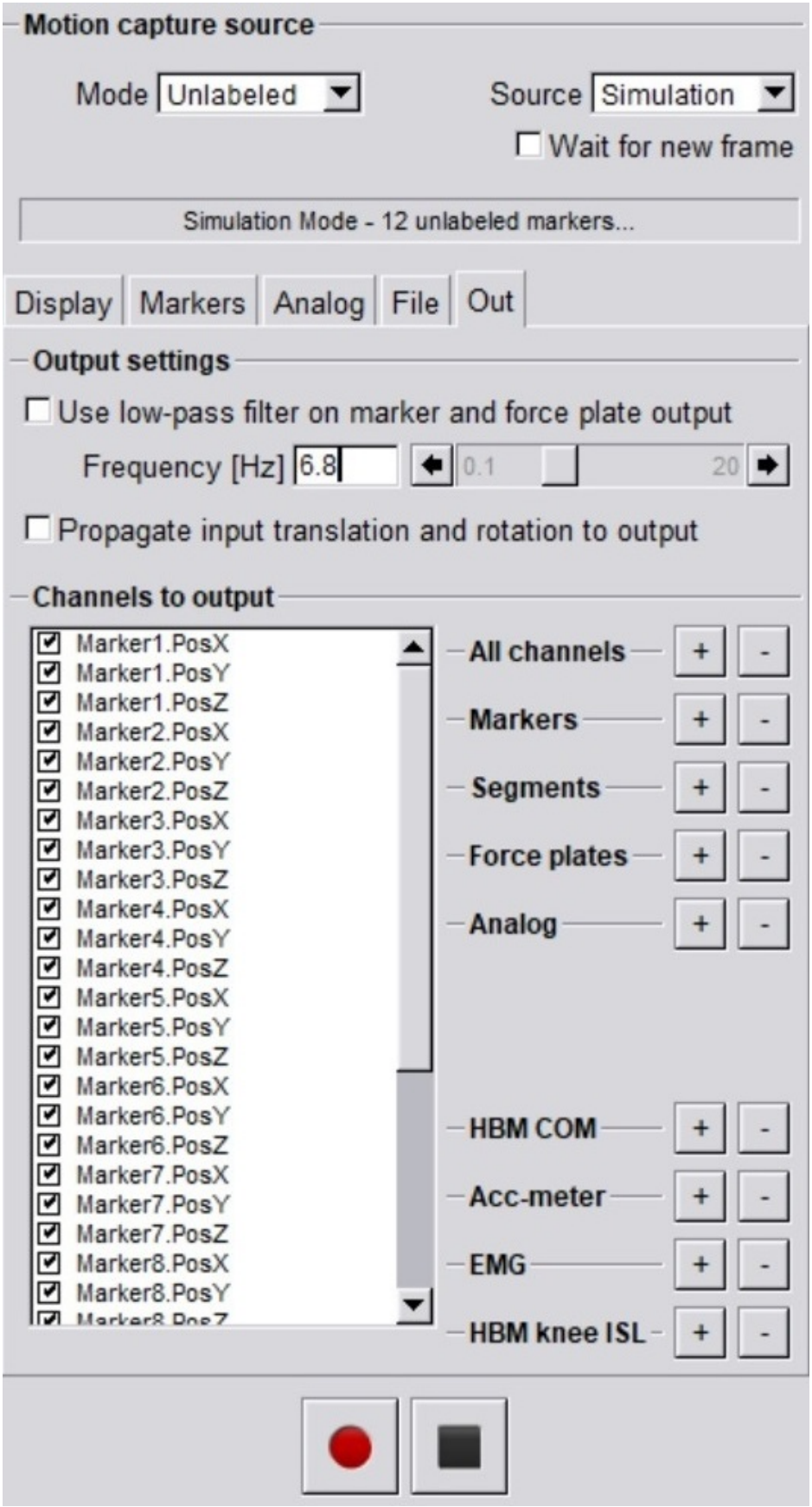


Figure 6.7: MoCap Module - Out Tab

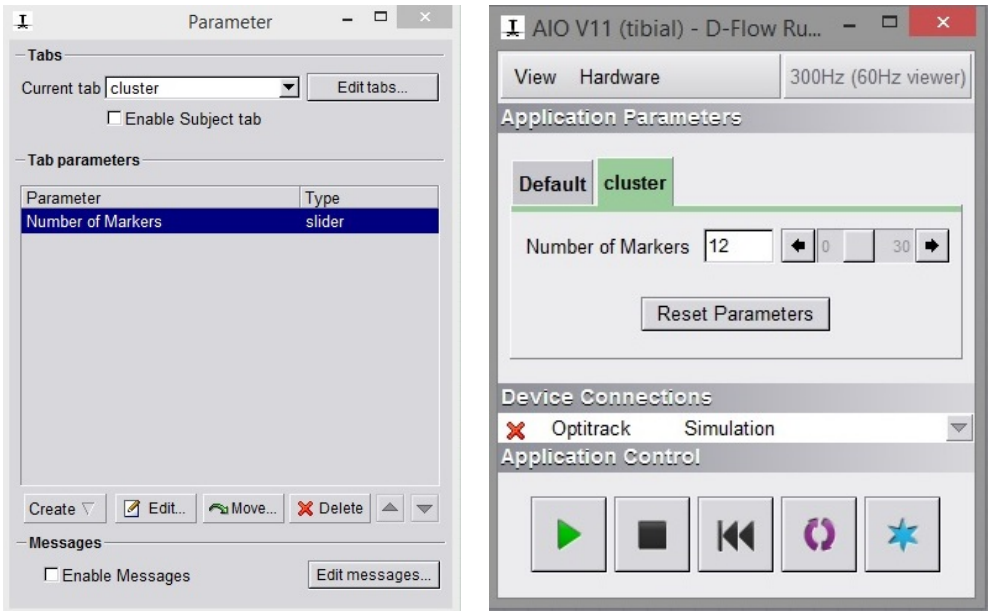
After setting the D-Flow configuration, the MoCap module was configured. Figures 6.4, 6.5, 6.6, 6.7 shows the settings of the MoCap module. Each picture shows a different tab settings that was needed. The first thing that was set was the source menu button, the default option for the source is "simulation", which generated random floating data for the simulated markers in the DRS visualization window in D-Flow. But, in order to get the working markers from Motive, the source had to be changed to "live". If Motive is disconnected, then D-Flow automatically switches the source back to "simulation", in this circumstance the MoCap module would need to be put back to live once Motive was started. The second step was to check the Show markers check box in the Display tab in order to be able to display the motive markers on the DRS window. The show markers label is a very good option for early tracing of the markers especially if the developer is not yet familiar with the working cluster shapes. One last configuration was made in the Display tab and that was the Markers Diameter, the default value for this setting was 0.04 m, this value represents the diameter of the drawn markers in the DRS window, in the presented testing through many D-Flow applications this value was sometimes considered too big and sometimes too small. Generally speaking, when the working cluster sets were far away from each other it became impossible to zoom in the DRS window and see the tiny moving markers. When visualisation was performed on a single cluster or adjacent cluster sets, increasing the markers size too much caused marker overlap. See Figure 6.4 for the previous details on the Display tab.

Moving on to the Markers tab, see Figure 6.5, the number of markers was set to 22 markers as this was the maximum number of markers that could be used in this application (4 markers on the Endeffector cluster, 4 on the Base cluster, 4 on the Femoral cluster, 4 on the Tibial cluster, 3 on the Blunt Probe cluster and 3 on the Sharpe Probe cluster). When the number of markers set in D-Flow was greater than the number of received markers from Motive, the extra markers were placed at (0,0,0) as (X, Y, Z) (the origin) location in D-Flow. If the origin was on the track of a moving cluster or close enough, these missing markers might have caused that cluster to be unrecognised. Therefore, these missing markers

could confuse the algorithm. A marker detection algorithm which dealt with this problem was required and will be discussed in detail in the cluster detection module section.

The File tab was used to record marker data for a period of time, see Figure 6.6. This could be achieved by creating a file for saving the data using the module. When then the red button at the bottom of the module was pressed to start recording. Afterwards, when the required data had been recorded, the stop button was pressed. The recorded data can be later used as the input for the MoCap module instead of the live system by selecting the source to be a file instead of live. Checking the "loop" check box causes repeating of the data if required. This feature was very helpful while updating the code as data could be played back rather than needing to be recaptured live.

The last tab to configure was the output tab, see Figure 6.7. This tab was responsible for formatting and channelling the output from the MoCap module to other modules.



(a) Setting Window (b) Runtime Window

Figure 6.8: Parameter Module

A single script named "constants" provided some common variables that were required in all the developed scripts. The constants script provided data required

for various supporting operations to be discussed later. Another input module was used: the parameter module. The parameter module provided great support for the application to become more flexible, for example the number of used markers was variable during the development phase and this allowed the number of markers to be changed for example to 12 using a slider as shown in Figure 6.8b. By using this method, the application's load on the running computer was reduced greatly. For example using the parameter module to set the number of markers to the exact number prevented the software from processing missing markers. Because there were eight different scripts in this application that depended on these input markers, reducing the code from reading 22 markers -each have three (x,y,z)- to reading 12 markers (4 on the Femoral cluster, 4 on the Tibial cluster and 4 on Base cluster then replaced with a pointer which has 3 on top) reduced the reading instructions from $22x3x8 = 528$ to $12x3x8 = 288$. The reading instruction was the slowest instruction. Therefore, reducing it to almost half was a great improvement in runtime efficiency. As the number of used markers changed in most tests, the parameter module was used in all applications. In summary the parameter module allowed the operator/developer to allow a variable to be adjusted at the D-Flow runtime window during operation of the application, see figure 6.8b. This control parameter can be created as a slider, list, checkbox, button, separator, value display and GroupCheckbox. After selecting the input type and range of values the tab name and colour was set as shown in Figure 6.8b.

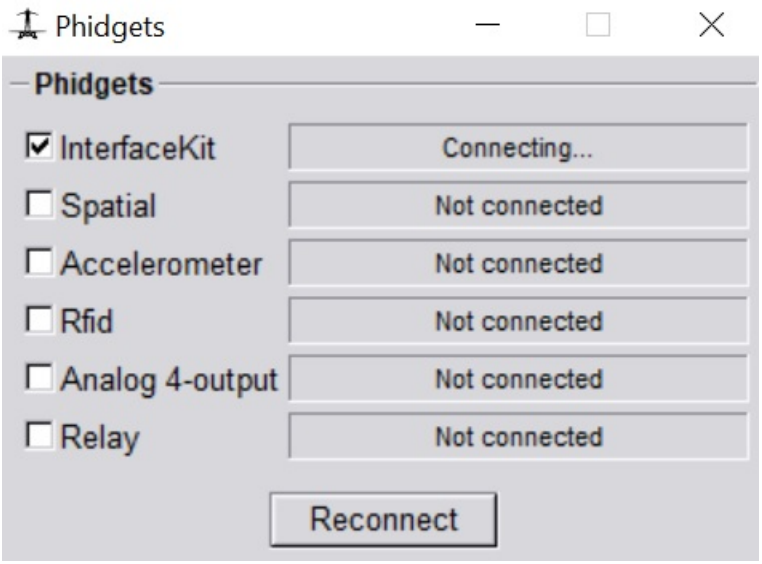


Figure 6.9: Phidgets module

The next and last module before scripting was the Phidgets module. The Phidgets module allowed the developer to use Phidgets hardware in order to communicate with D-Flow. This can be set-up easily after adjusting the hardware board and then connecting it to the operated computer via USB. The Phidget module was loaded, the board type was selected and then the connect button was pressed (Figure 6.9). After the connecting status turned green the module output the data from the Phidget device in this case a on-off foot-switch.

The last input module was the constants module mentioned before. It was a scripting module which was developed using the LUA programming language. This module had a very unique and specific task which was initializing all other scripts with the common variables values. For example all the used clusters needed a set of data so that their shape could be defined in runtime. In the constants module the shape dimension array was loaded from a file named after the cluster and saved on the hard drive of the computer. Rather than repeating this File load for each script, all data needed was loaded to a single script –the Constants script– and then passed to all six clusters scripts.

Moving on to the Clustering scripts in the Markers group (see Figure 6.2). The Clustering scripts were a series of scripts that were responsible for clusters detection and tracking. There could be a number of clusters that were used in the

surgical procedure. These were for the Femur, Tibia, Blunt probe, sharp probe, robot and burr tip. These clusters had one of two tasks. The first task was acting as a locater for a body part or surgical tool. This could be accomplished by placing the cluster on a visible surface of the part that needs to be tracked by the motion capture system. The second task was acting as a pointer (or a probe), the pointer was simply a smaller cluster on a tool with a pointy end that registers location of the tip to identify individual points, (Figure 6.10) .

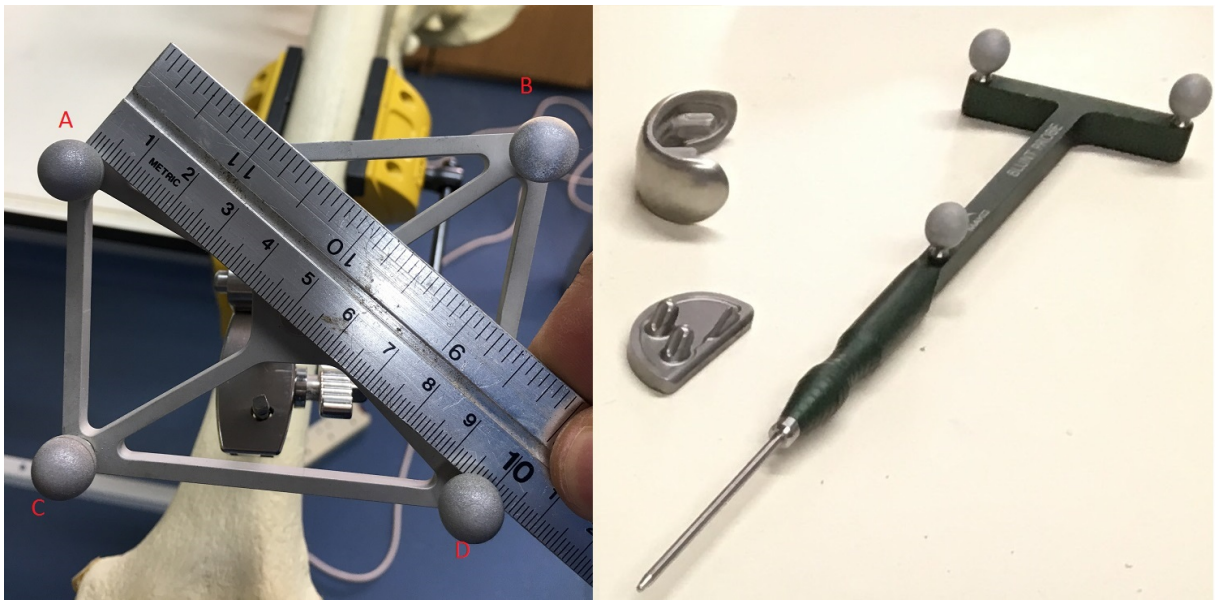


Figure 6.10: Mako's Tibia Cluster and Blunt pointer

All the clusters used in this application were Mako clusters. Two of these clusters, the Tibia cluster and the blunt probe are shown in figure 6.10. The Tibia cluster was used to locate the Tibia location when either it was moving or fixed. The blunt probe was used to register points on bone surfaces without damaging the surface. If this probe is used to locate points on the patient's knee joint there will be an extra couple of millimetres for the cartilage layer. Therefore, these is also Mako cluster set on another "sharp" pointer which can penetrate the cartilage layer and get an accurate bone surface point location.

Clusters were created to be distinctive, for example in the tibial cluster shown in Figure 6.10, the shape of the cluster is asymmetric with markers A,B,C and D in specific locations on the cluster. In the Mako cluster sets, the markers'

relative positions to each other were unique. Therefore, in D-Flow each cluster was recreated using the location and relative position of the markers, all six cluster scripts ran the same algorithm but on different cluster six-dimensional array datasets. Six ran in parallel to provide real-time detection of all clusters within one frame at 100 Hz.

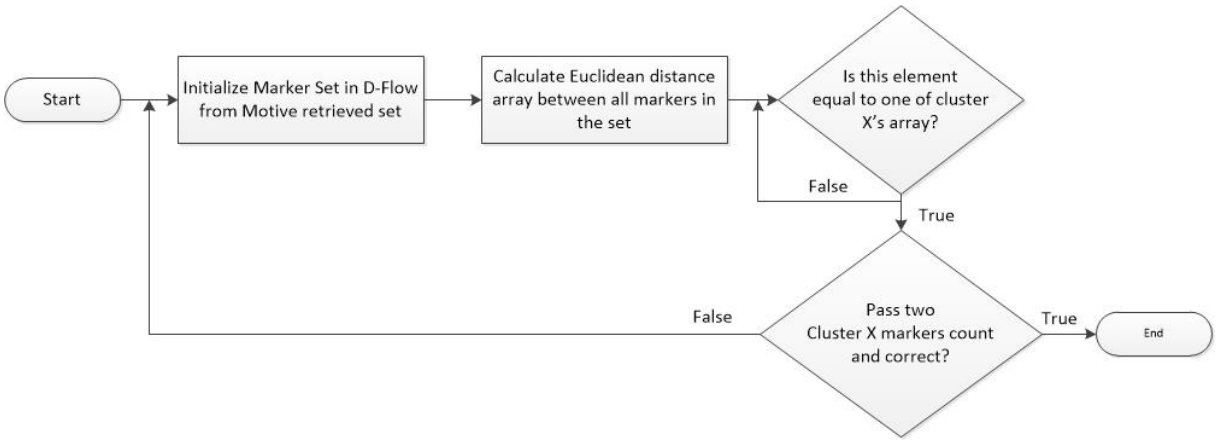


Figure 6.11: Cluster Detection Algorithm Flow Chart

The cluster identification process – in any of the clustering scripts – starts by calculating the Euclidean distances between markers, knowing that:

$$Euclideandistances(Y, W) = \sqrt{(Y_x - W_x)^2 + (Y_y - W_y)^2 + (Y_z - W_z)^2}$$

where Y and W (from the equation above) are both three dimensional points. This calculation was made for all pairs of markers sent by the Motive software, (Figure 6.11). Once this the Euclidean distance array was calculated, the module compared between this array and another array of the original cluster distances. For example the Tibia cluster – on the left side of Figure 6.10 – has four markers which meant that there were six different Euclidean distances between the markers. These distances were unique to each cluster for all Mako clusters in the set. For the Tibia cluster, the distances were: 51.8, 60.5, 67.9, 74.7, 97.2 and 127.4 mm, the 97.7 mm and 127.4 mm. These distances were measured previously by using the OptiTrack motion capture system and a special written D-Flow application to create a specific file for each cluster. Each cluster module used that cluster's pre-recorded data file to compare with the online data. If the distances

could not be matched after the comparison operation had completed, then the cluster had not been found. On the other hand, if all data items had a match then the cluster and the markers on it had been identified.

A second check was made to verify that each distance belonged to the correspondent cluster. This was necessary as sometimes, coincidently, two clusters may be close enough to have one of the unique cluster distances between one marker from a certain cluster and one from the other cluster so confusing the algorithm. Further the operator was moving the pointer to register some points of the knee joint, hence the pointer could also interfere in this way. This happened a couple of times during the experimental phases but this error was completely avoided by this checking procedure. To implement this process, a counter was made for each marker as each Euclidean distance was defined by two markers, so for this example each marker of the Tibia cluster will be shared by three Euclidean distances on known length with the other three cluster markers, (Figure 6.10). For example marker(A) will have Euclidean distances with marker(B), marker(C) and diagonally with marker(D) (this can be visualised from the figure as letters A,B,C and D were placed on respective markers), so if the case mentioned before occurred and one marker had equal Euclidean distance with a marker from another cluster, this procedure output would have been five markers where a Euclidean distance had been matched rather than four and two markers from these five will have been confused however, one of them will only have one match in the compared set while the other will have three markers (from the known Euclidean distances of the cluster), the one with three matches (out of 4) is the correct one. For the probes there were three markers and hence three Euclidean distances and if the above error occurred, the correct marker for the probe would be the one with two lengths count matched.

Here is the procedure implementation in LUA code for the green probe.

```
function countMarkers(foundMarkers, number)
found = false
for i = 1, #foundMarkers do
if foundMarkers[i]["Value"] == number then
```

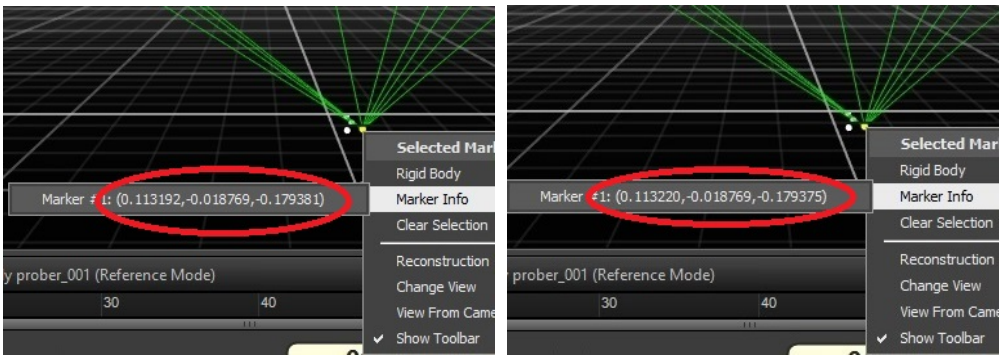
```
foundMarkers[i]["Count"] = foundMarkers[i]["Count"] + 1
found = true
return foundMarkers
end
end
if found == false then
c = (#foundMarkers)+1
foundMarkers[c] = {}
foundMarkers[c]["Value"] = number
foundMarkers[c]["Count"] = 1
end
return foundMarkers
end
function doubleCheck()
foundMarkers = {}
foundMarkers[1] = {}
foundMarkers[2] = {}
foundMarkers[1]["Value"] = greenProbeArray[1][2]
foundMarkers[1]["Count"] = 1
foundMarkers[2]["Value"] = greenProbeArray[1][3]
foundMarkers[2]["Count"] = 1
for i = 2, #greenProbeArray do
foundMarkers = countMarkers(foundMarkers, greenProbeArray
   ↪ [i][2])
foundMarkers = countMarkers(foundMarkers, greenProbeArray
   ↪ [i][3])
end
counter = 1
temp = {}
for i = 1, #foundMarkers do
if foundMarkers[i]["Count"] > 1 then
```

```
temp[counter] = markers[foundMarkers[i]["Value"]]
counter = counter + 1
end
end
if #temp == 3 then
greenProbeArray = temp
return true
end
return false
end
function findGreenProbeArray(A, tF, itrationNumber)
counter = 1
for i = 1, expectedNoOfCombinations do
if A[i][1] >= GreenProbeArrayDims[1]-tF and A[i][1] <=
   ↪ GreenProbeArrayDims[1]+tF then
greenProbeArray[counter] = A[i]
counter = counter + 1
elseif A[i][1] >= GreenProbeArrayDims[2]-tF and A[i][1]
   ↪ <= GreenProbeArrayDims[2]+tF then
greenProbeArray[counter] = A[i]
counter = counter + 1
elseif A[i][1] >= GreenProbeArrayDims[3]-tF and A[i][1]
   ↪ <= GreenProbeArrayDims[3]+tF then
greenProbeArray[counter] = A[i]
counter = counter + 1
end
end
if counter < 4 then
greenProbeArray = {}
elseif doubleCheck() then
return "found"
```

```
elseif itrationNumber < maxNoItrations then
return findGreenProbeArray(A, tF -0.0001, itrationNumber
   ↪ + 1)
end
greenProbeArray = {}
return "not found"
end
```

The functions explained in the code above were responsible for the detection process. The procedure starts by creating the "eculideanDistanceArray" which was a two-dimensional array of Euclidean distances with each element holding the distance between a pair of markers. After calculating all distances, the array was sorted in an ascending order then the function "findGreenProbeArray(A, tF, itrationNumber)" was called from the main loop. At the function call the "A" variable was set as the "eculideanDistanceArray", the "tF" was initialized as the "toleranceFactor", and the "itrationNumber" was set to 1. The function started by setting the counter variable –a counter for the number of found markers– value to 1, then the loop started with the goal of finding all input markers that belonged to the working cluster.

In the loop a comparison was made between each of the Euclidean distance values from the live data and all the previously recorded dimensions array that identified that cluster. However, the comparison wasn't perfect, because when the markers' positions were retrieved, the system noise found there was about 100 µm plus or minus error.



(a) (b)

Figure 6.12: Motive Marker Position Tracking

An experiment was conducted to test a cluster and calculate the difference in marker positions overtime (Figure 6.12). The marker coordinates were captured in Motive and after two seconds the coordinates were recaptured. The markers belonged to the blunt probe set and the probe was stationary on the operation table during the recording. The values of a typical pair of markers changed from 0.113192 to 0.113220 so that was a 28 µm difference. But for a different pair, only a 6 µm difference occurred. This noise had the possibility to confuse the above algorithm. If one of the distances in the sharp probe data (0.106377) were reduced by the same amount it would be smaller than another value in the Femur cluster data (0.106154), see Appendix B for all clusters data. Therefore, a further system of checks was needed for the algorithm to be robust against this noise.

A tolerance was therefore introduced and Euclidean distances tested within a range of values + or - this tolerance. Testing showed that setting the tolerance factor too big caused more distances to be identified. However, setting the tolerance factor too low caused the compared value to be out of the comparison range and hence not found.

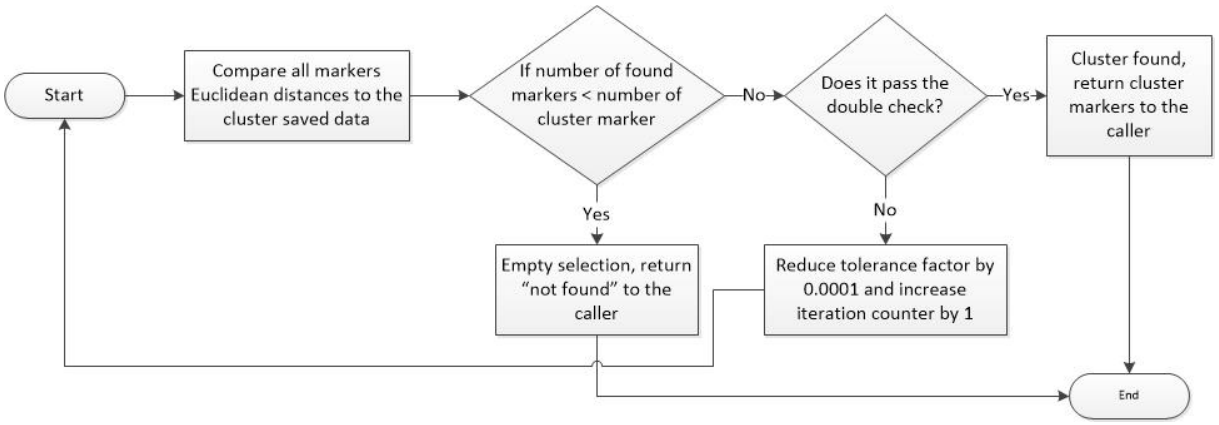


Figure 6.13: Cluster Detection Algorithm 2.0 Flow Chart

As the tolerance value varied, the tolerance factor needed to be varied as well. The detection algorithm started with a relatively large tolerance and if the detection failed the function would reduce the tolerance factor value by a small amount and recursively repeat the same process again until the cluster was uniquely detected, Figure 6.13 shows the flow diagram for the new algorithm.

In the code, a series of conditions were made to organize the recursive call. If the number of in-range markers –counter variable– was less than the number of cluster markers –in this case three markers for the blunt probe– then the function would finish, set the cluster marker's array to empty then return "not found" to the caller. If the counter was equal to or more than the number of cluster markers then a double check would be done; either to make sure that the selected markers were the right markers or to clear the extra ones. If the markers passed the double check, then the function would return "found" to the caller. If neither occurred, the function would repeat but this time with a smaller tolerance factor (tF) value and also the iteration number will be incremented by one. This procedure continued and the (tF) value was decremented for each iteration. If the double check kept on failing, then after 20 iterations the function would stop and return "not found", this would be because of one or more markers from the cluster wasn't in the camera's field of view or was behind an obstacle.

The doubleCheck() function works by checking each marker count, if there were the correct number of markers with the correct number of Euclidean dis-

tances found then the function would return true else it returned false.

For example, in the previous example of the blunt probe, the correct number of markers should be three, and each should have an Euclidean distance found count of two.

If the cluster was found, the cluster's markers location would be forwarded to the next module and also an audio assist announcement would be made for example "tibial cluster found", if the cluster was a pointer type then further calculations would be performed to calculate the tip of the pointer for the registration process. This is explained in the next section.

When each cluster was detected, the "Sounding" script module provided voice assistance to the operator. This "Sounding" script was not mandatory for the application but this feature has provided great help for the operator. Its necessary for the operator to know that the cluster or pointer has been detected by the system in order to insure a correct tracking or registering process. The voice assisting feature negates the need for checking the screen continuously as it speaks out loud when clusters are detected or invisible, as sometimes in the surgery procedure or during using the pointer, the operator gets in the way of the cameras for short periods of time and that could mess up the registration process or other tracking functions. The fact that when a cluster was detected, a signal was sent to this script to play a unique recorded audio message that announced its discovery, and if the cluster went missing the script played another recorded message to announce that the cluster had gone missing proved to be a great assistance to the operator of the system during the cutting procedure.

Here is the script for the voice assistance.

```
--FUNCTIONS-----------------------------------------
function checkNoSoundPlaying()
for i = 1, table.getn(sounds) do
if sound.isplaying(sounds[i]["F"]) or sound.isplaying(
   ↪ sounds[i]["NF"])then
return false
end
```

```
end
return true
end
--init variables-------------------------------------
init = init or 0
allinputs = allinputs or {}
soundOrder = soundOrder or {}
sounds = sounds or {}
previousSoundPlayed = previousSoundPlayed or {0, 0, 0, 0,
   ↪  0, 0}
--init the code--------------------------------------
if init == 0 then
for i = 1, 6 do
allinputs[i] = "Channel␣"..i
sounds[i] = {}
end

sounds[1]["F"] = sound.create("C:/CAREN␣Resources/Sounds/
   ↪ Tibial_Found.wav")
sounds[1]["NF"] = sound.create("C:/CAREN␣Resources/Sounds
   ↪ /Tibial_Missing.wav")
sounds[2]["F"] = sound.create("C:/CAREN␣Resources/Sounds/
   ↪ Femur_found.wav")
sounds[2]["NF"] = sound.create("C:/CAREN␣Resources/Sounds
   ↪ /Femur_missing.wav")
sounds[3]["F"] = sound.create("C:/CAREN␣Resources/Sounds/
   ↪ Endeffector_Found.wav")
sounds[3]["NF"] = sound.create("C:/CAREN␣Resources/Sounds
   ↪ /Endeffector_Missing.wav")
sounds[4]["F"] = sound.create("C:/CAREN␣Resources/Sounds/
   ↪ Base_Found.wav")
```

```
sounds[4]["NF"] = sound.create("C:/CAREN_Resources/Sounds
   ↪ /Base_Missing.wav")
sounds[5]["F"] = sound.create("C:/CAREN_Resources/Sounds/
   ↪ Blue_Probe_Found.wav")
sounds[5]["NF"] = sound.create("C:/CAREN_Resources/Sounds
   ↪ /Blue_Probe_Missing.wav")
sounds[6]["F"] = sound.create("C:/CAREN_Resources/Sounds/
   ↪ Green_Probe_Found.wav")
sounds[6]["NF"] = sound.create("C:/CAREN_Resources/Sounds
   ↪ /Green_Probe_Missing.wav")
sound.setvolume(sounds[1]["F"],100)
sound.setvolume(sounds[1]["NF"],100)
sound.setvolume(sounds[2]["F"],100)
sound.setvolume(sounds[2]["NF"],100)
sound.setvolume(sounds[3]["F"],100)
sound.setvolume(sounds[3]["NF"],100)
sound.setvolume(sounds[4]["F"],100)
sound.setvolume(sounds[4]["NF"],100)
sound.setvolume(sounds[5]["F"],100)
sound.setvolume(sounds[5]["NF"],100)
sound.setvolume(sounds[6]["F"],100)
sound.setvolume(sounds[6]["NF"],100)

inputs.setchannels(unpack(allinputs))
init = 1
end

for i = 1, 6 do
soundOrder[i] = inputs.get("Channel_"..i)
--print(i, soundOrder[i], previousSoundPlayed[i],
   ↪ checkNoSoundPlaying())
```

```
if checkNoSoundPlaying() and previousSoundPlayed[i] ~= 1
   ↪ and
 soundOrder[i] == 1 then
sound.play(sounds[i]["F"])
previousSoundPlayed[i] = 1
elseif checkNoSoundPlaying() and previousSoundPlayed[i]
   ↪ ~= 2 and
 soundOrder[i] == 2 then
sound.play(sounds[i]["NF"])
previousSoundPlayed[i] = 2
end
end
```

First the script initializes by creating sound objects for all recorded audio messages and setting its volume value to maximum. Second reading all incoming inputs from the cluster scripts. Finally, the audio message will be played if it hadn't been played before and if there isn't another audio message commenting playing, else it will be placed in a queue for playing. This was required so that the sounds do not overlap.

The Green probe module had two tasks, to locate each marker right position on the cluster, then to locate the probe tip. In order to accomplish that, markers needed to be established in a consistent order on the cluster. In order to be able to do that, let's take the Tibia cluster as an example, see Figure 6.10, the procedure starts by calculating the Euclidean distance array just as in the previous script, then figure out the markers order. What needs to be achieved is to ID each marker in the software to its known position in the cluster. In the array below four positions each occupied by any of the four markers (Marker A, B, C, D), this positions were marked on the cluster in Figure 6.10, let's assume that the Euclidean distance array was as follows:

Table 6.1: Example of Data Inside the Array

$$
\begin{array}{c}
\begin{array}{ccc} \textit{Eq.Distance} & \textit{Marker1} & \textit{Marker2} \end{array} \\
\left[\begin{array}{ccc}
51.8 & 3 & 4 \\
60.5 & 2 & 1 \\
67.9 & 4 & 2 \\
74.7 & 1 & 3 \\
97.2 & 3 & 2 \\
127.4 & 1 & 4
\end{array}\right]
\begin{array}{c} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \end{array}
\end{array}
$$

The first column in the array represents the Euclidean distances, the second and third columns represent the two markers that have this distance between them. Now, to assign the identified cluster markers (1, 2, 3, 4) from previous script (the unordered markers) to match the marker in correct order/position (A, B, C, D) on the cluster as drawn previously in Figure 6.10. Distance 127.4 mm was is the length of the largest diagonal (Euclidean distance), therefore markers on second and third column which in the last line in table above are markers (1, 4) can be in C and B positions, again see Figure 6.10 for more visualization. Again, from R_6 it is clear that at position C from Figure 6.10 was either marker 1 or 4 from the retrieved markers. From R_3, the distance 67.9 mm was the Euclidean distance between position C and D. So now it is clear that the common position from R_3 and R_6 was the position C which was equivalent to marker (4). Similarly from R_3 the remaining position was D which is occupied by marker (2). By applying the same logic on R_6, it implies that position B was occupied by marker(1), which leaves position (A) to be occupied by marker (3). By applying the same algorithm to each cluster all markers will be matched to their own unique positions on their own cluster, here is the code for this set markers function.

```
function setMarkers()  --to number each marker in the
   ↪ segment
for i = 1, noOfMarkers do
if (i == eculideanDistanceArray[3][2] or i ==
   ↪ eculideanDistanceArray[3][3]) and (i ==
   ↪ eculideanDistanceArray[2][3] or i ==
```

```
   ↪ eculideanDistanceArray[2][2]) then
markers[i]["pos"] = "far"
far = markers[i]
elseif (i == eculideanDistanceArray[1][3] or i ==
   ↪ eculideanDistanceArray[1][2]) and (i ==
   ↪ eculideanDistanceArray[2][3] or i ==
   ↪ eculideanDistanceArray[2][2]) then
markers[i]["pos"] = "left"
left = markers[i]
elseif (i == eculideanDistanceArray[3][3] or i ==
   ↪ eculideanDistanceArray[3][2]) and (i ==
   ↪ eculideanDistanceArray[1][3] or i ==
   ↪ eculideanDistanceArray[1][2]) then
markers[i]["pos"] = "right"
right = markers[i]
end
end
end
```

This code also identified the marker locations on the blunt probe. As the blunt probe had only three markers, the script operation was simpler than for the four marker clusters as the noOfMarkers was three in this script. For each loop the Euclidean distance array were sorted in ascending order. The last two distances (the longest) always had one marker in common. Therefore, the first condition in the loop was to compare and get the software-generated number of that marker, that was done by comparing numbers from 1, 2 and 3 to these two distances and find out the common number. It's position was saved as "far". The same logic was applied to find the order of the other two markers, Therefore, the marker marked as "left" was the common marker in the first two distances and the other marker was the "right" marker. The position naming was set according to the location of the markers on the cluster when facing up and having the base with the two close markers on the operator side.

Now that each marker was in order, a global coordinate system could be made. Each cluster had its own coordinate system (Local coordinate system) as each shape was unique. In order to be able to register the tip of the probe in the global coordinate system, a transformation needed to be done. The steps required to accomplish the transformation are as follows:

1. Establishing probe (local) coordinate system.
2. Creating magnitudes of the local coordinate system.
3. Creating unit vectors of the local coordinate system.
4. Transforming local coordinate system to global coordinate system.
5. Adding the local tip location to the transformation.

The next part will show a portion of the code implementing previous steps and will be followed by step by step illustration.

```
function getAVG(point1, point2)
a = {}
a["x"] = (point1["x"] + point2["x"]) / 2.0
a["y"] = (point1["y"] + point2["y"]) / 2.0
a["z"] = (point1["z"] + point2["z"]) / 2.0
return a
end
function getVector(point1, point2)
vector = {}
vector["x"] = point1["x"] - point2["x"]
vector["y"] = point1["y"] - point2["y"]
vector["z"] = point1["z"] - point2["z"]
return vector
end
function getUNIVector(vector, magV)
vector["x"] = vector["x"]/magV
```

```
vector["y"] = vector["y"]/magV
vector["z"] = vector["z"]/magV
return vector
end
function crossProduct(vector1, vector2)
--[[
Cx = aybz - azby
Cy = azbx - axbz
Cz = axby - aybx
--]]
vector = {}
vector["x"] = vector1["y"] * vector2["z"] -
   ↪ vector1["z"] * vector2["y"]
vector["y"] = vector1["z"] * vector2["x"] -
   ↪ vector1["x"] * vector2["z"]
vector["z"] = vector1["x"] * vector2["y"] -
   ↪ vector1["y"] * vector2["x"]
return vector
end
function getMagnitude(point)
vector = (point["x"]^2 + point["y"]^2 + point["z"
   ↪ ]^2)^0.5
return vector
end
function transform(aP, bP, cP, fourthMarker)
tfm = {}
tfm["x"] = aP["x"] * fourthMarker["x"] + aP["y"]
   ↪ * fourthMarker["y"] + aP["z"] *
   ↪ fourthMarker["z"]
tfm["y"] = bP["x"] * fourthMarker["x"] + bP["y"]
   ↪ * fourthMarker["y"] + bP["z"] *
```

```
    ↪ fourthMarker["z"]
tfm["z"] = cP["x"] * fourthMarker["x"] + cP["y"]
    ↪ * fourthMarker["y"] + cP["z"] *
    ↪ fourthMarker["z"]
tfm["obj"] = markers[4]["obj"]
tfm["pos"] = "fourth"
return tfm
end
function transformTranspose(aP, bP, cP,
    ↪ fourthMarker)
tfm = {}
tfm["x"] = aP["x"] * fourthMarker["x"] + bP["x"]
    ↪ * fourthMarker["y"] + cP["x"] *
    ↪ fourthMarker["z"]
tfm["y"] = aP["y"] * fourthMarker["x"] + bP["y"]
    ↪ * fourthMarker["y"] + cP["y"] *
    ↪ fourthMarker["z"]
tfm["z"] = aP["z"] * fourthMarker["x"] + bP["z"]
    ↪ * fourthMarker["y"] + cP["z"] *
    ↪ fourthMarker["z"]
tfm["obj"] = markers[4]["obj"]
tfm["pos"] = "fourth"
return tfm
end
function getTipLocation()
localOrigin = getAVG(left, right)
a = getVector(far, localOrigin)
bt = getVector(left, localOrigin)
c = crossProduct(a, bt)
b = crossProduct(a, c)
```

```
magA = getMagnitude(a)
magB = getMagnitude(b)
magC = getMagnitude(c)

aUNI = getUNIVector(a, magA)
bUNI = getUNIVector(b, magB)
cUNI = getUNIVector(c, magC)
markers[4] = transformTranspose(aUNI, bUNI, cUNI,
   ↪  fourthMarker)
--markers[4] = transform(aUNI, bUNI, cUNI,
   ↪ fourthMarker)
markers[4]["x"] = markers[4]["x"] + localOrigin["
   ↪ x"]
markers[4]["y"] = markers[4]["y"] + localOrigin["
   ↪ y"]
markers[4]["z"] = markers[4]["z"] + localOrigin["
   ↪ z"]
end
```

After the “getTipLocation()” function was called from the main loop, the tasks on the list starts by setting-up the local coordinate system and this can be achieved by creating three perpendicular axes.

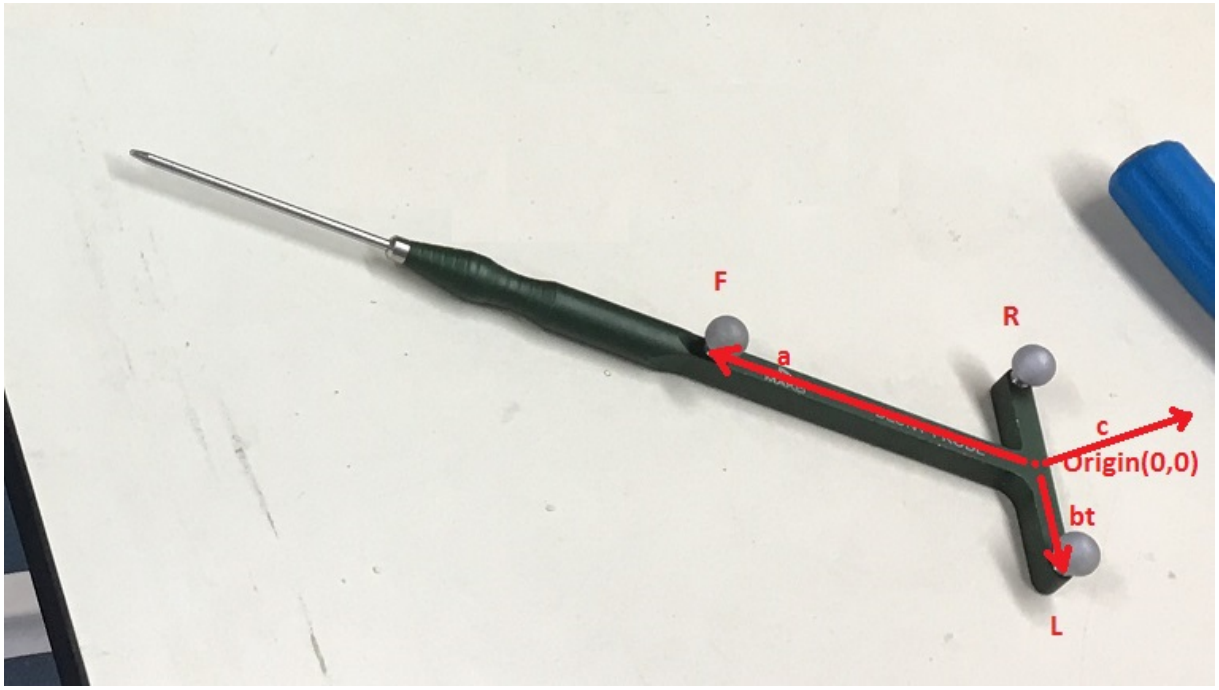


Figure 6.14: Blunt Probe Local Axis

Figure 6.14 shows the local axis that was implemented in the code. The origin was located at the middle point between marker(L) and marker(R), where (L) refers to the left marker and (R) refers to the right marker. The origin was calculated by calling the "getAVG(point1, point2)" function and sending the left and right points which were simply two arrays each holding the XYZ coordinates, the "getAVG(point1, point2)" function received the two points then created a new array to store the new values as shown in the code. Finally, the result was returned to the caller.

The first vector to be created was $\overrightarrow{a}$ which represented the local X axis. This was achieved by calling the "getVector(point1, point2)" function, which simply received the two points that form the vector and then it subtracted the first argument from the second one, afterwards it returned the result array to the caller. So $\overrightarrow{a}$ was created by the equation:

$$\overrightarrow{a} = far - localorigin$$

far and localOrigion are arrays each holding XYZ coordinates. The same rules

were applied to get $\overrightarrow{bt}$:

$$\overrightarrow{bt} = left - localorigin$$

The t in $\overrightarrow{bt}$ stands for temperately because the angle between the $\overrightarrow{a}$ and $\overrightarrow{bt}$ axes was not 90 degrees. In order to produce a local right angle Cartesian axes set. Vector $\overrightarrow{c}$ perpendicular to the plane of $\overrightarrow{a}$ and $\overrightarrow{bt}$ was created using the cross product. This was calculated by:

$$\overrightarrow{c} = \overrightarrow{a} \times \overrightarrow{bt}$$

Refer to Figure 6.14 for the axis direction. Creating the a true perpendicular $\overrightarrow{b}$ was achieved bt the cross product of $\overrightarrow{a} and \overrightarrow{c}$:

$$\overrightarrow{b} = \overrightarrow{a} \times \overrightarrow{c}$$

These calculations were achieved by calling the "crossProduct(vector1, vector2)" function which cross products vector1 by vector2, so if vector1 was v1 and vector2 was v2 and if the result was stored in c array, the following calculations are performed:

$$c_x = v1_y * v2_z - v1_z * v2_y$$

$$c_y = v1_z * v2_x - v1_x * v2_z$$

$$c_z = v1_x * v2_y - v1_y * v2_x$$

In order to produce unit vectors the cross product results need to be divided by the magnitude of the vector. The magnitude of a vector is simply the square root of the sum of all three coordinates squared, for example:

$$\overrightarrow{a} = \sqrt{a.x^2 + a.y^2 + a.z^2}$$

The formula was implemented in a function named "getMagnitude(point)" that took the vector as an argument then return the magnitude to the caller. This was how magA, magB and magC were calculated each to its respective vectors. The Third step was to calculate the unit vector of the three vectors. The unit vector calculation was calculated by dividing each XYZ coordinate by the vector magnitude. getUNIVector(vector, magV) function was created to provide

calculations as:

$$\overrightarrow{a(unit)_x} = \frac{\overrightarrow{a.x}}{a(magnitude)}$$

these three unit vectors form the object rotation matrix.

The inverse rotation matrix for the object was calculated by taking the transform of the rotation matrix. The "transformTranspose(aP, bP, cP, fourthMarker)" function was created to calculate the transpose matrix where aP variable should receive the $\overrightarrow{a}$ unit value, bP should get the $\overrightarrow{b}$ unit value and cP should receive the $\overrightarrow{c}$ unit value. The fourthMarker was an array that held the tip location locally relative to the origin. If the fourthMarker name was L, then the tfm (the transpose function matrix) should be:

$$tfm_x = aP_x * L_x + bP_x * L_y + cP_x * L_z$$

$$tfm_y = aP_y * L_x + bP_y * L_y + cP_y * L_z$$

$$tfm_z = aP_z * L_x + bP_z * L_y + cP_z * L_z$$

Now the final calculation to get the tip location was performed by adding the location of the origin of the probe coordinates (the localOrigin variable) to the transpose translation matrix. After the tip location was calculated, the coordinates were forwarded to the next script.

Moving on to the last script, the "DoCuttingFile" script. This was the last script that was responsible for creating the cutting file which will later be sent to the Arduino machine over the network to do the cutting.

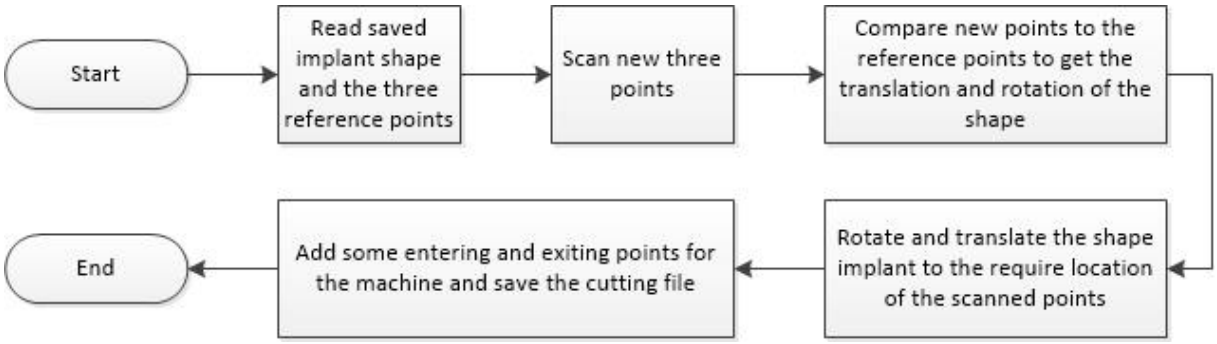


Figure 6.15: Cutting Algorithm Flowchart

During the development of the application there was a shape acquisition process, in which the required implant shape was scanned using the MoCap system and

probe. Three key points on the implant were also captured with the shape file. These three points were used to orientate the implant to the bone. During cutting path alignment these three points were located and used to orientate and align on the bone by the blunt probe the stored shape to the required (on-bone) location. The cutting algorithm flowchart as shown in Figure 6.15. The process started by reading the file in which the implant shape and its reference points were stored. Then, a notification message was played from D-Flow to inform the operator to start locating the three points, which is also known as the registration process of the knee using pointer, in which the blunt probe tip location script was used for this. When the D-Flow alert played, the blunt probe was placed at the first location and the foot switch was pressed to store the tip location at the time, then the same process was repeated for the other two points. The next step was to compare the scanned points (registered points with the pointer) with the stored points (the correlating points from the file) in order to align and orientate the shape to be cut. Accordingly, transformations and rotations were made on the shape file to set the location and orientation of the burr path to that required on the bone.

In order to produce the shape files for the tibia and femur and turn them into cutting file for the burr the following procedure was adopted.

(a) (b)

Figure 6.16: Femur Implant

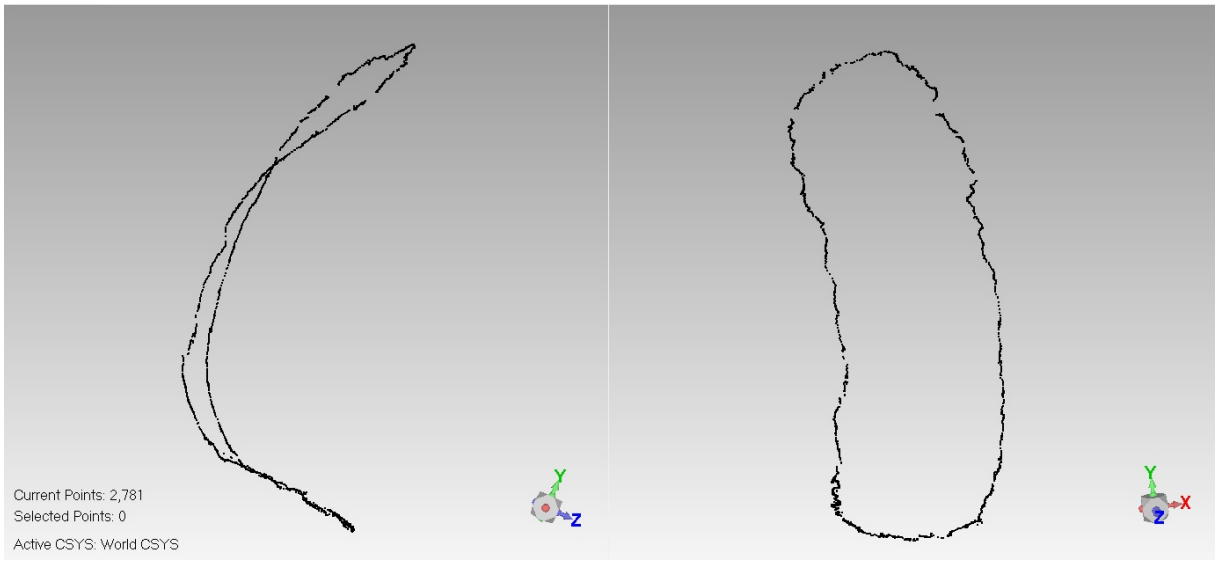


Figure 6.17: Femur Implant Full Resolution

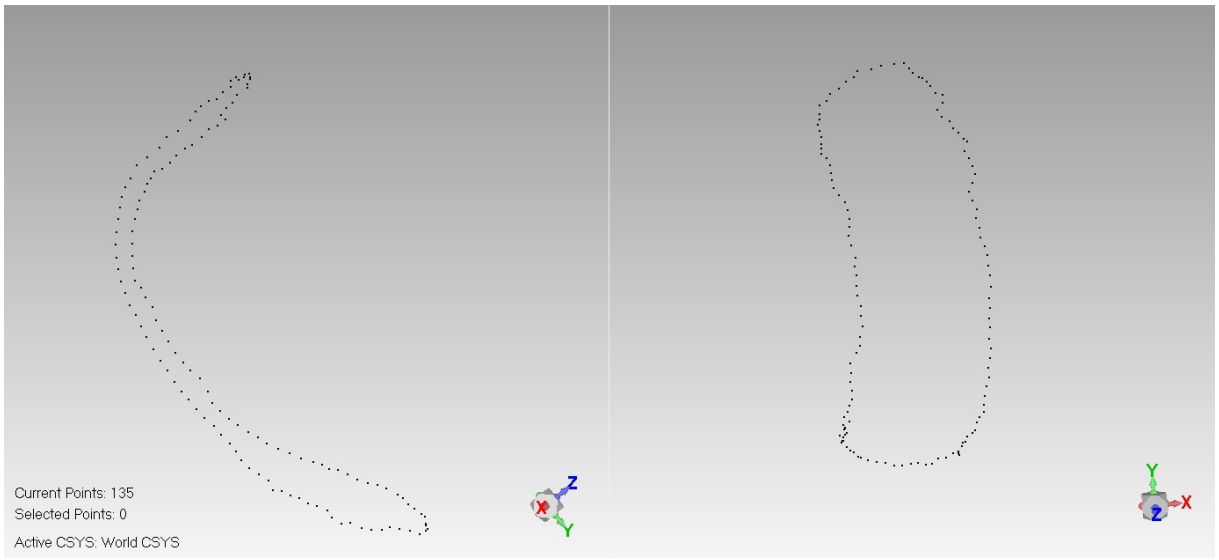


Figure 6.18: Femur Implant Reduced Resolution

In order to be able to make an XYZ file for the implant, a scan was made using the sharp probe and a separate D-Flow application, see section 6.3.2 to follow. The circumference of the implant was scanned, see Figure 6.17 for the femoral implant scan as shown by Geomagic software. The shape consisted of 2781 XYZ points. However, if the data file of the implant in Figure 6.17 was used to create the cutting path, the machine would make 2780 moves excluding the required moves to move in and out of the knee and would only cut round the implant circumference. This number of points would excessively increase the cutting time. The implant shape was therefore reprocessed to reduce the resolution to $1mm$. As the burr spherical tip had a $5mm$ radius, a $1mm$ cutting step should result in a very smooth cut. Geomagic studio was used to apply a $1mm$ uniform distribution to the shape, the output shape is shown in Figure 6.18. The modified shape had 135 points which lead to 134 cutting moves by the machine. Now that the modified implant data file was ready, the path planning algorithm was applied.

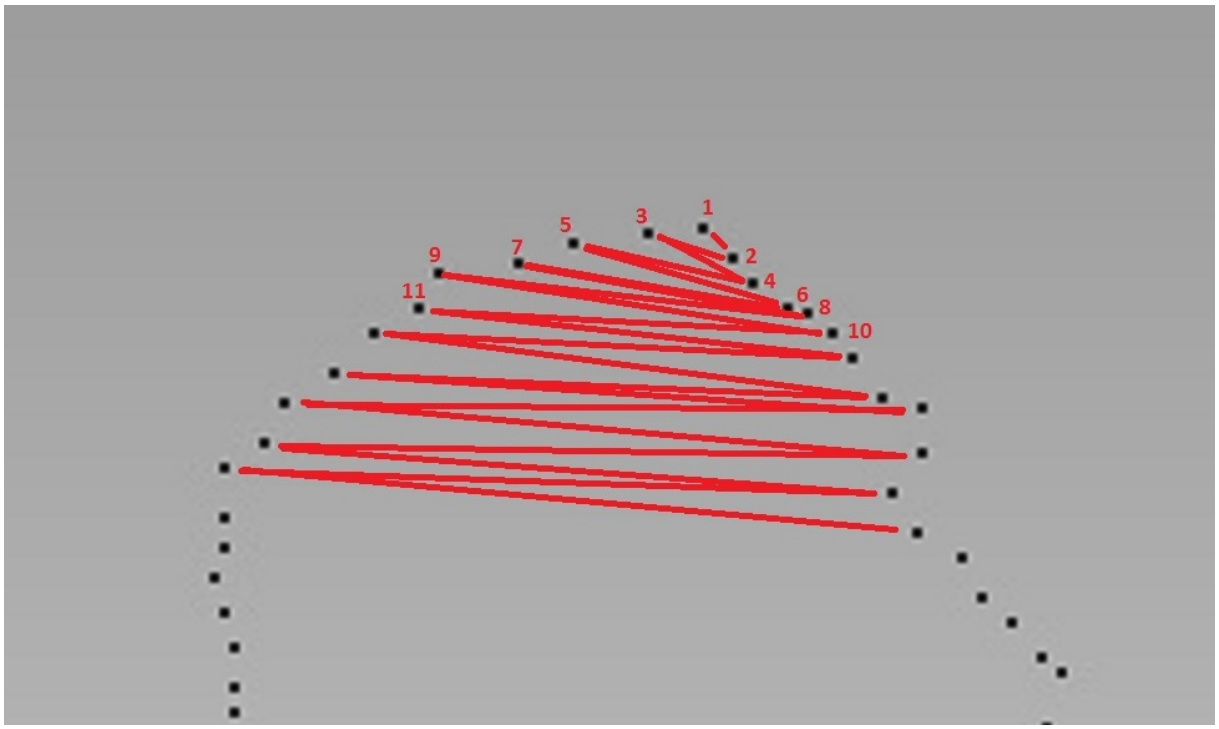


Figure 6.19: Femur Implant Path Planning Algorithm

The path planning algorithm was simple, it picked out the top point according to the global Y-axis, this point was marked as point 1 (Figure 6.19). The algorithm started by calculating the shortest distance from point 1 to the next closest point which was marked as point 2. This was repeated for the next nearest point which was marked as point 3. As points 2 and 3 were the closest, and because the data have been processed in a uniform distribution of $1mm$ between points, point 2 and point 3 had to be in different directions relative to point 1. So, the burr would move into point 1 then towards point 2 then across the surface to point 3. After picking point 2 and 3, the algorithm would again choose the closest point to point 2 which was point 4 and then add it to the path. If then picked the closest point in the other direction, which was the closest point to point 3 and that would be point 5, see Figure 6.19. This cycle would go on until the chain of points were completed and all the points in the implant data list were picked.
After the path file was written and saved, this part of the application was completed. The next step in the procedure was to load and send the file by the Java written communication application over the wireless network to the Arduino to

control the CNC machine and undertake the cutting process, this will be covered in the next chapter.

6.3.2 Developing a Three Dimensional Pointer Scanner Using in Theatre Motion Capture System

Three-Dimensional (3D) surface scanning is the process of obtaining tri-dimensional models of objects. It has been used in many applications including medical and industrial ones. The method used to create a scanner in this project was to use the existing in theatre OptiTrack navigation system to scan and record surfaces in terms of series of points using the blunt probe. While the Mako method captures one point at a time, in reality the navigation system is operating at a 100HZ, it can therefore monitor the position at the tip of the probe at real-time. This allows the probe to be moved across the surface of the bone and for the shape of the bone to be given by the location of the tip of the probe, provided the probe is kept in contact with the bone at all times. In this way a three dimensional scan of the bone surface can be build by moving the probe across the bone surface by continuously monitoring its location using the camera system.

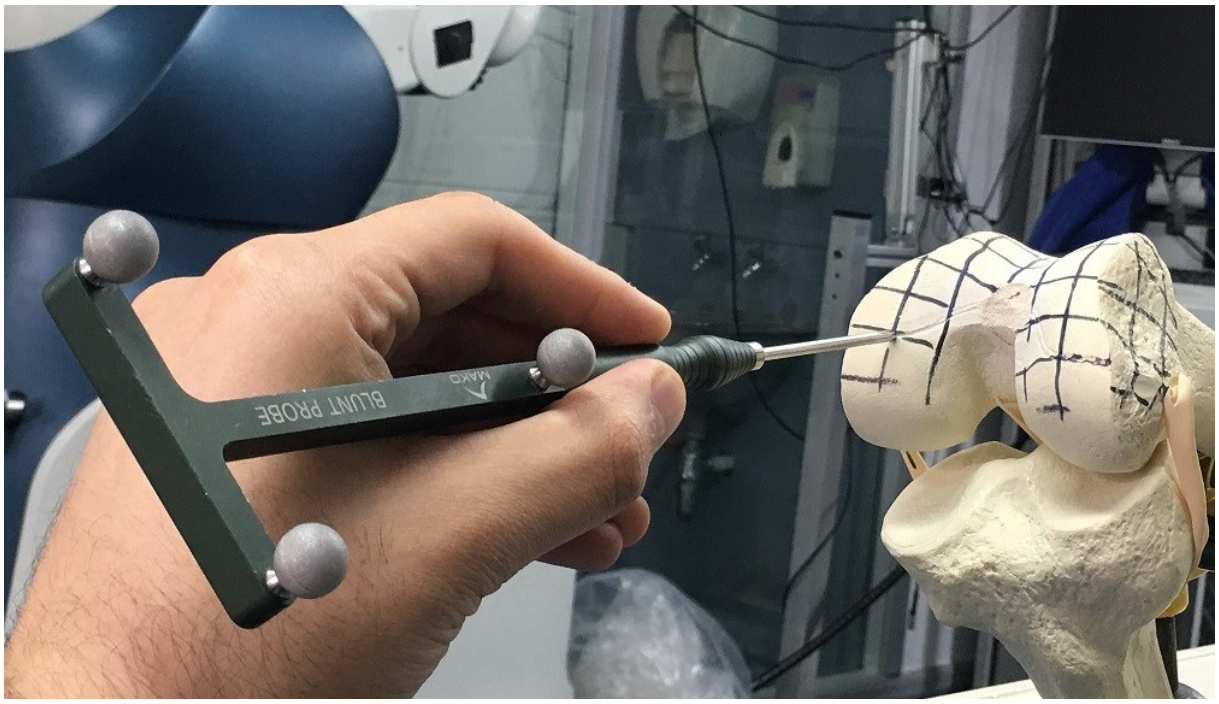


Figure 6.20: The Blunt Probe During Scanning the Femur

Our method used one of the Mako Rio pointers (The Blunt Probe) to undertake this scanning process. As shown in Figure 6.20 above, the pointer had three markers set in fixed locations relative to each other and to the probe.

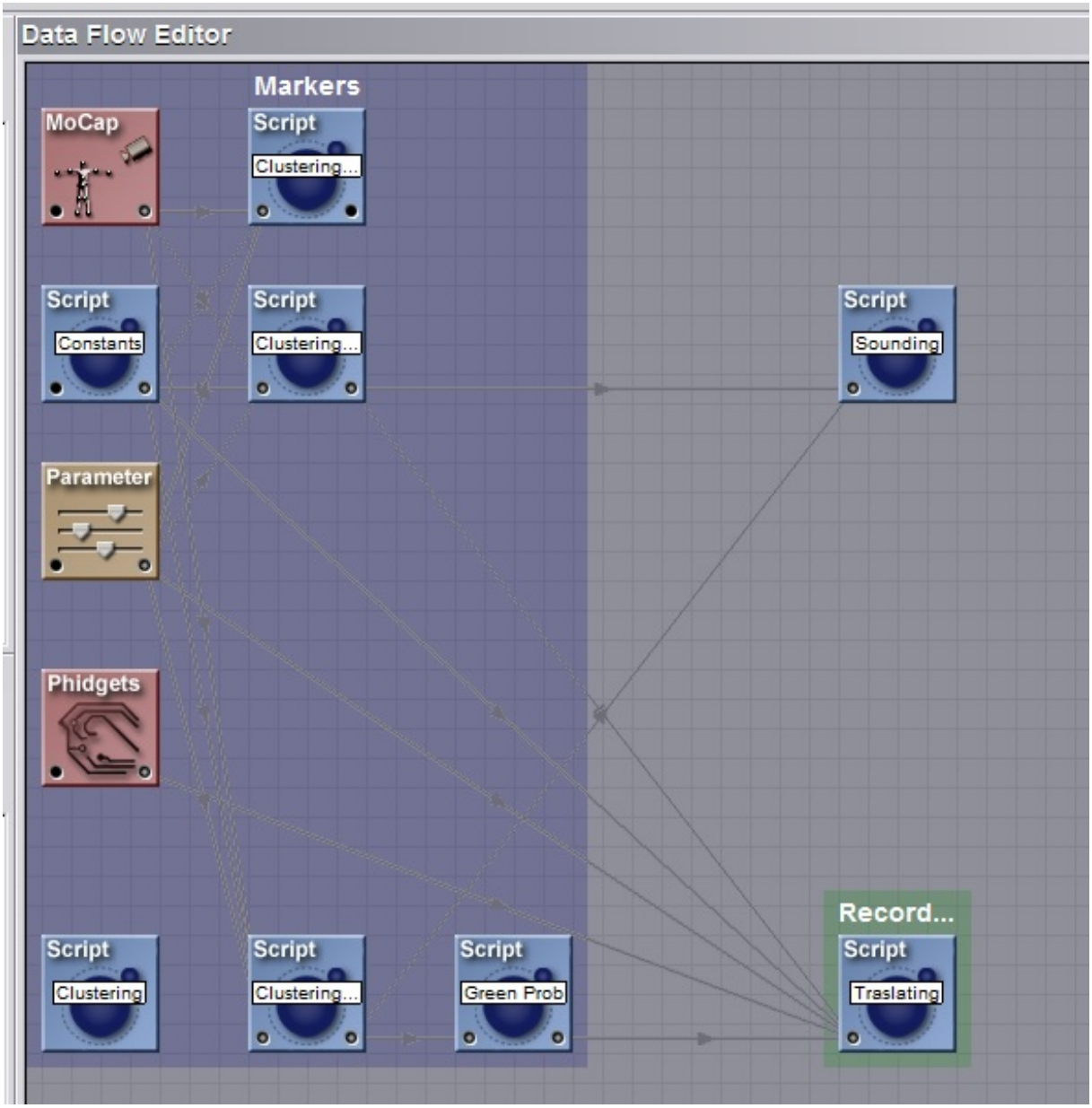


Figure 6.21: Scanning Application D-Flow Editor View

The tracking application was created in LUA via D-Flow software (Figure 6.21). As shown in the figure, all used modules are the same as the main application except for the translating script in the record group (At the bottom right of the figure), the code for the translating script can be found in the Appendix A.2 and has been explained previously.

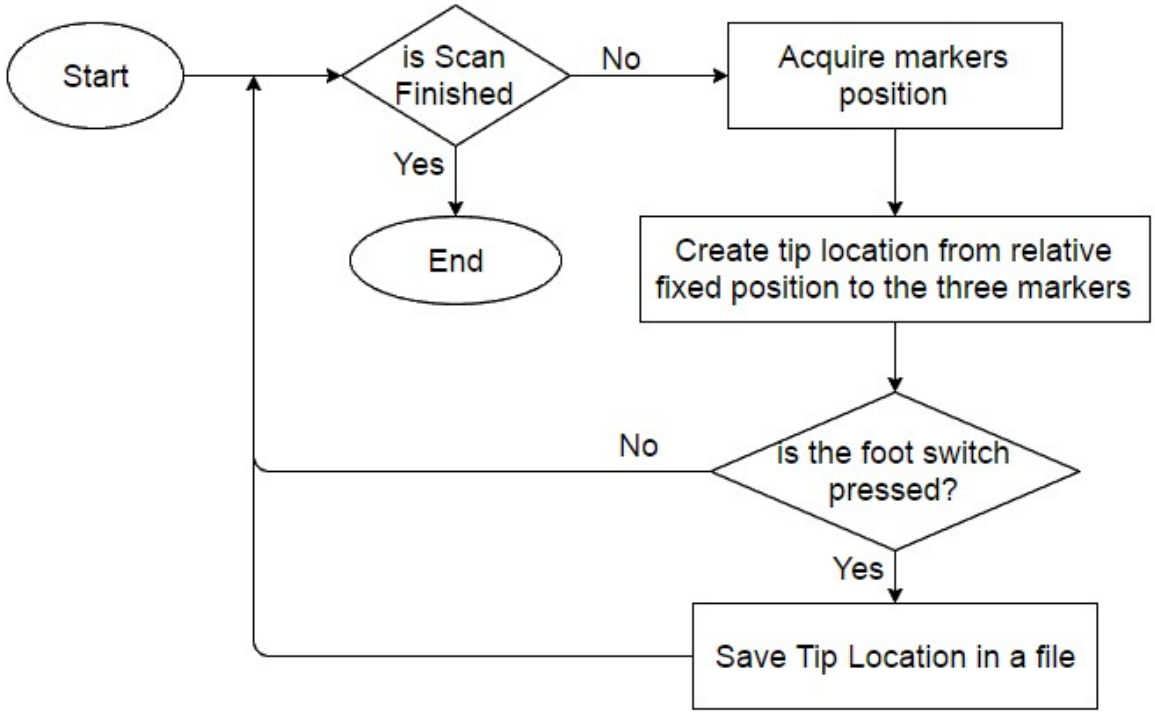
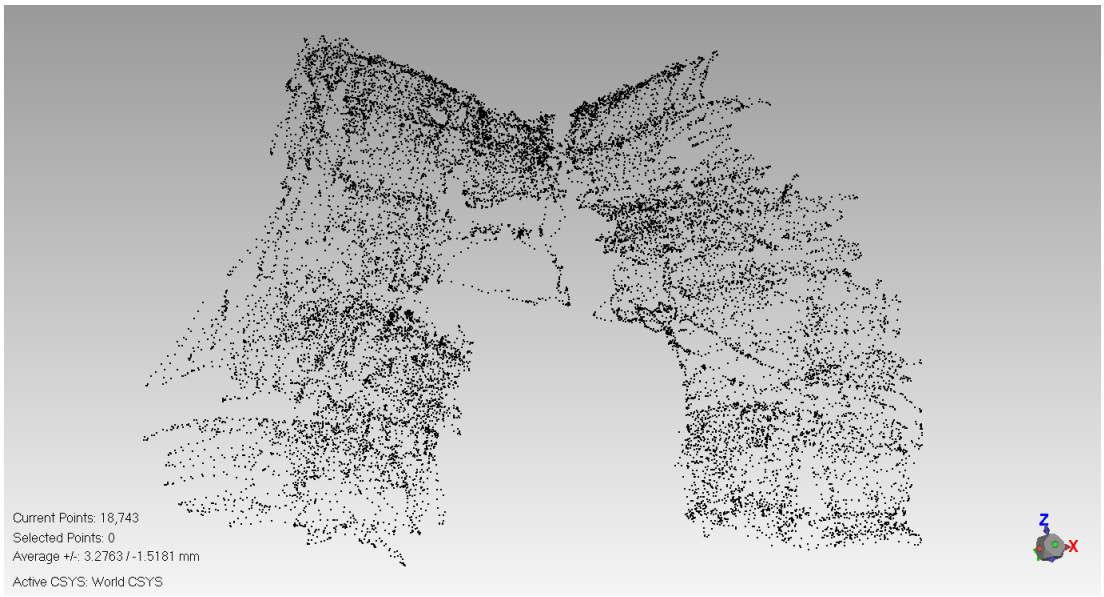


Figure 6.22: Scanning Application Flowchart

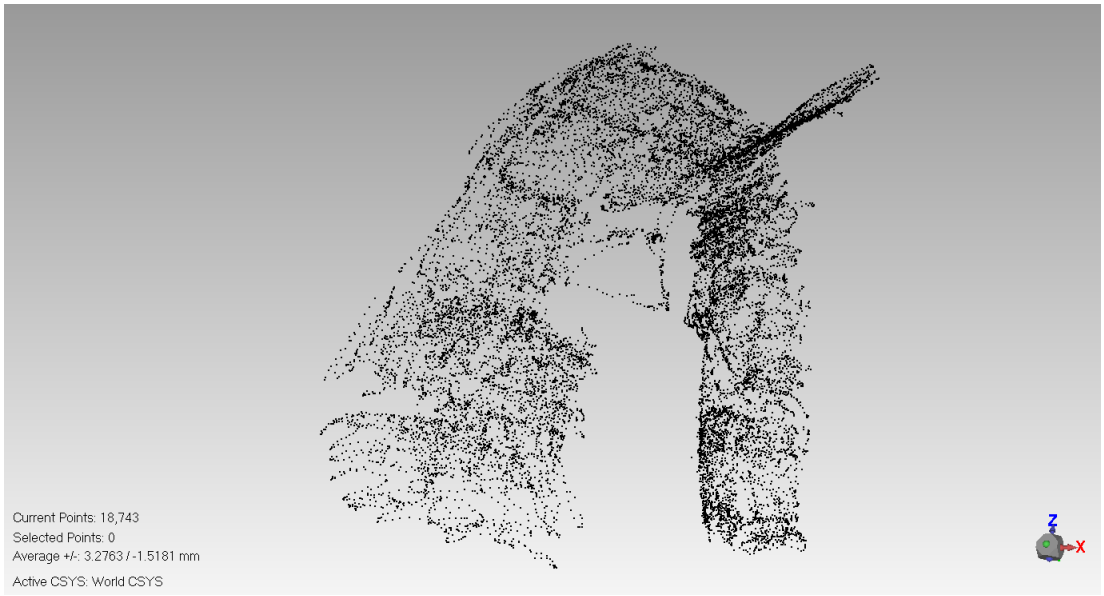
The flowchart shown in Figure 6.22 above illustrates how the application works. First the markers location is acquired and exported from Motive to the LUA module. The LUA module directly calculates the position of the pointer tip location in global coordinates from the Green Probe script as discussed in the previous Section 6.3.1. The application then reads the values of the foot switch from its linked module and if its pressed then the pointer tip location is retrieved and stored in a shape file. If the foot switch is not pressed then the user is probably moving the pointer away from the surface usually to placed it in another desired area to start scanning its surface and hence this data is not stored. The scan is performed by pressing the foot switch while the moving the pointer tip on the target object surface and as the pointer moves, the XYZ locations are stored line by line in the file. The application is set to finish after all the surface has been scanned and the output is a .XYZ point-cloud file which is a readable extension to many 3D mesh software packages such as: Geomagic, Solidworks and Meshlab. The used pointer belongs to the Rio robot but any pointer could be used instead

as long as it has a set of three track-able non-colinear, asymmetric fixed markers giving unique distances between each other and to the pointed end. The distance between each marker is unique, and in this way the application can map the orientation of the pointer.

6.4 Results of the 3D Scanning Application



(a)



(b)

Figure 6.23: Scanned Femur Knee Joint Representation as XYZ Points Frontal and Side View

An experiment was undertaken to test the three-dimensional scanner. The experiment time was relative to the area of the scan. In Figure 6.23 above, 16507 points of the femur knee joint surface were scanned in approximately 9 minutes

which considered to be a long process to be used in theatre of operations. Figure 6.23 shows the point cloud displayed in Geomagic software in two different orientations. The shown figure is the result of a scan of the grid marked area on the bone joint as shown in Figure 6.20.

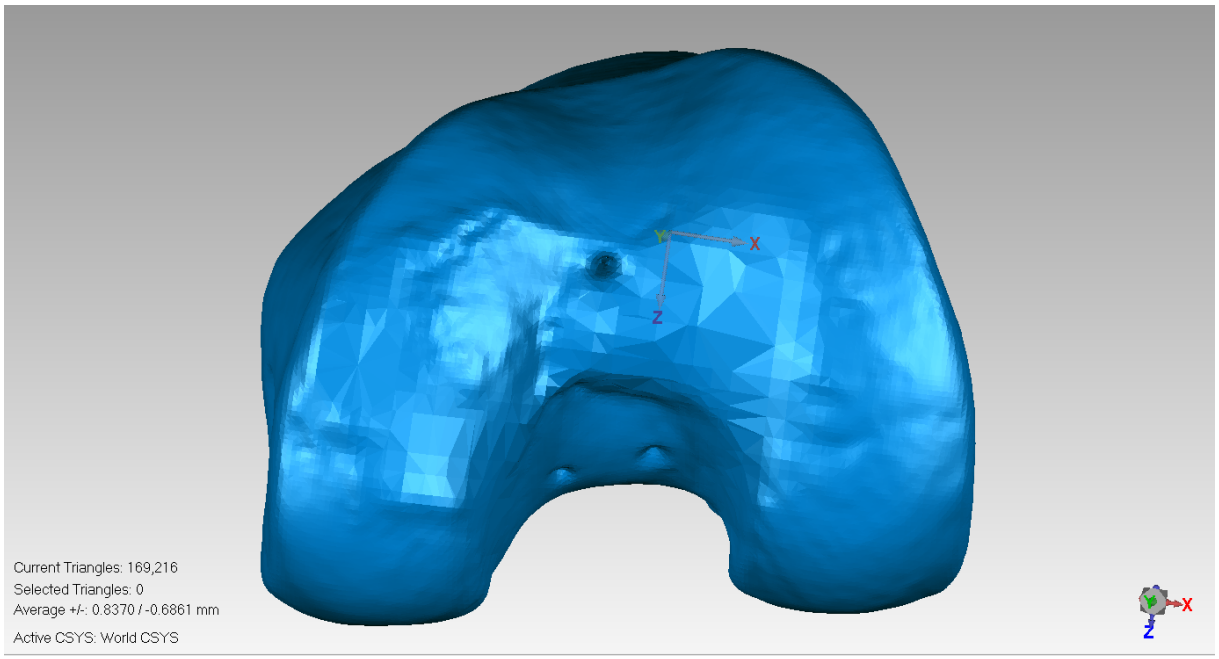


Figure 6.24: Femur Knee Joint Scan by Matter and Form Laser Scanner

To verify the acquired bone shape in Figure 6.23 above, another scan was made of the same Femur bone and the same surface area was scanned using Matter and Form Laser Scanner as shown in Figure 6.24 above. To compare the two 3D shapes in Geomagic, the mocap scanned points had to be wrapped into a body as shown in Figure 6.25.

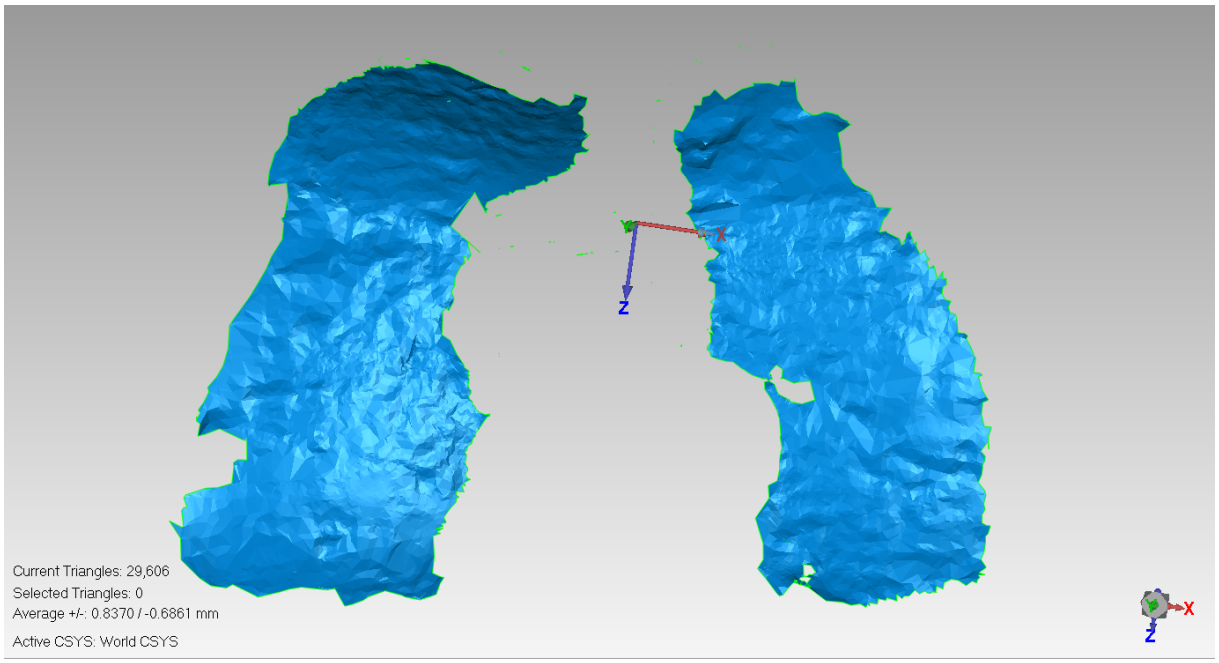
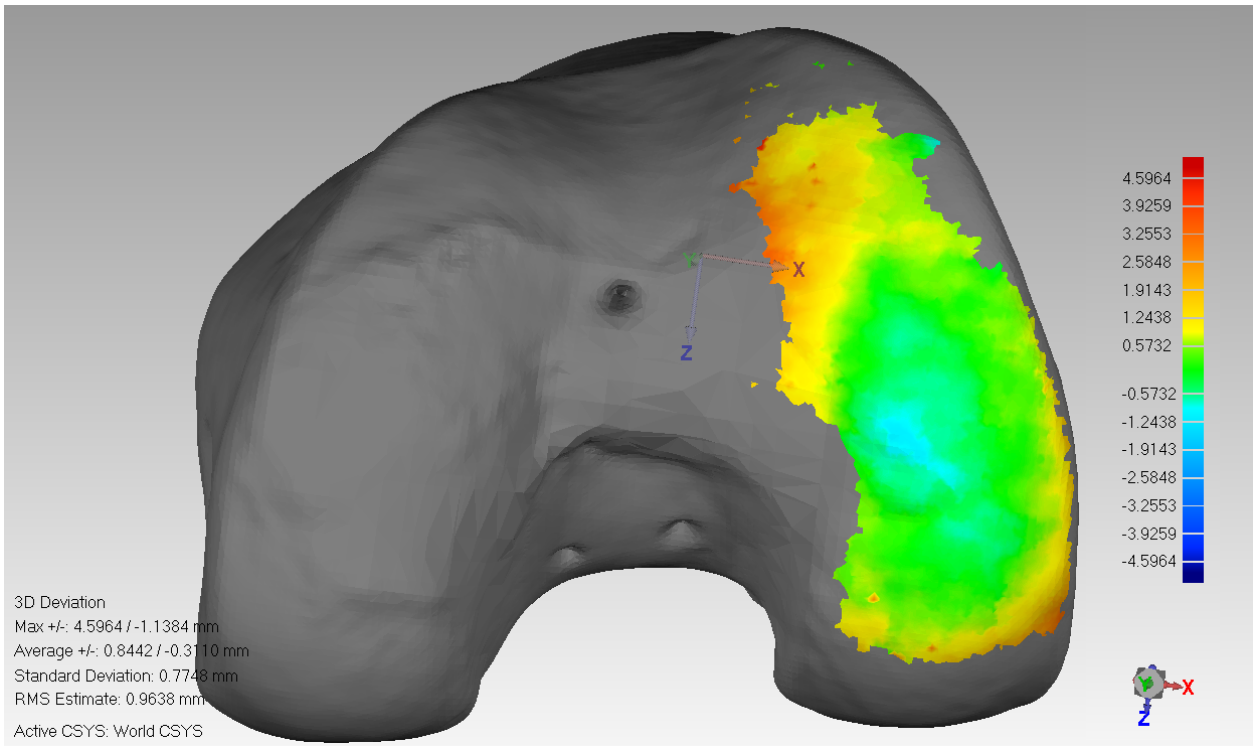
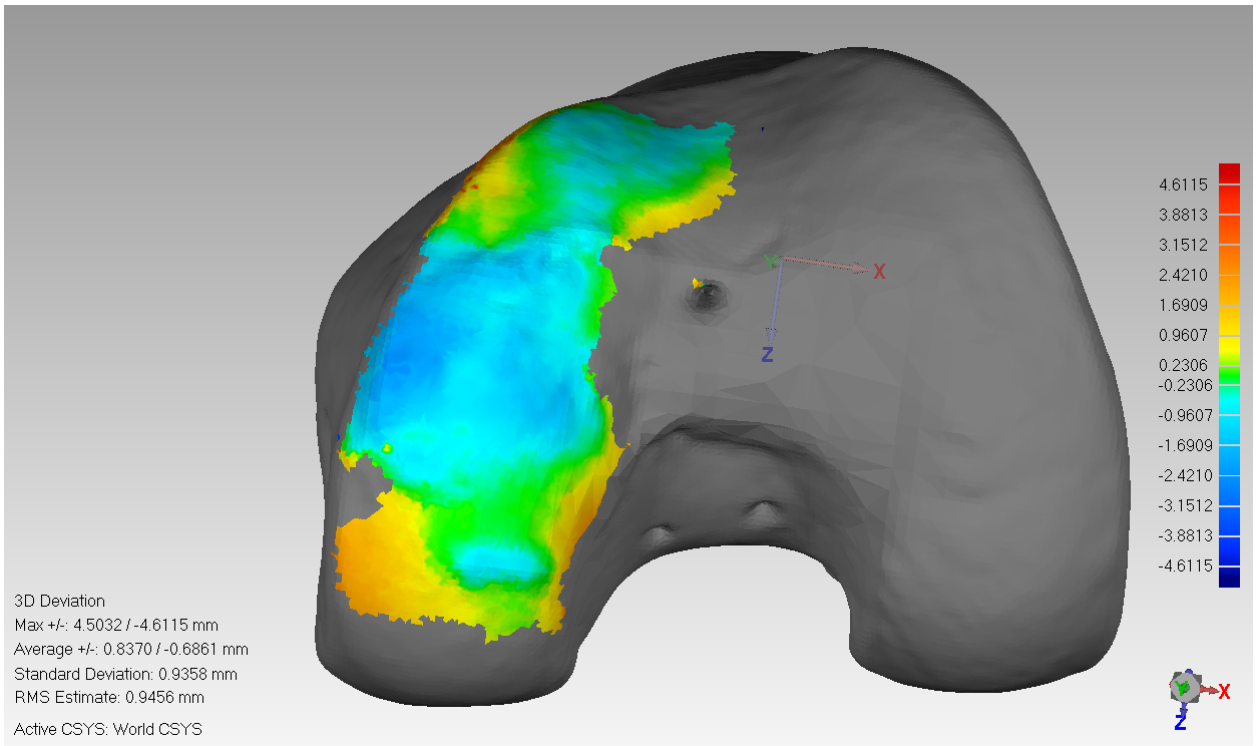


Figure 6.25: The Scanned XYZ Points of the Lateral and Medical condyle Wrapped as a Body by Geomagic



(a) Lateral Condyle



(b) Medial Condyle

Figure 6.26: Comparison Result

The result of the 3D comparison between the two bodies are shown in Figure 6.26.

The first image 6.26a was the comparison result between both lateral condyle. The images show a coloured representation of the Euclidean distance between both objects as each point was compared with the nearest and the difference in distance was represented by a colour. Each colour have a range which is displayed on the scale shown on the left side of the image. The green colour means that points where identical, then the scale shows two different colours, yellow for tolerance of $0.57mm$ and light blue for tolerance of -$0.57mm$. The second image 6.26b was the comparison between both medical condyle. The image shows some green colour but mostly light blue but in this comparison the light blue tolerance was -$0.23mm$, as each comparison have different scale. The image also shows little parts of yellow which have tolerance of $0.23mm$.

6.5 Conclusion

The D-Flow Software has proved to be ideal for supporting motion capture systems and provide sufficient support to develop the prototype. D-Flow software was used to create the main application for the procedure. The MoCap module was used to assist marker coordinates transfer from Motive software to the scripting modules. D-Flow also provided other modules for communication. The Phidgets module was used to take input from a Phidgets foot switch and other modules were used to input application data at runtime.

D-Flow also allows scripting modules using LUA programming language. Various scripts were written to implement the needed procedures and to complete the creation of the cutting file which was responsible for controlling the machine. After the cutting file was made, another application was created by Java, which was responsible for sending the file to the machine over the wireless network. This application will be discussed in the next chapter. However, D-Flow was used to call and run java applications.

D-Flow allowed the development and synchronization of the applications to occur with relative ease. This is a process often difficult in other software platforms which were not designed for real-time visualization of motion capture data. The

data produced and recorded by D-Flow was sufficiently consistent for the desired application.

Creating a three-dimensional scanner with the OptiTrack motion capture system and a tracked pointer was successful with a tolerance of a submillimeter.

Chapter 7

Methods: Overview of Communication

7.1 Introduction

In the developed system, the surgery required an advanced communications technology in order to enable the surgeon to remotely control the CNC robotic machine described previously while in the operating theatre. The Java application (explained in the Methods section of this chapter) communicated with the Arduino board controlling the robot via an Arduino Wi-Fi shield. Network control was required on two levels, either manual control or automatic control which enables the operator to work remotely from the machine. Also, a control override was required and was implemented using hard wired joysticks on the CNC machine to provide direct control.

7.2 Aims and Objectives

- Create Application to safely send the cutting file over the network (the Java application/ client side).
- Create Application to safely receive the file and drive the motors precisely (The Arduino application/ server side).

- Create robust network protocols to insure safety data transmission over the network.
- Provide manual control feature to the operator.

7.3 Methods

The first part of the method outlines the Java application which acted as the client side of the network communication and transmit the cutting file from the computer to the Arduino micro-controller.

7.3.1 The Java Application

This application was implemented with a graphical user interface to assist the selection and manual control operations and was resident on the PC.

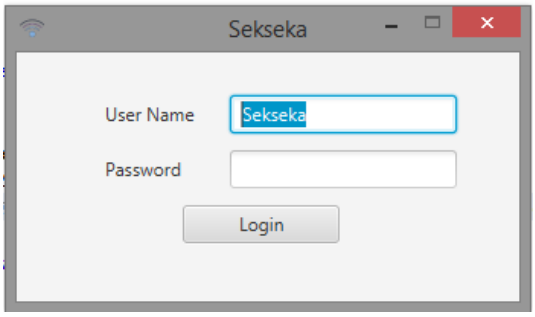
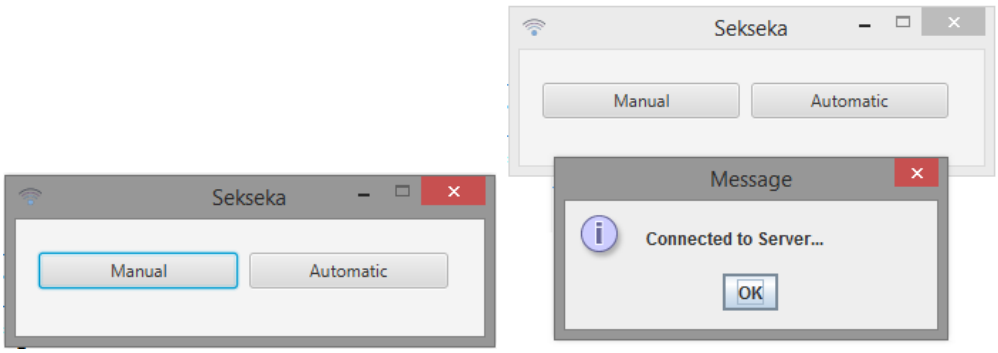
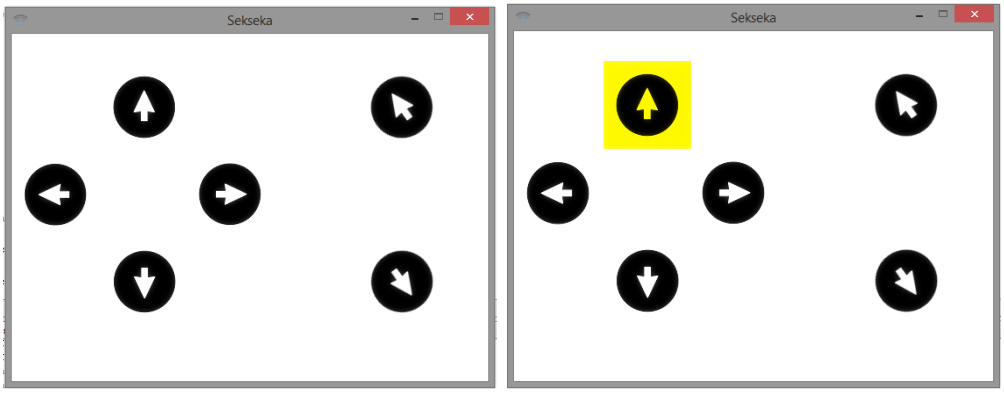


Figure 7.1: Java Application Login Window



(a) (b)

Figure 7.2: Java Application Selection Window



(a) (b)

Figure 7.3: Java Application Manual Control Window

The application starts with the login window, Figure 7.1. The operator must provide the user name and password to access the selection window shown in Figure 7.2. The selection window provides a selection between "manual" or "automatic" control. If the "automatic" option is selected, the application initiates communication with the server (The Arduino Application) and sends the word "Automatic". If the "Manual" option is selected, the application does the same and starts communication with the Arduino application but with the word "Manual" sent to the server. After communication is initialized, another window opens to provide control for the operator, Figure 7.3. Here is the Controller class of the Java code for the Application:

```
1  package sekseka;
2  import java.io.BufferedReader;
3  import java.io.FileReader;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.io.PrintWriter;
7  import java.net.Socket;
8  import java.util.logging.Level;
9  import java.util.logging.Logger;
10 import javafx.scene.control.Label;
11 import javafx.scene.paint.Color;
12 import javax.swing.JOptionPane;
13
```

```
14 /**
15 *
16 * @author Omar Shalash
17 */
18 public class Networking extends Thread
19 {
20   private String mode;
21   private PrintWriter out;
22   private BufferedReader in;
23   private BufferedReader implant;
24   private Label statusLabel;
25   private final int COMMON_DELAY = 20;
26   private int countLoops = 0;
27   Networking(Label statusLabel, String mode)
28   {
29     this.statusLabel = statusLabel;
30     this.mode = mode;
31   }
32   @Override
33   @SuppressWarnings("SleepWhileInLoop")
34   public void run()
35   {
36     if(mode.equals("Manual"))
37     {
38       try
39       {
40         String direction, previous = "notSetYet";
41         Socket client = new Socket("192.168.2.100", 80);
42         if(client.isConnected())
43         {
44           JOptionPane.showMessageDialog(null, "Connected to
                 ↪ Server...");
45         }
46         else
47           JOptionPane.showMessageDialog(null, "Connected NOT to
                 ↪ Server...");
48         out = new PrintWriter(client.getOutputStream(), true);
```

```
49          in = new BufferedReader(new
                ↪ InputStreamReader(client.getInputStream()));
50          System.out.println("Sending!!");
51          out.println("Manual");
52          while(!in.ready())
53            Thread.sleep(1);
54          System.out.println("Reading!!");
55          while(!in.readLine().equals("Manual"))
56          {
57            out.println("Manual");
58          }
59          out.println("OK");
60          System.out.println("OK");
61          while(true)
62          {
63            System.out.println(++countLoops);
64            direction = FXMLDocumentController.getDirection();
65            System.out.println(direction);
66            out.println(direction);
67            previous = direction;
68            Thread.sleep(COMMON_DELAY);
69          }
70        }
71        catch (Exception ex)
72        {
73          System.out.println(ex.getMessage());
74          out.close();
75        }
76      }
77      else if(mode.equals("Automatic"))
78      {
79        try
80        {
81          Socket client = new Socket("192.168.2.100", 80);
82          if(client.isConnected())
83          {
84            JOptionPane.showMessageDialog(null, "Connected to
```

```
            ↪ Server...");
85          }
86          else
87            JOptionPane.showMessageDialog(null, "Connected NOT to
                ↪ Server...");
88          out = new PrintWriter(client.getOutputStream(), true);
89          in = new BufferedReader(new
                ↪ InputStreamReader(client.getInputStream()));
90          out.println("Automatic");
91          while(!in.ready())
92            Thread.sleep(1);
93          while(!in.readLine().equals("Automatic"))
94          {
95            out.println("Automatic");
96          }
97          out.println("OK");
98          implant = new BufferedReader(new
                ↪ FileReader("C:\\burring.txt"));
99          //implant = new BufferedReader(new
                ↪ FileReader("C:\\MovementsPins.txt"));
100         //implant = new BufferedReader(new
                ↪ FileReader("C:\\Movements LowerPin.txt"));
101         //implant = new BufferedReader(new
                ↪ FileReader("C:\\Filling.txt"));
102         String line;
103         while(true)
104         {
105           System.out.println(++countLoops);
106           line = implant.readLine();
107           System.out.println(line);
108           out.println(line);
109           while(!in.readLine().equals(line))
110           {
111             out.println(line);
112             Thread.sleep(COMMON_DELAY);
113           }
114           out.println("OK");
```

```
115                 System.out.println("OKed");
116               }
117           }
118           catch (Exception ex)
119           {
120             //statusLabel.setTextFill(Color.RED);
121             //statusLabel.setText("Not Connected");
122             System.out.println(ex.getMessage());
123             out.close();
124           }
125       }
126     }
127 }
```

The Networking class is initiated when either the “Manual” or “Automatic” button is clicked in the Java application window, see Figure 7.2. If the “Manual” button is clicked then the Network class constructor is invoked with a status label and “Manual” as String data type to initialize the mode variable. If the “Automatic” button is clicked then the constructor is initialized with and “Automatic” for the mode variable.

As the Networking class inherits the properties of the “Thread” class then the run method will be invoked automatically when the start method is initiated from the Network created object, the run method will then work in parallel with the main code. Inside the run method a condition was set to distinguish between the two modes: Manual and Automatic.

In the Manual mode the program establishes a connection with the server (the Arduino application) based on a User Datagram Protocol (UDP). In the manual condition a socket was created to the server stream with the server IP and application access port number.

After the connection is initiated, a GUI message notifies the operator that connection to the server has been achieved. The next step is creating input and output streams using the established socket to enable sending and receiving of text characters between the server and client. The client then sends “Manual” or “Automatic” to the server so that the server can redirect the received text into the

correct part of the program. The server now returns the same word –"Manual" or "Automatic"– as an acknowledgement of receiving the message. The client waits until the server sends the text. If the received text wasn't as expected, then the client application will resend the word again to the server. This process will go on until the server returns the expected word. After receiving the correct word, the client application sends "OK" to the server and communication is live.

In the manual client application, the code now retrieves the direction indicated by the control buttons on the CNC machine from the controller class and sends this information to the server. Then it waits for 20ms and then sends the new retrieved direction. This process goes on as long as the connection is maintained. If the client or the server closes the connection, this loop breaks and the application will jump to the catch block to handle the exception. At the catch block a message will notify the operator that the connection has closed and will then close the connection from the client side.

The 20ms delay –the COMMON_DELAY constant– is very important to maintain the synchronization between the client and server. If the delay time between sending lines of text were shorter or longer it would create the producer consumer problem, in which either the receiver(the server) would run out of buffer because the sender(the Java application) maintained too fast a rate of sending data, or the server would remain idle for too long and may time-out.

In the Automatic mode the client program on the PC implements a Transmission Control Protocol(TCP) which insures that all transmitted data are received at the receiver side (the server), which in this case was the path coordinates of the cutting burr from the cutting file.

In the automatic mode the application starts in the same way as the manual mode by establishing a connection with the server. Also, after creating input and output streams, the application starts by sending the word "Automatic" to the server and waits until the servers reply back with the same word –"Automatic"– as an acknowledgement of receiving the selection key word and redirects the code to receive movements coordinates instead of directions from the control buttons. The application then sends "OK" to notify the server that the coordinates sending

process has initiated. The client application then sends the cutting file using a buffer controlled by the BufferedReader class.

The sending file code starts by taking one line from the buffer then sends it to the server which in return replies with the same line to acknowledge receiving it. If the line received by the application is the same as the sent one, then the application moves on and sends the next line. If the line is different, then the server hasn't received the line correctly and the application sends the same line again. This process goes on until the buffer is empty.

As before, a time delay is needed after each line send operation to keep the server and client in sync. In this operation the same constant of 20ms was used (COMMON_DELAY). If the connection times-out or closes then the code moves to the catch block to handle the exception as before and will reconnect and resend the whole file. Once the sending part is complete the program returns to the main menu.

7.3.2 The Arduino Application

The Arduino application forms the server-side of the communication link. It starts by initializing the libraries and calling the setup() function once in order to initialize necessary variables and set-ups needed for the board utilities. In this application four libraries were used, see Appendix C for the application code. The first library included was "LiquidCrystal" which is responsible for supporting the used LCD. In order to initialize this library the LCD pins were fixed into the code using the following code line:

```
1   LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
```

The pins number(4,6,11,12,13,14) from the LCD, as mentioned in 5.3.3, will be wired to pins (13, 12, 11, 10, 9, 8) correspondingly on the Arduino board, see Figure 7.4 below.

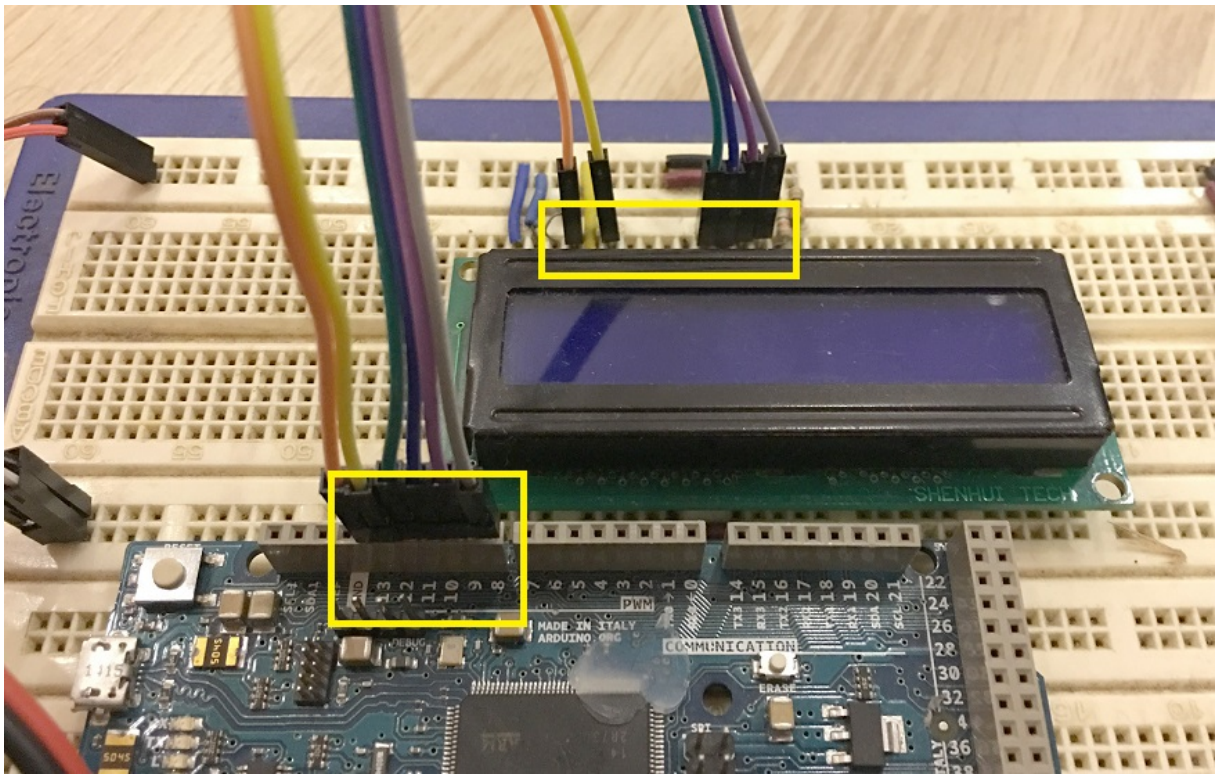


Figure 7.4: Arduino Board Wiring with the LCD

Now that the library is initialized, all library properties can be used using the "lcd" object.

The second library that is included is the "AccelStepper" library, it provides all needed control for the CNC motors by controlling the speed and distance in terms of motor turns. For example, if the motor needs to move 10 *mm*, the distance function of the motor created object will take 2000 as its argument input which also represents the number of turns needed for the motor to move it's carriage 10 *mm*, refer to Chapter 5 for more data about the design.

The next line of code below was used to create motors objects from "AccelStepper":

```
1 AccelStepper stepperM(1,MOTOR_M_PULL, MOTOR_M_DIRECTION);
```

The first argument in the constructor was set to 1 to declare that a driver was used to control the motor and only two wires were required to control the driver from the Arduino board. The other two arguments were to assign the pull and direction pins from the motor driver to the Arduino pins. "MOTOR_M_PULL"

and "MOTOR_M_DIRECTION" in the code were previously defined integer constants. The same method was used to create and initiate the other motor drivers, (Appendix C).

The Third library was the Serial Peripheral Interface library "SPI", which provides serial communication between the Arduino board and the computer using a USB cable. The main usage of the "SPI" library for this application was for monitoring and tracing the code while running. The library functions were used using the associated object "Serial".

The fourth library included in the program is the "WiFi" library. It provides control over the used Arduino Wi-Fi shield module. The next lines of code were required to set up the shield:

```
1  IPAddress ip(192,168,2,100);
2  WiFiServer server(80);
3  WiFiClient client;
```

The first line of the code above was used to set this specific IP address in the "IPAddress" class. The second line assigns the port number. The third line creates a client object, this object would include all client data such as the client IP address and input and output stream. This object initiates with these data when the server(this application) accepts the client(the computer running the Java application) request.

Moving on to the setup() function, this function runs only once when the Arduino board gets powered on, after it finishes the loop function takes over and loops continuously. The first thing to be initialized is the serial communication, the code line below was used to setup a 9600 bps transmitting speed for serial communication:

```
1  Serial.begin(9600);
```

The next on the setup list is the lcd, the first line of code below specifies the lcd size which is 16x2 (16 columns and two rows). The second line prints welcome message on the LCD screen, see code below:

```
1  lcd.begin(16, 2);
```

```
2   lcd.print("Welcome...");
```

The print function will start printing the string from where the LCD cursor was located. If the cursor wasn't moved then the print command would start printing at (0,0) location which represents the first cell in the LCD.

The next piece of code implements a check on the Wi-Fi shield board in case the board was removed or damaged. The first line in the code below is a condition in which status function is called, each return value of the function represents a different condition; if the returned value is equal to 255 that would indicate that there is no Wi-Fi shield connected, all status values are stored as constants inside the library, for example the "WL_NO_SHIELD" has the value 255 stored in as a constant.

```
1   if (WiFi.status() == WL_NO_SHIELD)
2   {
3     lcd.setCursor(0, 1);
4     lcd.print("Shield problem!");
5     Serial.println("Shield problem!");
6     // don't continue:
7     while (true);
8   }
9   WiFi.config(ip);
```

If the shield status was "no shield found" then two alerts are produced to inform the operator. The first alert prints "Shield problem!" on the LCD after moving the cursor to first column and the second row (0,1), see Figure 7.5 below for examples about cursor locations mapping.

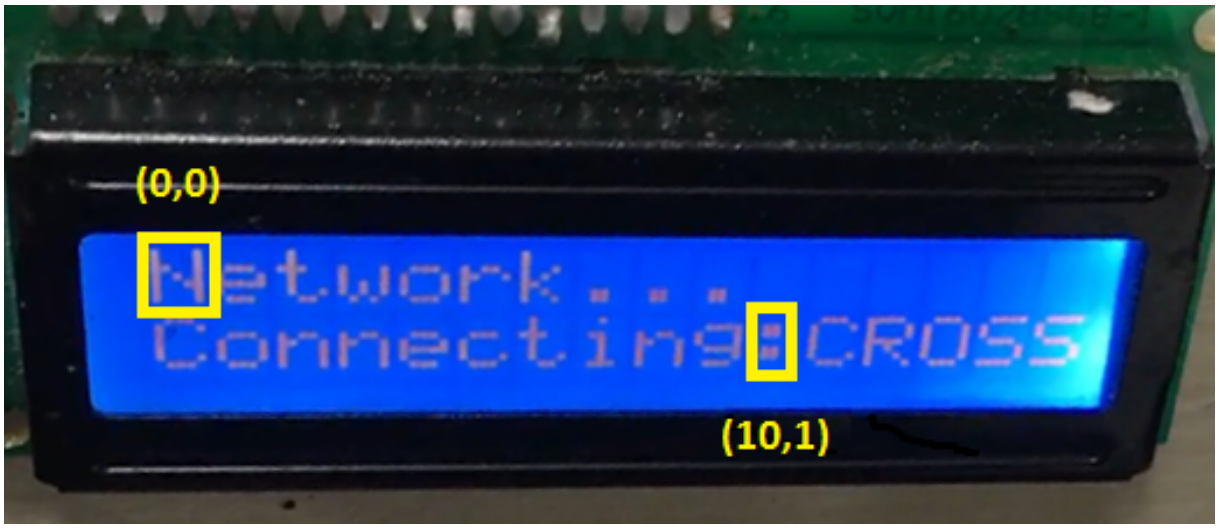


Figure 7.5: LCD Cells location Noted

The second alert sends the same message over the serial communication. On the seventh line of the code the program freezes on infinite loop. If the condition was false, on line 9 of the code above, the config() function was called to set the previously created IP to the shield. Now that the board has been set with a specific IP address and port number it can act as a server and accept clients.

The rest of the setup() function initializes more variables, the code below is for setting the speed, acceleration and maximum speed for one of the motors.

```
1   stepperM.setMaxSpeed(MAX_SPEED);
2   stepperM.setSpeed(MAX_SPEED);
3   stepperM.setAcceleration(MAX_ACCELERATION);
```

This code was repeated to set all five motors where "MAX_SPEED" and "MAX_ACCELERATION" were an already defined constants with the values 3000 and 2500 correspondingly. The speed and acceleration had to be the same values so that the movement of the two motors on the same axis were synchronized.

The next function is the loop function. It starts with a condition of the mode. The mode variable was initially set to zero. So, when the loop function starts for the first time, the first mode condition would be successful as (mode == 0) condition would be true. The code in this condition was for the operator to select the control mechanism of the machine either manual control using the joysticks

or the network control. The first step to implement this task was to print the selection message for the operator using the LCD library functions. The LCD cursor was moved to the second row and then the message “Manual<.>Network” was printed, as the LCD was positioned between the two joysticks, see Figure 7.6 below, then pressing the joystick on the left side of the image would select the left side of the text, the “Manual” option and pressing the right joystick in the image selects the “Network” option.

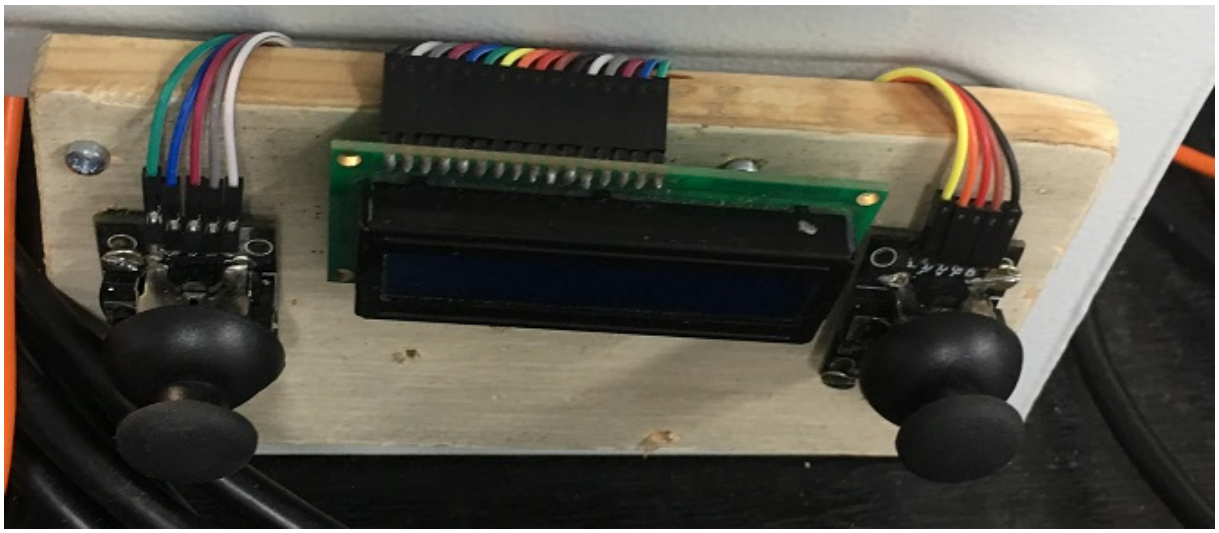


Figure 7.6: LCD Position Between Joysticks

After the LCD was set, the getReadings() function is called. This function was created to take in joysticks analog input. The code below shows the getReadings() function along with other functions which together form full management and control of the inputs performed by the operator:

```
1  void getReadings()
2  {
3    if(checkReset())
4      return;
5    //Joystick 1 >>
6    if(analogRead(JOYSTICK1_VRX) > 790)
7      directions[0] = "DOWN";
8    else if(analogRead(JOYSTICK1_VRX) < 720)
```

```
9       directions[0] = "UP";
10    else
11      directions[0] = "NAH";
12    if(analogRead(JOYSTICK1_VRY) > 790)
13      directions[1] = "LEFT";
14    else if(analogRead(JOYSTICK1_VRY) < 720)
15      directions[1] = "RIGHT";
16    else
17      directions[1] = "NAH";
18    if(analogRead(JOYSTICK1_SW) == 0)
19      directions[2] = "PRESSED";
20    else
21      directions[2] = "NAH";
22    //Joystick 2 >>
23    if(analogRead(JOYSTICK2_VRX) > 790)
24      directions[3] = "DOWN";
25    else if(analogRead(JOYSTICK2_VRX) < 720)
26      directions[3] = "UP";
27    else
28      directions[3] = "NAH";
29    if(analogRead(JOYSTICK2_VRY) > 790)
30      directions[4] = "LEFT";
31    else if(analogRead(JOYSTICK2_VRY) < 720)
32      directions[4] = "RIGHT";
33    else
34      directions[4] = "NAH";
35
36    if(analogRead(JOYSTICK2_SW) == 0)
37      directions[5] = "PRESSED";
38    else
39      directions[5] = "NAH";
```

```
40      if(directions[2] == "PRESSED" && directions[5] == "
           ↪ PRESSED")
41      {
42        mode = 0;
43        delay(300);
44      }
45    }
46
47    bool checkReset()
48    {
49      if(digitalRead(RESET)==HIGH)
50      {
51        reset();
52        return true;
53      }
54      return false;
55    }
```

The getReadings() function starts by checking if the operator has pressed the reset button. This was coded inside another function called checkReset(), in the code above. If the reset button is pressed then the checkReset() would return true and a return statement would be activated to return from the getReadings() function to the caller, but if the reset wasn't pressed the code would continue. The next part of the conditions would handle the input readings from joystick1 (the left one in Figure 7.6).

The joystick module is a simple two potentiometer circuits in which each axis is controlled by a potentiometer rotation, and the rotation is produced by the stick. Each potentiometer circuit sends an analog value to the Arduino board, this value has a range of (0 to 1024) as a digital value after conversion from analogue. The joystick's resting position ((0,0) X,Y position was the same state for both joysticks shown in Figure 7.6) delivers value of 755 for both axis. The Y axis value would be delivered through a module pin named VY, the X-axis value

would be delivered from VX and if the stick was pressed it would trigger a logic high input value from the SW pin from the module. The target was to decide when the joystick was moving up, down, to the left, to the right and pressed; so if the retrieved value from the VY pin was less than 755 then the stick was moved upwards but if the stick was moved downwards then the value would be bigger than 755, given that strategy the stick would deliver a very sensitive control to the motors as slightest touch would change the potentiometer value and the motors would start to move with only the slightest touch on the stick, so a gap was set between either up and down to increase the required value from 755 to either from zero to <720 (Up) or from maximum to >790 (Down).

An array was made to save all acquired input data in the getReadings() function, the array was named directions. The VRX, VRY and SW pins from both joysticks modules were wired into the Arduino pins and in the code the Arduino pins numbers were defined as constants in the global scoop, each constant was named by the corresponding joystick and pin name, for example JOYSTICK1_VRX, etc. Back to the code, the first condition was made to check on the value delivered by the VRX of joystick1, see line 6 of the code above for more details, if the retrieved value was bigger than 790 then the stick was moved down and the "DOWN" string would be saved in the direction array, see line 7 in the code above, if the value wasn't bigger, then another check was made to decide if the same value was less than 720, if the check was true then that would indicate that the stick was moved upwards then the value "UP" would be saved in the directions array, see code lines 8, 9 in the code above. If the value of JOYSTICK1_VRX wasn't bigger than 790 or less than 720, then the stick was in the normal state and didn't move, in that case the value "NAH" would be saved in the directions array to indicate that there was no movement on joystick1, see code lines 10, 11 in the code above for more details. Other conditions were made for joystick1 VRY which retrieves the value for the Y-axis, applying same conditions as in the VRX case with "LEFT or "RIGHT" or "NAH" string saved. This was saved in the next element in the directions array(directions[1]), see code lines from 12 to 17 in the code above. The value of VRX retrieved from the stick movement is the up and down direction

and the value of VRY retrieved from the stick movement is the left and right direction. That was because both modules were fixed after 90 degrees rotation on the wooden board. The last pin to control that was made for joystick1 was the SW pin (the press button). Two more conditions were made for the SW pin. When the stick was pressed the retrieved value was 0, so if the reading was zero the string value "PRESSED" was saved in the next element in the directions array (directions[2]), if the value wasn't zero then "NAH" string value would be saved, for more details see code lines from 18 to 21 for more details.

Code lines from 23 to 39 in the code above were created to implement control for joystick2 with the same concept as the implementation in joystick1. The last portion of code in the getReadings() function was made to reset the mode selection only and not the whole application. In order to do this, a condition was made to check if both joysticks were pressed then the mode was set again to 0, see code lines from 40 to 44 for more details.

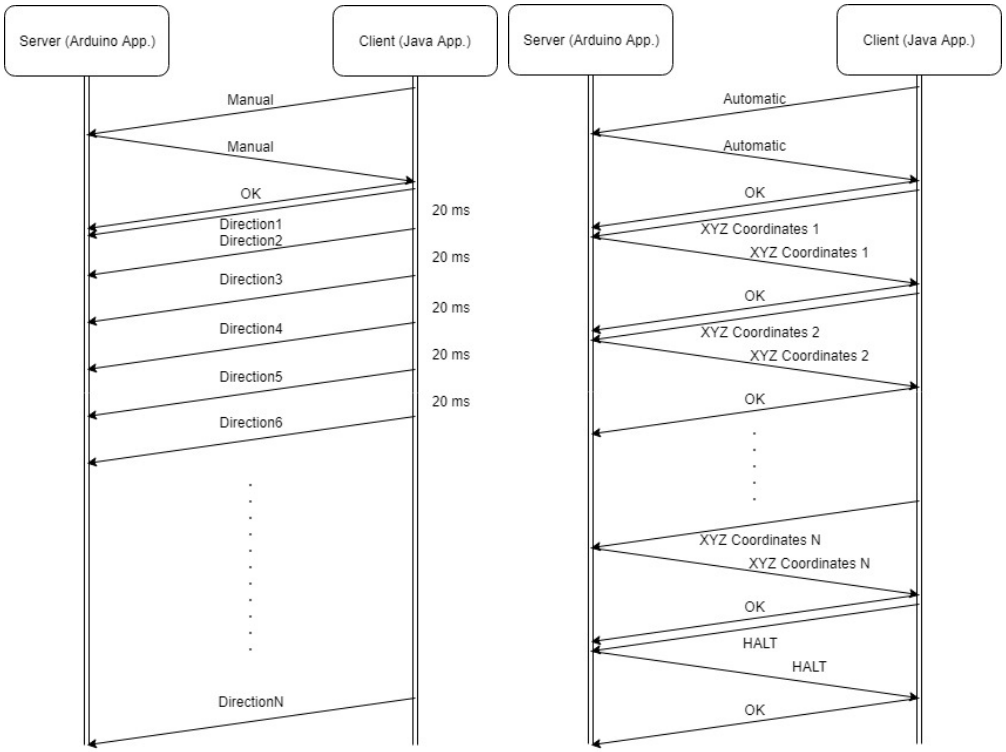
After the getReadings() function finishes, the code continues from where it was called. Now back to the code in the loop function, from line 105 in the code in Appendix C. Conditions were made to check on the selection to set the mode to 1 ("Network" selection) or 2 ("Manual" selection), the selected option would also be printed on the LCD screen.

If the mode was set to 1, the code would enter another condition in which the network connection would be initiated for one time only, see code from line 124 in Appendix C. First the getReadings() function will be called again to check if the reset button was pressed then a condition would be applied to check if Wi-Fi module have connected to the network, this condition would be true only in two cases: either the first time for the program to run or the reset() function was called which resets the mode back to 0 as one of its tasks and hence the network gets disconnected.

The next step was to send network status over serial communication then print status if the connection was successful on the LCD. The Server would be initiated using the begin() function which made the server starts listening for clients, see code line 157 to 166 in Appendix C. The next step was to initialize the client

object from WiFiClient class, the server.available() would return data about the client if and only if there was a client connected, if not it would return false. If the client was connected (the client is the Java application), then a message would be printed on the LCD screen on the second row "Client connected", the boolean variable alreadyConnected would be set to true to prevent the code from entering this if condition and restarting the server, also the networkMode string variable would be set to "starting".

Before discussing the client condition code, the nature of the network protocols and some assisting function need explanation.



(a) Manual Network Protocol (b) Automatic Network Protocol

Figure 7.7: Network Protocols

Figure 7.7 above shows the used network protocols. Figure 7.7a shows the communication protocol for manual control from the client side (Java application). In this protocol the client would initiate the communication by sending "Manual" to the server (The Arduino application). The server would confirm the message by sending back the same massage "Manual" to the client. When the client receives

the message back the message was compared again with the last sent message, if it was a match then it was confirmed that the message was received successfully. If the received message wasn't a match, then the client would send the same message again and wait for the server to send a match, this process would repeat until the received message at the client was a match. The client would then send "OK" to inform the server of the successful communication and to prepare the server to start receiving the manual directions right away. The client then sends another message with the clicked direction from the Java application, see Figure 7.3 in the Java application section for more details. The client would then send a new direction every 20 ms. In this phase the client application doesn't require the server to resend the message, if a message went missing from the client to the server it's not important because in 20 ms there would be an update to the direction. This control would take over the Arduino until the client disconnects. If disconnection occurs, the Arduino application would automatically reset and would start again giving the operator the same choices as before (Manual or Network).

The automatic protocol starts the same as the manual protocol but this time the client would send "Automatic" to the server, see Figure 7.7b for more details. Again, the server would replay with the same message. the client would check the message, if the received message matches the last sent one, then the client would send "OK". The difference in this protocol was that all data in the cutting file is needed, no single line could be missed, a single missing line would put the burr in the wrong position. So, every single line would need reception confirmation from the server. The client (Java application) would read the cutting file and start sending line by line to the server, each line contains an update for the XYZ position of the burr, for more details about the cutting file please refer to section 6.3.1. The client would send each line to the server and wait for the same line to be received from the server, if not the line would be sent again until a match was achieved. The client would send "OK" after each correct match and then sends the next line. This process would go on until the end of the cutting file. At the end of the file the word "Halt" was written by the D-Flow application. When the

"HALT" message was received at the server side, the Arduino application resets.

```
1  String getNetworkReading()
2  {
3    int counter = 0;
4    char temp[60];
5    String reading = "";
6    while(!client.available())
7    {
8      delay(5);
9    }
10   String networkMsg = client.readStringUntil('\n');
11   while(reading != "OK")
12   {
13     if(reading != "OK" && !reading.equals("\0") &&
         ↪ reading != 0)
14       networkMsg = reading;
15     networkMsg.toCharArray(temp,60);
16     client.write(temp);
17     while(!client.available())
18     {
19       delay(5);
20     }
21     reading = client.readStringUntil('\n');
22     reading.trim();
23   }
24   return networkMsg;
25 }
```

The code above implements the getNetworkReadings() function which implemented the message confirmation part in the network protocol. The function starts by declaring some variables then waits for the client to send data to be read by the server. In line 10, the next message sent by the client would be

stored in networkMsg String. The code would enter the while loop as the condition would be true for the first time at least, as the reading string was set to "" (empty value). The if condition also would be false for the same reason. In line 15, the received string value in networkMsg would be converted into character array and the result was stored in the temp variable. In line 16, the value stored in temp variable would be sent back to the client. The next loop was to hold the code until the message from the client was received either as "OK" or the correct message. In line 21, the message would be read and stored but this time in reading variable. The first iteration of the while loop in line 11 is finished, now back to the check. If the message received from the client was "OK" then the loop is finished and line 24 would return the first received message (the message before the "OK" at line 10) to the caller. If the value stored in the reading variable wasn't "OK", that means the first received message wasn't correct or the message was corrupted as it was sent back to the client and in both cases the client would resend the correct message. In the case that the if condition in line 13 was true, the value in the reading variable would be stored in networkMsg. The rest of the process would continue as before and the loop would repeat until the "OK" was received from the client, then in line 24, the correct message would be returned to the caller.

```
1     if (client)
2     {
3       if(networkMode == "starting")
4       {
5         networkMode = getNetworkReading();
6       }
7       networkMode.trim();
8       if(networkMode.equals("Manual"))
9       {
10        String in = client.readString();
11        in.trim();
12        // echo the bytes back to the client:
```

```
13          if(in == "up")
14          {
15            directions[4] = "RIGHT";
16          }
17          else if(in == "down")
18          {
19            directions[4] = "LEFT";
20          }
21          else if(in == "left")
22          {
23            directions[1] = "LEFT";
24          }
25          else if(in == "right")
26          {
27            directions[1] = "RIGHT";
28          }
29          else if(in == "in")
30          {
31            directions[4] = "LEFT";
32          }
33          else if(in == "out")
34          {
35            directions[4] = "RIGHT";
36          }
37          else if(in == "idle")
38          {
39            directions[1] = "Nah";
40            directions[0] = "Nah";
41            directions[4] = "Nah";
42          }
43          runIt();
```

```
44        }
45        else if(networkMode == "Automatic")
46        {
47          char in[60];
48          String data = getNetworkReading();
49          data.toCharArray(in,60);
50          data.trim();
51          if(data == "HALT")
52          {
53            reset();
54            Serial.println("HALTTED!!!");
55          }
56          else
57          {
58            int i = 0;
59            for(i; i < data.length();i++)
60            {
61              if(data.substring(i,i+1) == ";")
62              {
63                x = data.substring(0,i).toFloat();
64                break;
65              }
66            }
67            i +=1;
68            int b = i;
69            for(i; i < data.length();i++)
70            {
71              if(data.substring(i,i+1) == ";")
72              {
73                y = data.substring(b,i).toFloat();
74                break;
```

```
75              }
76            }
77            i +=1;
78            z = data.substring(i,data.length()).toFloat();
79            stepperM.moveTo(cmToTurns(x*100));
80            stepperZ1.moveTo(cmToTurns(z*100));
81            stepperZ2.moveTo(cmToTurns(z*100));
82            stepperY1.moveTo(cmToTurns(y*-100));
83            stepperY2.moveTo(cmToTurns(y*-100));
84            while(stepperM.distanceToGo() != 0 || stepperZ1
                 ↪ .distanceToGo() != 0 || stepperY1.
                 ↪ distanceToGo() != 0)
85            {
86              delay(COMMON_DELAY);
87              stepperM.run();
88              stepperZ1.run();
89              stepperZ2.run();
90              stepperY1.run();
91              stepperY2.run();
92            }
93            delay(10);
94          }
95        }
96      }
97      else if(alreadyConnected)
98      {
99        lcd.setCursor(0, 1);
100       lcd.print("Client␣disConnected");
101       reset();
102     }
103   }
```

The portion of code above represents the client condition which includes the code for network communication whether manual or automatic. The part of code shown above are from code lines 167 to 269 from the application in the Appendix C.

The client condition would be entered only if the client object returns true, that would occur if the connection between the client and server was active. The networkMode string variable was set to "starting" as it wasn't set yet to the correct mode by the client, so the first condition was to set the variable and that was done by the getNetworkReading(), see code line 5 in the code above. This condition would run once before setting the network mode and after calling reset() if happened. If the return value of getNetworkReading() function was "Manual", then the code would continue in the next condition in line 10 in the code above. If the "Manual" condition was entered, see code line 5 in the code above, then the first part of the manual protocol has already been achieved inside the getNetworkReading() function and the client has sent the "OK", see the manual protocol in Figure 7.7a. The next step in implementing the protocol was to start receiving the directions from the client. In line 10 in the code above, the received direction from the client was stored in the variable "in". A series of conditions have been applied to set the received movement in the right slot in the directions array, see code lines from 13 to 42 in the code above for more details. In line 43, the function runIt() was called, this function was responsible for running all the motors using the stored directions in the directions array. The stored direction would drive the motors until the clients update the array which means each run takes 20ms. The runIt() function will be discussed in detail later in this section. The previous paragraph was about the "Manual" protocol, now if the received mode at line 5 in the code above was "Automatic" then the code would fail the condition on line 8 and enter the condition on line 45 in the code above. The mode starts by receiving next line from the client's cutting file. If the received message was "HALT" the system would reset immediately by calling the reset() function. If the message wasn't "HALT" then the string would be split into three

sub-strings using the ';' character as the end of number identifier, for example:

$$0.0076619630143402; 0.0037538121627129; 0.011794779546133$$

The String above represents a typical received message. The code lines from 58 to 66 would search for the first ';' digit number and extract a sub-string from 0 to the digit before that number, in the giving example the if condition in line 61 would be true when i = 18, then x = 0.0076619630143402 as a double value. The same process would be repeated for the y and z values.

The XYZ value was now ready in terms of the distance to be moved by the stepper motors but this needs to be converted to the number of turns. This was accomplished by suing the simple function cmToTurns(float x), see the code lines 399 to 402 in Appendix C for more details.

The cmToTurns(float x) function takes the distance in cm as input then returns the value multiplied by 2000, as discussed before each 1mm can be achieved by 200 turns so it takes 2000 for 1cm. The values were converted to cm as the received values were in meters. The returned value was then sent to moveTo function which was one of the "AccelStepper.h" library functions that when the run() function get called the motor would move only the number of turns sent to the moveTo function. Same process was applied for the five motors, see code lines above from 79 to 83. The while loop was made to make sure that the run() function would continue to be called as long as there were still distances to run in the three axis XYZ. As the loop function loops forever, this process would be repeated and each time a new line would be received from the client and the burr would move to the new position. This was repeated until the "HALT" command was received then the application would reset.

The reset() function simply resets all needed variables like the mode back to 0. The function also disconnects from the Wi-Fi. The last task was to drive the motors back to the local origin of the CNC, see code lines from 431 to 474 in Appendix C.

```
1   else //Manual
2   {
```

```
3     getReadings();
4     runIt();
5     delay(COMMON_DELAY);
6   }
```

If the network option wasn't selected in the first place by the operator then the code would only enter the code inside the else scope above, see code lines from 270 to 276 in the application in Appendix C. When the operator selects the Manual option as discussed before that means that the CNC would be driven from the joysticks, so simply in line 3 in the code above the getReadings() function would update the directions array from the joysticks then the method runIt() would be called to drive the motors.

The runIt() function was used in both cases of control whether from the joysticks or the network manual control. This function task went through the directions array present at the time the function was called. This array was filled with the required directions. The function would then configure the direction of the motors and that was accomplished by setting the speed by negative value as the "AccelStepper" library changes the motor direction with negative speed value, for example if the speed was set to 3000 to drive the carriage forward then resetting the speed to -3000 will drive the carriage backwards, see code line from 351 to 397 in Appendix C for more details.

7.4 Discussion

It was thought that building a surgical robot with wireless capability was a suitable add on to the full system which would reduce cable runs within the operating theatre. In reality the wireless methods introduced a problem. If the cutting resolution was increased, that would make cutting file much larger and would require more time for the cutting file to be sent. For example, the working resolution used was 1mm but if the resolution was increased to 0.05mm that would double the file size as the burr would move double the movements. This in turn could make a problem with working with Arduino Wi-Fi Shield. The Shield occasionally af-

ter running for a long time goes into a mode called silent mode which makes the shield disabled which requires restarting of the operation. In testing this occurred when the cutting file had more than 9000 lines. It was estimated that the cutting files in 1 *mm* increments for a typical uni-arthroplasty cut worked out to contain 2500 lines maximum. So, for the current development project the Arduino Wi-Fi shield was adequate. However, for more complex applications this limitation may prove problematic and hence the Arduino Wi-Fi Shield is not recommended to be part of any network implementation with data transfer protocols in future studies.

7.5 Conclusion

- Building a wireless system function was successful which enabled remote control of the device.
- Two network protocols was created to support different control mechanisms.
- Different control mechanisms were provided for the operator including manual control (from the robot its self or from the Java application on the PC) or Automatic control.

Chapter 8

Results of Saw Bone Automated Cutting

8.1 Introduction

This chapter will present the results of the surgical process developed when it is used to burr a set of tibia and femur artificial bones. The robotic system performs the cutting phase of the surgery only. The surgeon plans the implant position and orientation on the knee, then performs an incision. Then the surgeon/operator starts the knee registration process (to inform the robotic system where should it cut the implant shape). Then the robotic system performs the cutting procedure. The last step would be the surgeon handling the fixation and cementing phase.

Figure 8.1: Sawbone fixed and Ready for Registration

The experiments in this chapter tests only the cutting capability of the system. In this experiment the CNC machine will be used to cut nine tibial plateau's as flat surfaces followed by nine curved femoral surfaces. Their shapes will be established using a 3D scanner. For the tibia their shapes were compared using Geomagic software to the original sawbone shape, a flat surface and a set of previously cut sawbones from both the Mako and Blue Belt robotic systems. For the femur the shapes were compared by eye to the planned shape.

The cutting procedure starts by fixing the sawbone and the CNC machine in the centre of the theatre, see Figure 8.1 above, then running the Motive software and the D-Flow application. This step enables the tracking of the clusters. The next step was the registration process for the bone using the Blunt and Sharp probes and the foot switch. These were used to register three points on the surface of the femur or tibia to be cut. These three points were used to create a local coordinate frame in the bone surface relative to which the bone would be cut.

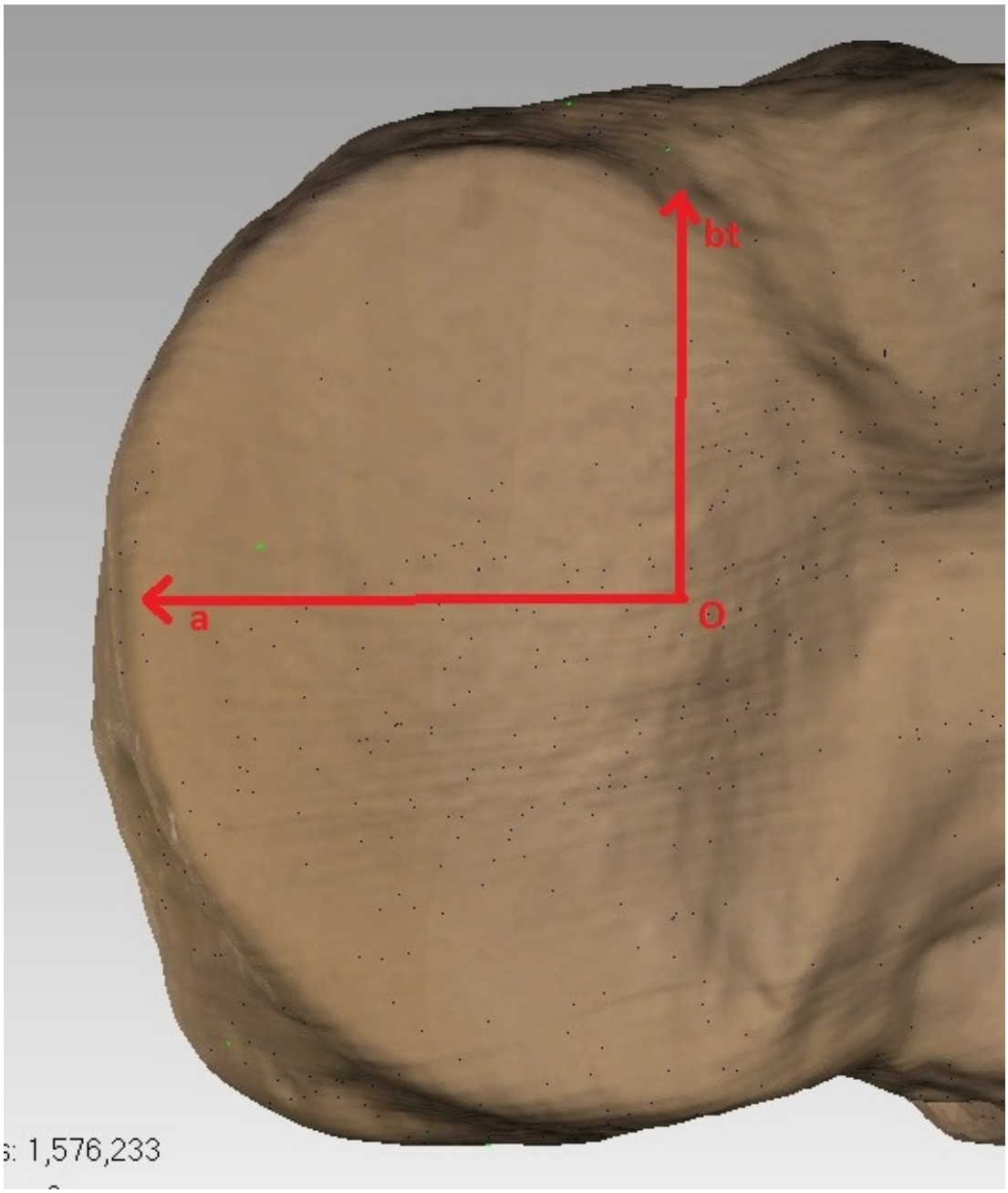


Figure 8.2: Tibia Reference Sawbone

The registration was achieved by registering the origin point first, which is represented by letter O in Figure 8.2 above, the photo is for the medial condyle of a tibia sawbone. Then another point was registered along the $\overrightarrow{a}$ vector, then another point along the $\overrightarrow{bt}$ vector. Then D-Flow application takes over and calculates $\overrightarrow{c}$ vector then true $\overrightarrow{b}$, for more information about the rotation and translation process refer to Section 6.3.1. After that registration process finishes

by registering the burr tip location.

The D-Flow application saves the cutting file in the local coordinate system of the bone after the last point has been registered. The CNC machine was in ready mode and connected to the Wi-Fi, via the Java application running on the same computer as the D-Flow software. The Java application controlled the CNC machine and after the Automatic option was set, the application transferred the required CNC movement coordinates to the CNC machine and the burring took place. This procedure was repeated for every sawbone (Femur and Tibia).

The Working System specifications of the computer running all used software in the procedure (Motive, D-Flow and the Java network application) were:

- **Processor:** Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz.
- **Graphics Card:** NVIDIA GeForce GTX 850M.
- **RAM:** 16.0 GB DDR3.
- **System:** Windows 8.1 64-bit.
- **D-Flow:** 3.18.0.

After the procedures were performed the cut sawbones were scanned using a "Matter and Form" laser scanner. The scanner software exported the scanned data file as a ".obj" file. The exported file was imported into another software package called "Geomagic". Geomagic software had many important features such as 3D comparison and object volume computing. First the 3D comparison feature was used to compare between the cut sawbones and an uncut sawbone (reference sawbone). The Surface cut in each sawbone was extracted. A 2D plane was inserted into the cut surface of the sawbone and the two surfaces was compared in order to determine the surface roughness.

8.2 Tibia Surface cut Analysis

8.2.1 Developed System Tibia Cuts Analysis

This section of the chapter will present an analysis of the bone removed by the cutting process for each bone. The bone removed from the sawbone can be determined by making a three-dimensional comparison between the cut sawbone and an uncut sawbone. This was achieved first by comparing the cut sawbone to another uncut sawbone, then another comparison was made by creating a flat surface in Geomagic software and manually fit it on to the cut. The first comparison should show how much cutting was performed on the bone surface (interior and posterior), while the other comparison was made to show the flatness of the cut on the bone surface.

Figure 8.3 below shows the scan of the uncut tibia(reference bone) for the 3D comparison. The cut sawbones were compared to this reference tibia. For the comparison each point in the reference bone digital object gets compared to a point that occupy the same location in the test object which in this case was one of the cut sawbones. If the point is found then it will be displayed as green in the coloured graph, but if the point isn't found then the nearest point will be displayed with a colour that maps the Euclidean distance between both points.

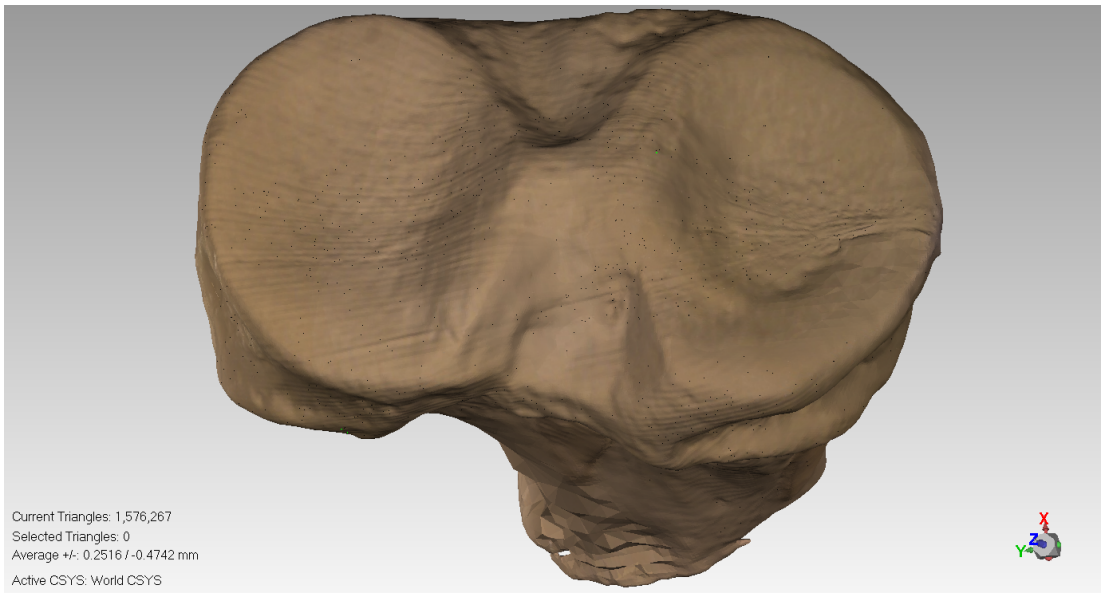


Figure 8.3: Tibia Reference Bone

Figure 8.4 shows the tibia implant and Figure 8.5 shows the cutting path for the burr.

Figure 8.4: Implant

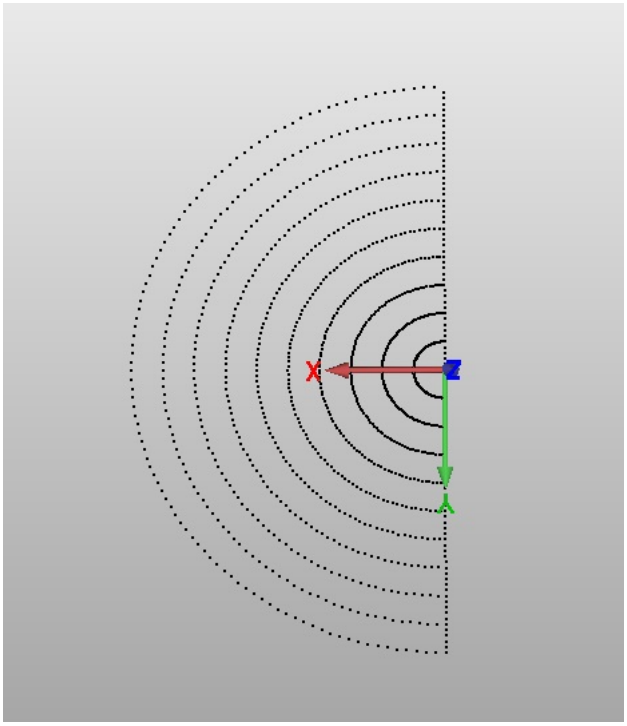
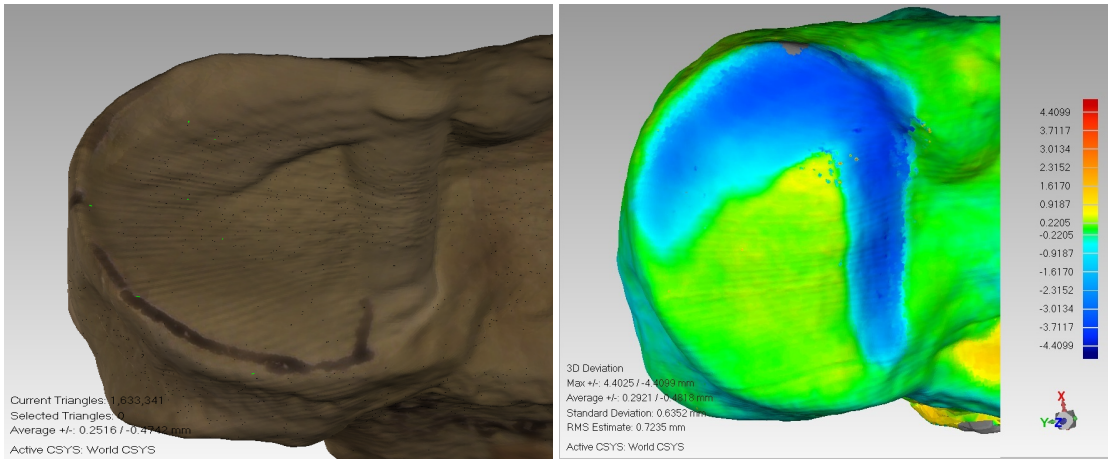


Figure 8.5: Cutting Path Represented as Points

Figure 8.6a shows the first tibia sawbone in the set and Figure 8.6b shows the 3D comparison coloured figure as a result of the comparison between the cut tibia sawbone 1 in Figure 8.6a and the uncut reference tibia bone in Figure 8.3.



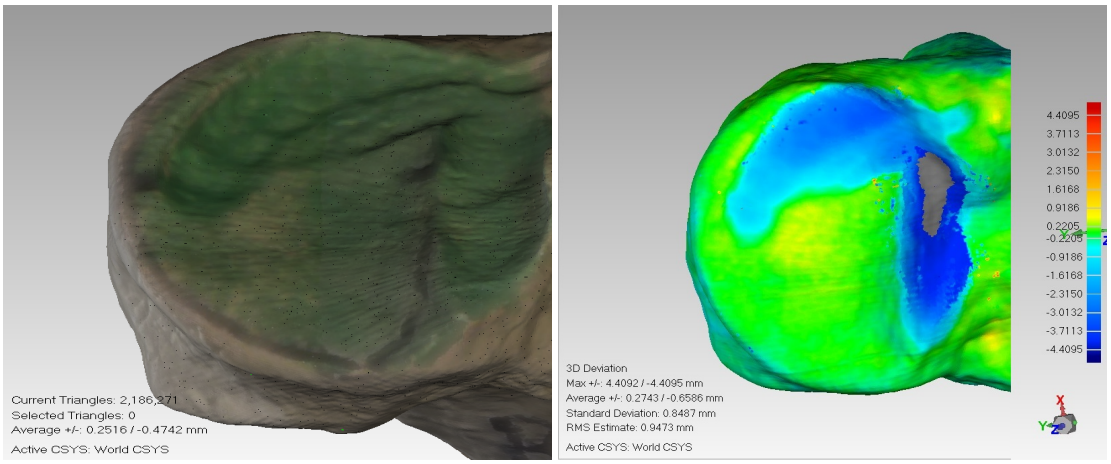
(a) Tibia Bone 1 - A Typical Tibial Bone After the Surface Has Been Burred

(b) Tibia 1 - Comparison of The Cut Surface of The Bone With a Reference Bone

Figure 8.6: Tibia 1

Figure 8.7: Tibia 1 Cut Bone View by Camera

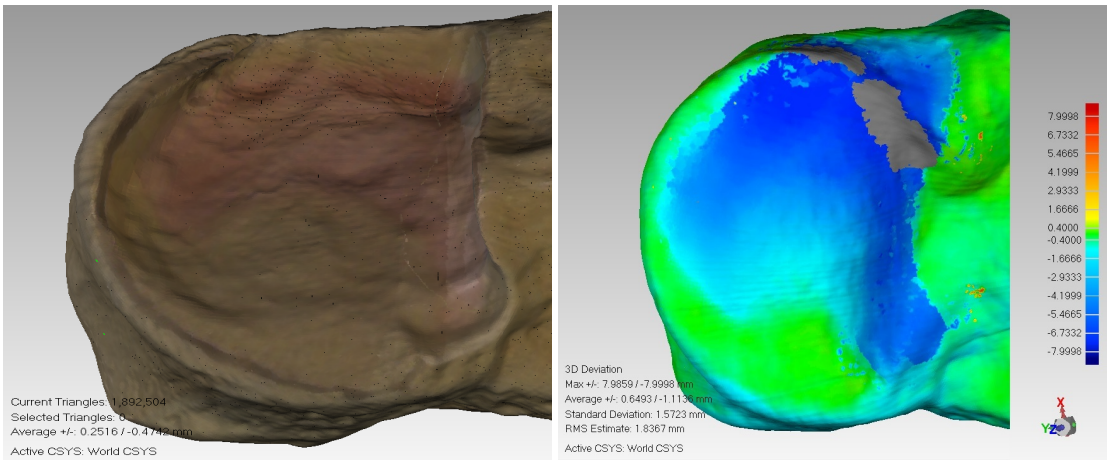
Figure 8.6b shows green colour for the areas that are uncut, as the green colour marks identical points on both surfaces. There is a blue area in the shape of an upside down "L". This is the cut area of the bone. The blue colour indicates a shallow cut which is only present in the "L" shaped area of the tibial plateau. There was clearly an orientation problem with the cut. The cutting plane of the burr was tilted anteriorly and medially which caused the cut path to remove material anteriorly and medially on the condyle but not posteriorly and laterally on the condyle. The cut was also far too shallow.



(a) Tibia Bone 2 - Cut View (b) Tibia 2 3D Comparison

Figure 8.8: Tibia 2

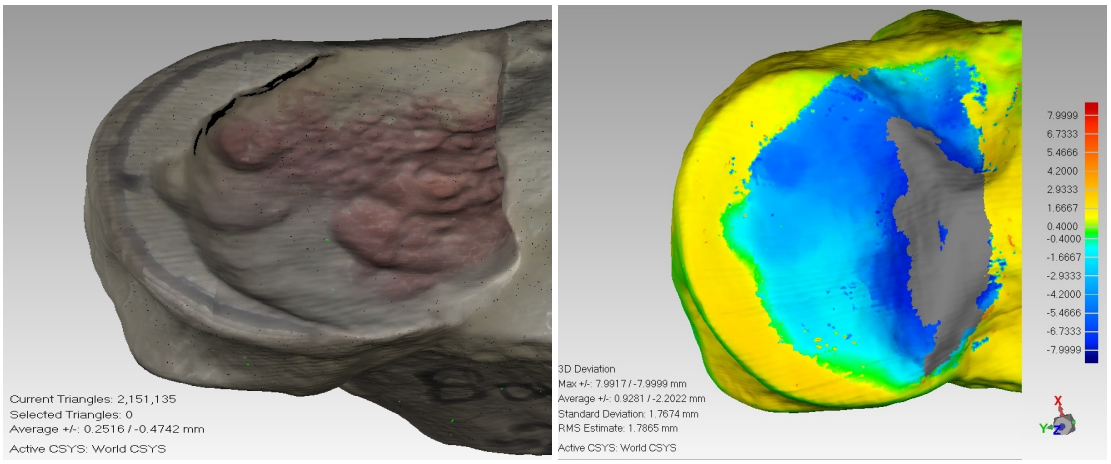
A second tibia was cut. In this bone (tibia 2) the same orientation problem occurred, as shown in Figure 8.8b almost same shape was curved as in tibia 1 but slightly shifted to the right. At this point it was recognized that the registration process was inadequate, the registration process was performed using the Blunt probe and its tip was hemispherical not pointed. Sometimes during the registration process it was noticed that the side of the hemisphere rather than the tip of the probe was in contact with the bone. The hemisphere diameter was about 2 to 3 mm which was sufficient for the cutting plane to be shallow and rotated as seen in the previous two bones. The Sharp probe was used to replace the Blunt probe in the registration process for the rest of the tibia set. The sharp probe had a specific and unambiguous tip.



(a) Tibia Bone 3 - Cut View (b) Tibia 3 3D Comparison

Figure 8.9: Tibia 3

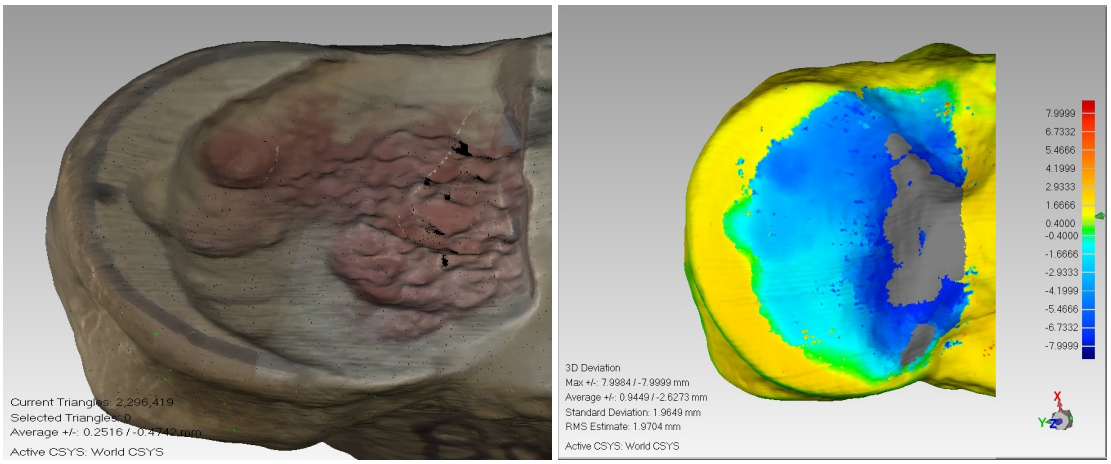
Using the sharp probe, the results improved. Figure 8.9a shows the cut shape and Figure 8.9b shows the cut comparison. The cut was a little bit shifted anteriorly and medially but of a better depth.



(a) Tibia Bone 4 - Cut View (b) Tibia 4 3D Comparison

Figure 8.10: Tibia 4

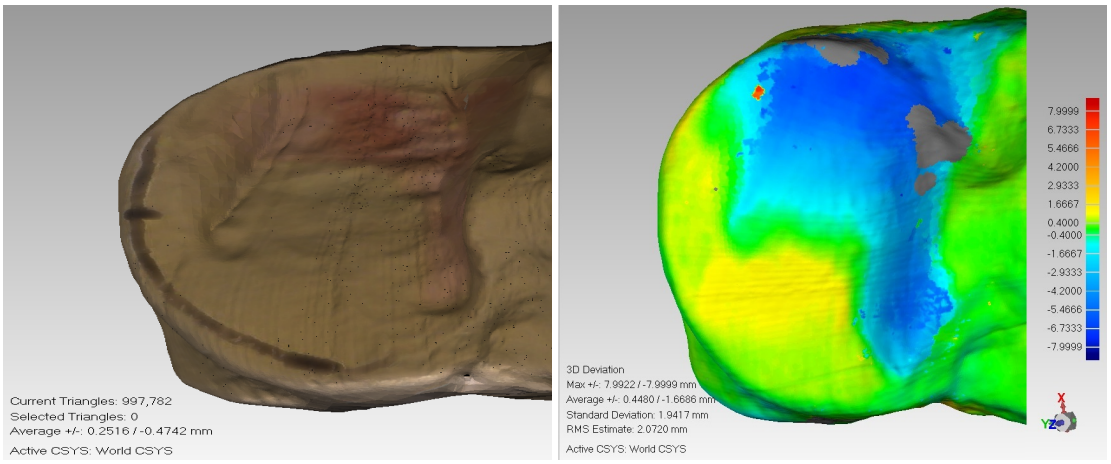
For bone 4 (Figure 8.10). The cut was shifted medially but a full depth was achieved.



(a) Tibia Bone 5 - Cut View (b) Tibia 5 3D Comparison

Figure 8.11: Tibia 5

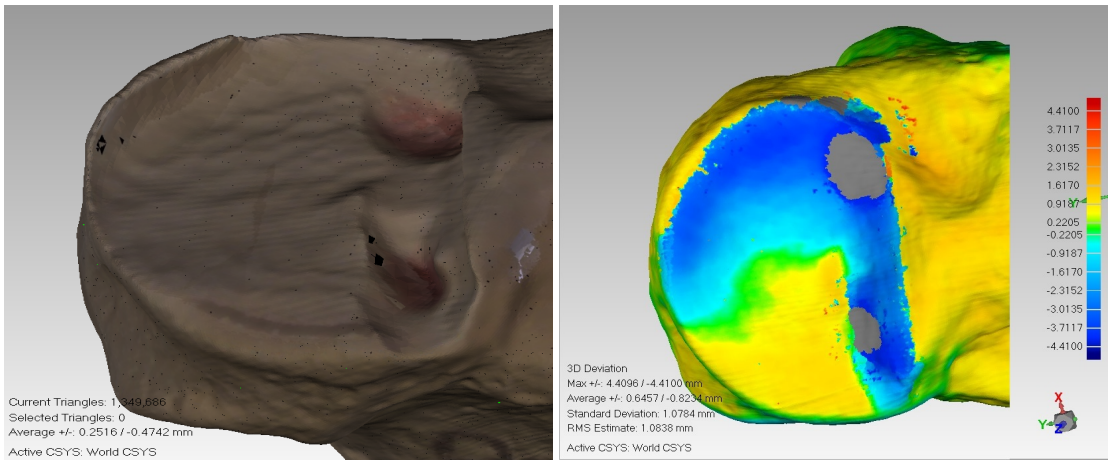
Figure 8.11a shows the cut shape and Figure 8.11b shows the cut comparison. The cut was also shifted medially. It was recognized that this matched the diameter of the burr and so a lateral shift in the cutting path equivalent to the radius of the burr was implemented.



(a) Tibia Bone 6 - Cut View (b) Tibia 6 3D Comparison

Figure 8.12: Tibia 6

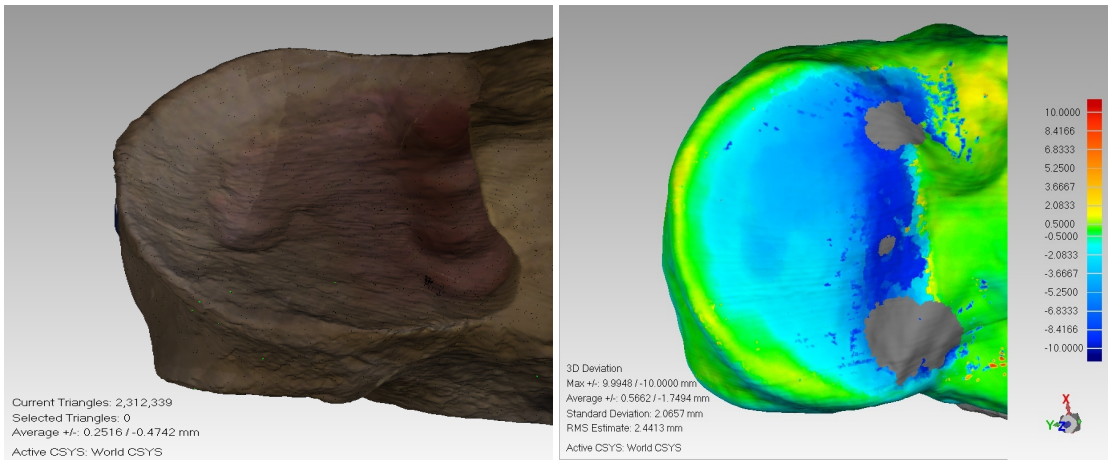
Figure 8.12a shows the cut shape and Figure 8.12b shows the cut comparison. The cut was shifted up.



(a) Tibia Bone 7 - Cut View (b) Tibia 7 3D Comparison

Figure 8.13: Tibia 7

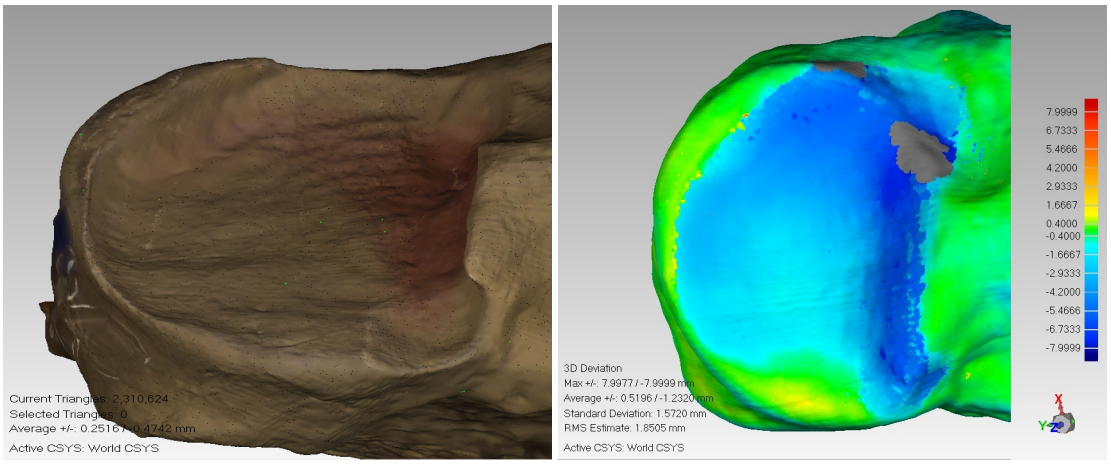
Figure 8.13b for bone 7 shows a slight orientation problem, most of the cut was performed perfectly but the posterior-lateral area wasn't all cut. The cutting pattern had also been modified to include the two peg holes for the implant which can clearly be seen in Figure 8.13a.



(a) Tibia Bone 8 - Cut View (b) Tibia 8 3D Comparison

Figure 8.14: Tibia 8

Tibia 8 (Figure 8.14) is one of the best tibia cuts performed by the system. The cut was shifted a little bit to the medial part but otherwise was correct.



(a) Tibia Bone 9 - Cut View (b) Tibia 9 3D Comparison

Figure 8.15: Tibia 9

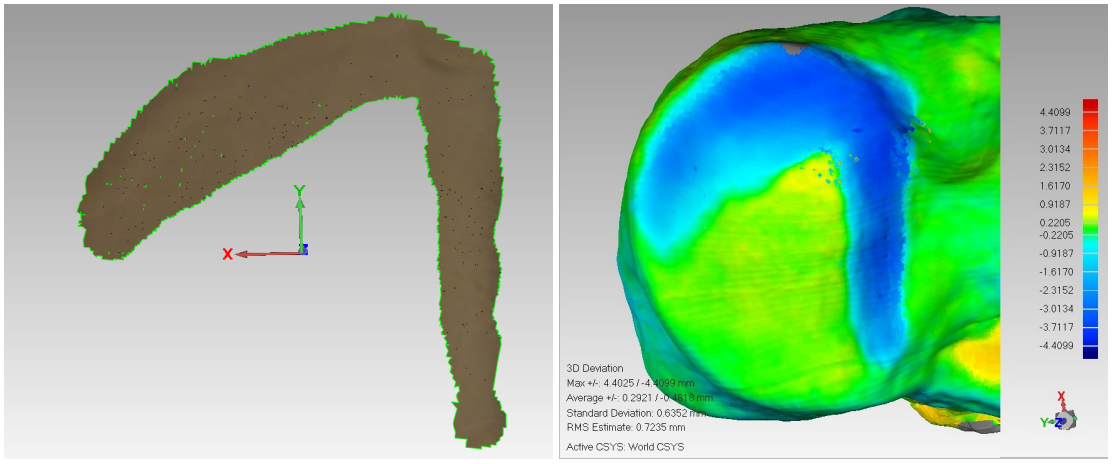
This was repeated for tibia 9 (Figure 8.15) which was also one of the best cuts in the tibia set. The cut is slightly tilted from the upper side to the anterior side. Having adopted the sharp probe and accounted for the burr diameter if proved possible to achieve the required cut in the tibial plateau using the three point registration. Despite achieving a good outcome in tibial cuts 8 and 9, it was recognized that the use of only three points to align and orientate the cut and was very limited and might lead to implant misplacement and in the future consideration should be given in using a greater number of these anchor points. The inadequacy of this method can be seen more clearly in the femoral cuts later in this chapter.

8.3 Tibia Surface Roughness

8.3.1 Developed System Tibia Cuts Fitting Analysis

The following images represent data for the nine tibial bones cut by the system. For each bone the flat cut surface was extracted from the bone image. This cut view was then compared to a flat plane and the result of the 3D comparison between the cut and the inserted plane shows the roughness of the cut surface. Each 3D comparison has a scale on the right side for the colour schema used.

This was automatically set by the Geomagic software and could not be specified by the user (except the maximum and minimum boundaries) so colour schema and the distances they reflect change from image to image. Figure 8.16a shows the cut surface extracted from the image for tibia 1 (Figure 8.16b). Figure 8.17 shows the flat plane and Figure 8.18 shows the difference between the cut surface and the plane.



(a) Tibia Bone 1 - Cut Only View (b) Tibia 1 3D Comparison

Figure 8.16: Tibia 1

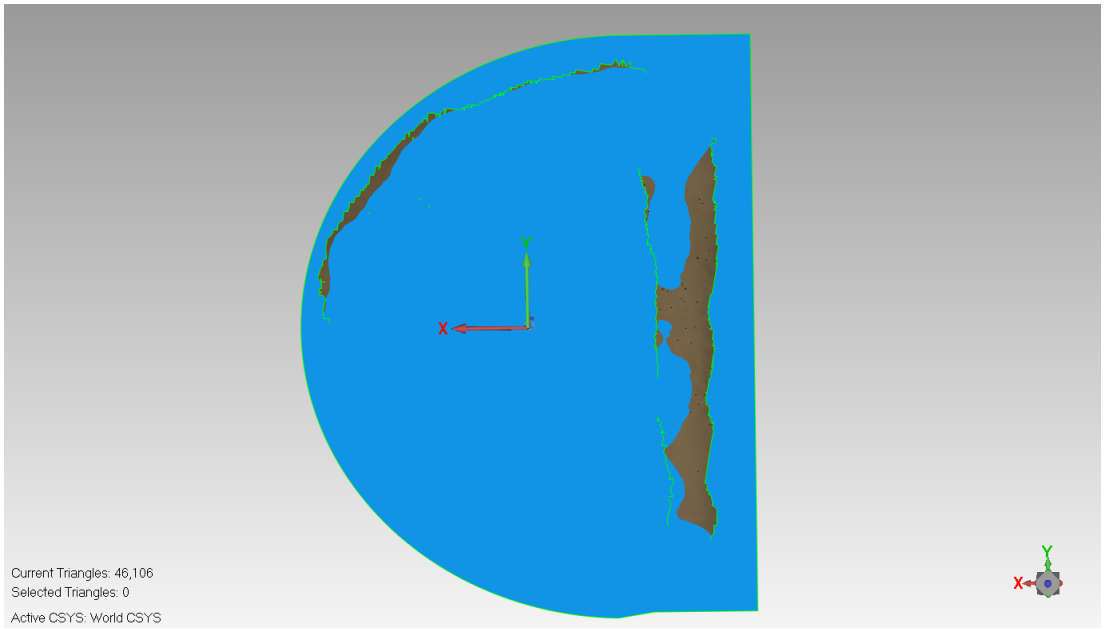


Figure 8.17: Tibia Bone 1 - Cut with Inserted Plane View

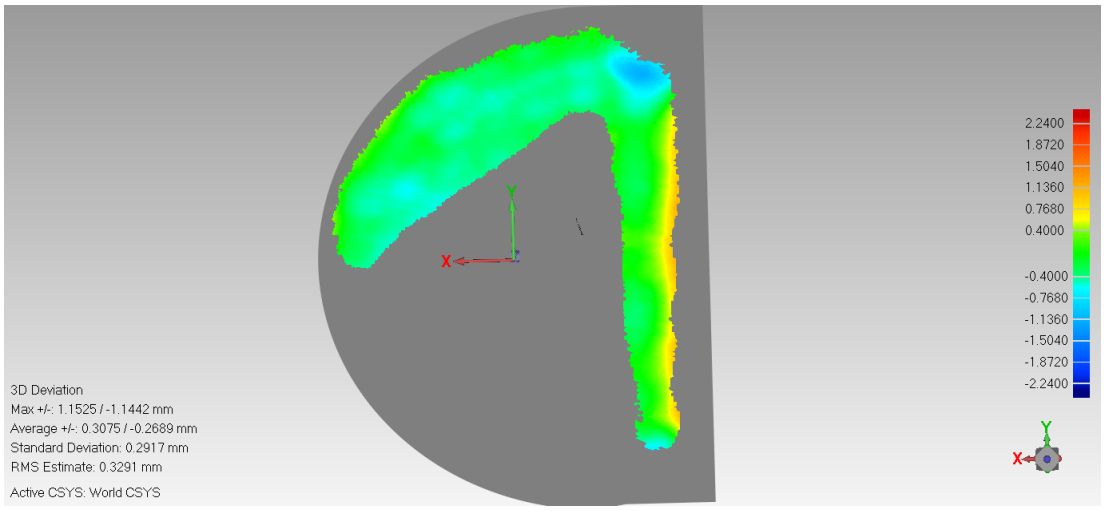
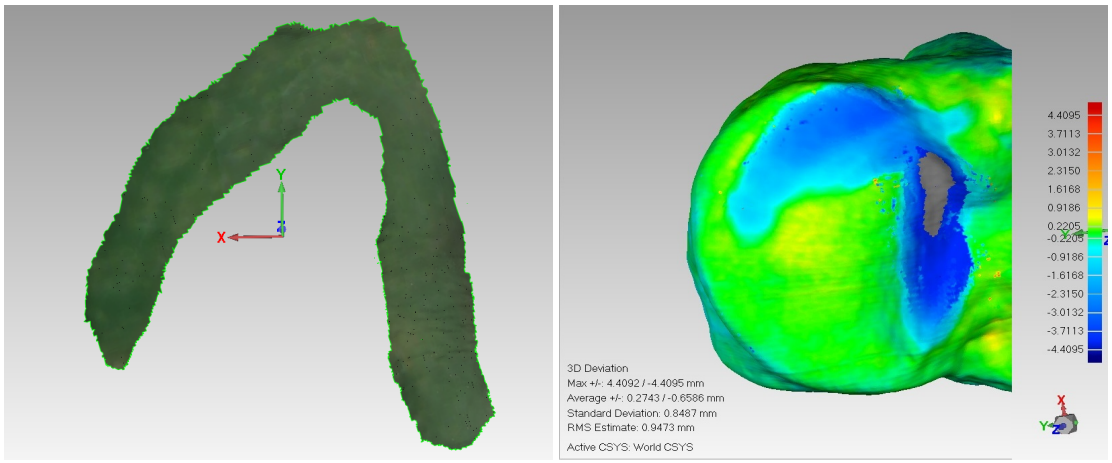


Figure 8.18: Tibia Bone 1 - 3D Comparison View

In this cut as mentioned previously the orientation wasn't accurate, so the burr moved away from bone at places. These areas were left out of the comparison and are coloured as Gray. Figure 8.18 shows green, yellow and light blue, each colour represents an error range on the scale on the right side of the image. The errors were in the range of less than -0.76 mm for light blue areas, range of $\pm$ 0.4 mm for green areas and range of less than 0.76 mm for yellow areas.

Figure 8.19 shows the area extracted for the cut surface for tibia 2.



(a) Tibia Bone 2 - Cut Only View (b) Tibia 2 3D Comparison

Figure 8.19: Tibia 2

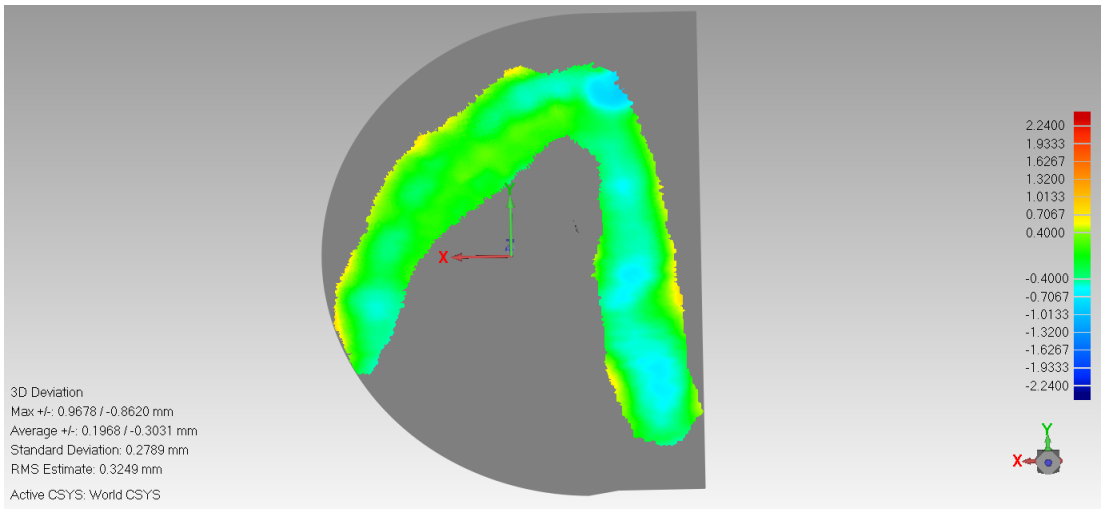
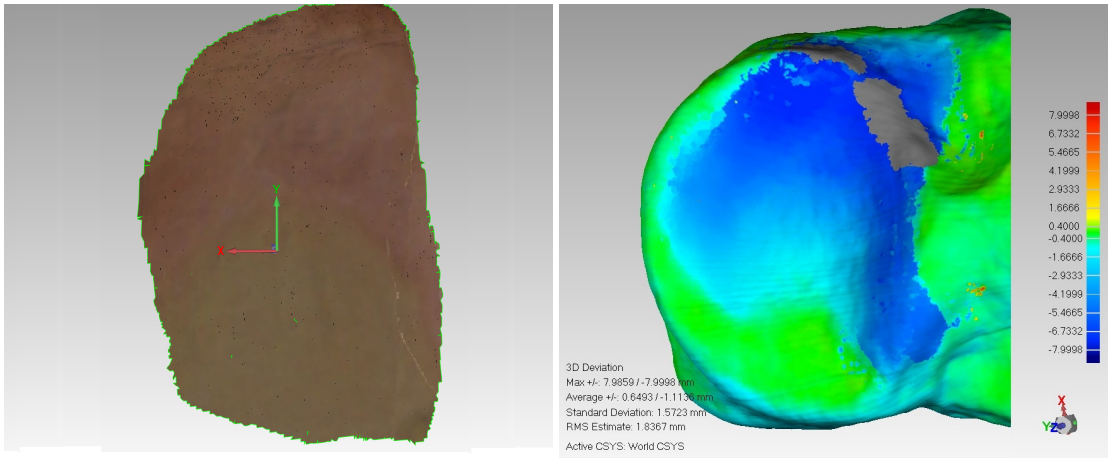


Figure 8.20: Tibia Bone 2 - 3D Comparison View

The comparison for tibia 2 (Figure 8.20) shows green as the main colour which on the scale is an error range of $\pm$ 0.4 mm. There is some light blue with error range of less than -0.7 mm.

Figure 8.21 shows the extracted cut area for tibia 3. This bone was not tilted and so the cut area extends across the whole lateral surface of the sawbone.



(a) Tibia Bone 3 - Cut Only View (b) Tibia 3 3D Comparison

Figure 8.21: Tibia 3

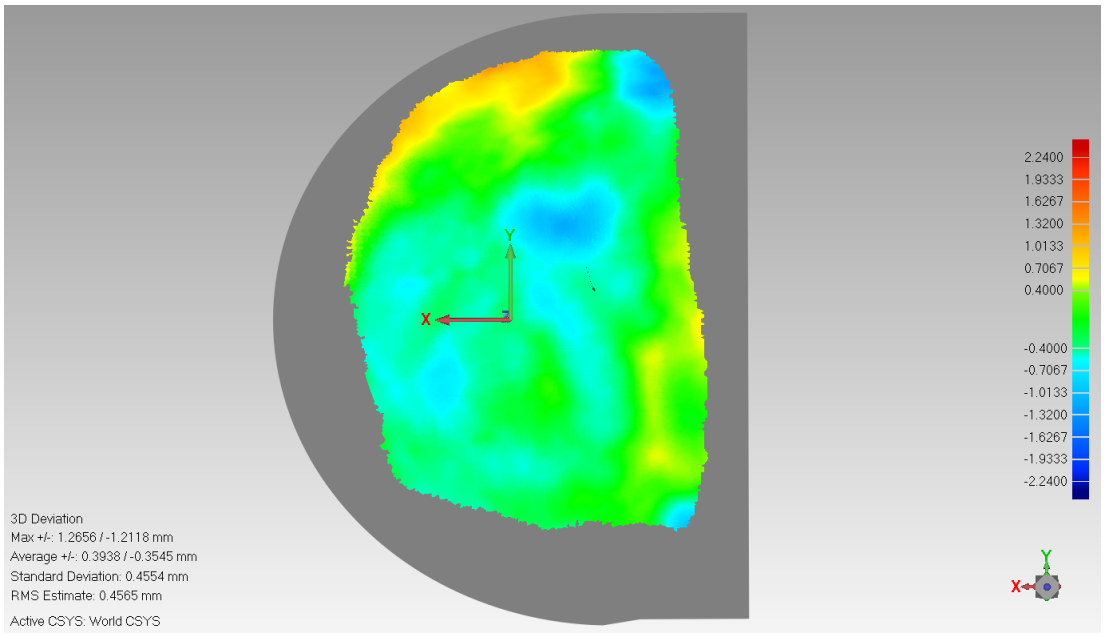
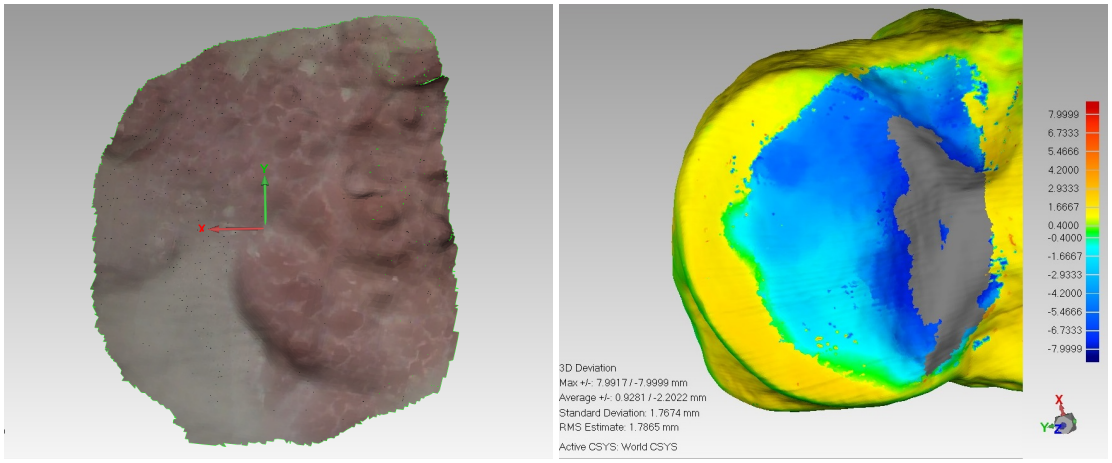


Figure 8.22: Tibia Bone 3 - 3D Comparison View

For the comparison (Figure 8.22) green is dominant with error range of $\pm 0.4mm$. While some areas show light blue and canary colours with error range of less than -0.7 mm and less than 1 mm.

Figure 8.23 shows the extracted area for tibia 4.



(a) Tibia Bone 4 - Cut Only View (b) Tibia 4 3D Comparison

Figure 8.23: Tibia 4

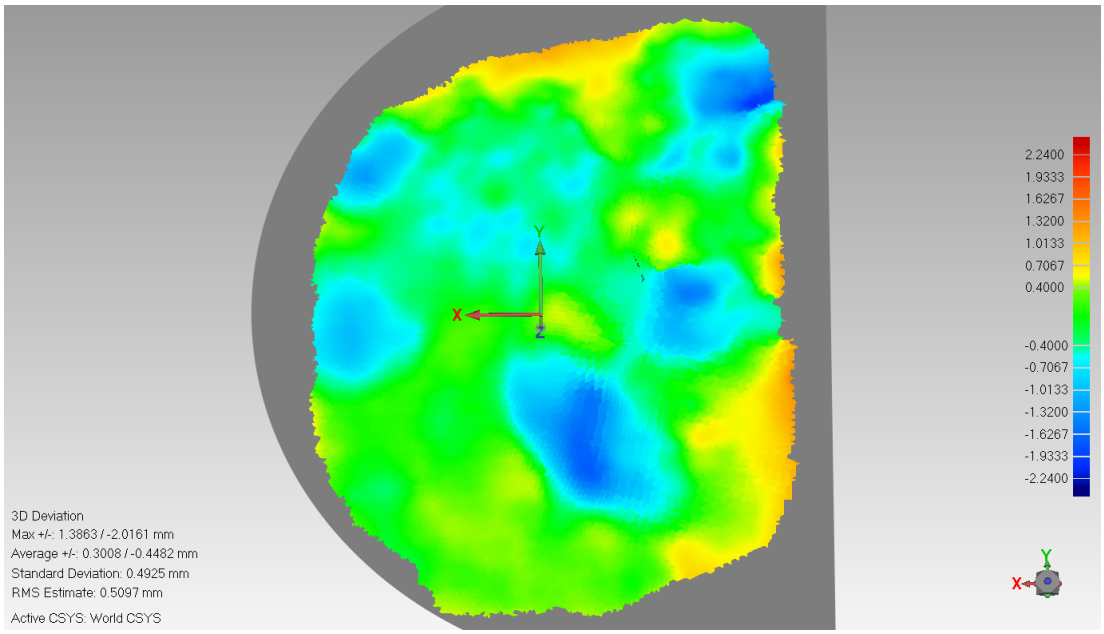
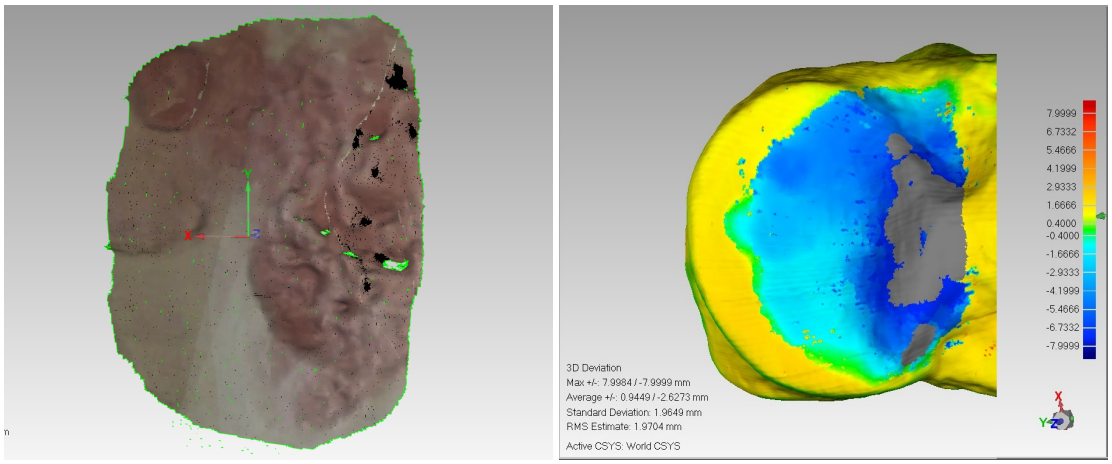


Figure 8.24: Tibia Bone 4 - 3D Comparison View

In the comparison (Figure 8.24) green is again dominant error of $\pm 0.4mm$, some areas showed light blue with error less than -0.7 mm and small areas near the edges were powder blue with error range of less than -1.3 mm. Other parts show yellow colour with error of less than 0.7 mm on the scale.

Figure 8.25 shows the area extracted for tibia 5.



(a) Tibia Bone 5 - Cut Only View (b) Tibia 5 3D Comparison

Figure 8.25: Tibia 5

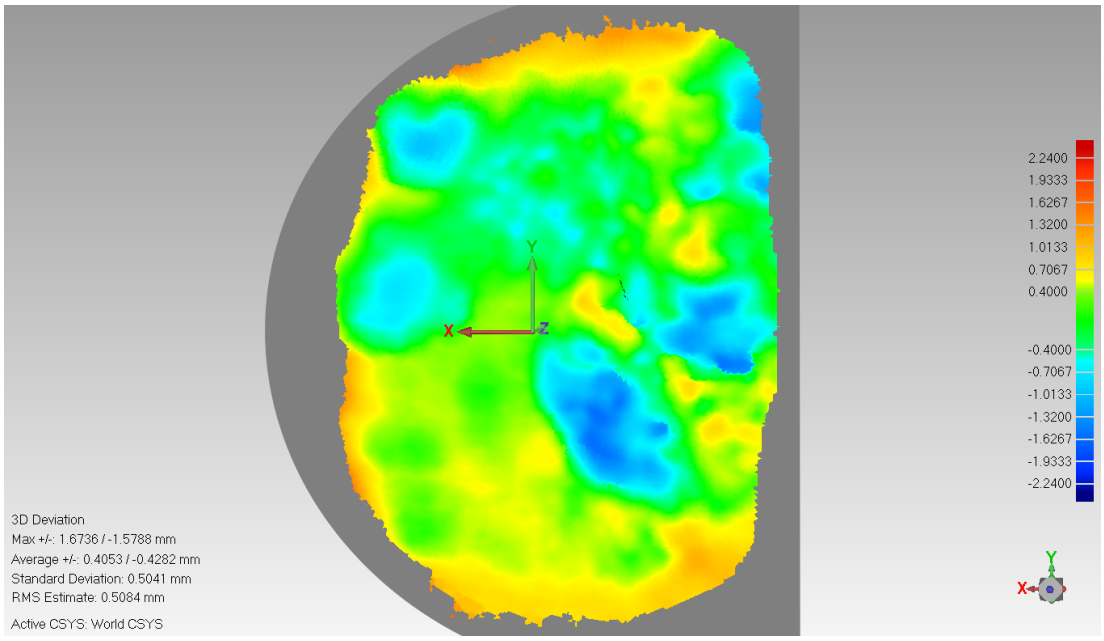
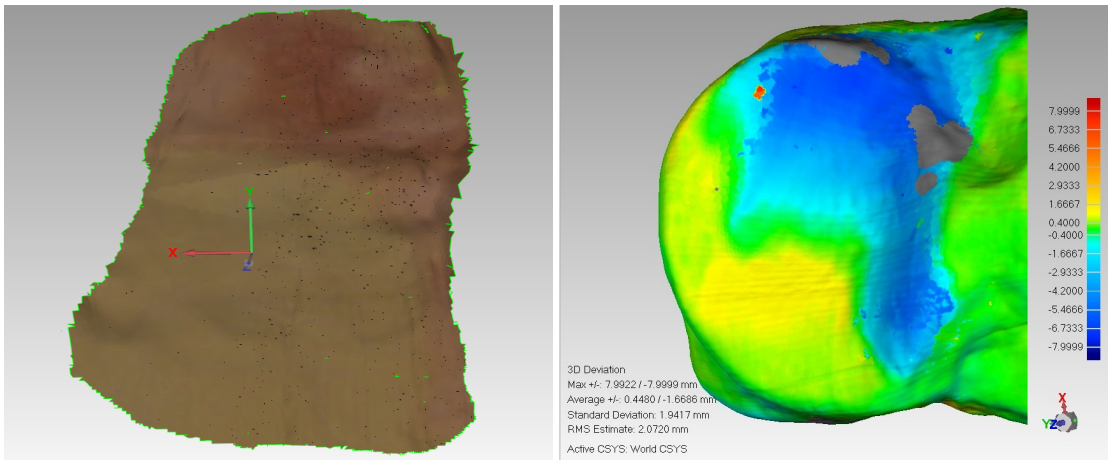


Figure 8.26: Tibia Bone 5 - 3D Comparison View

In the comparison (Figure 8.26) green is again dominant colour with error of $\pm 0.4mm$, some areas showed light blue with error less than -0.7 mm and small areas with powder blue with error range of less than -1.3 mm. Other parts show canary colour with error of less than 1 mm on the scale.

Figure 8.27 shows the extracted area for tibia 6.



(a) Tibia Bone 6 - Cut Only View (b) Tibia 6 3D Comparison

Figure 8.27: Tibia 6

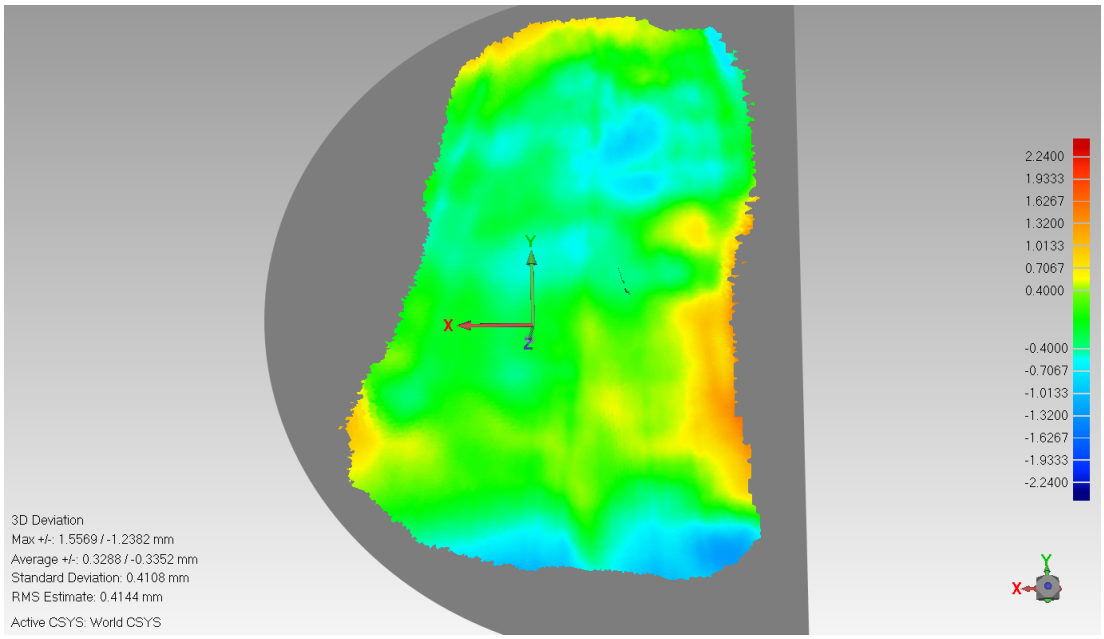
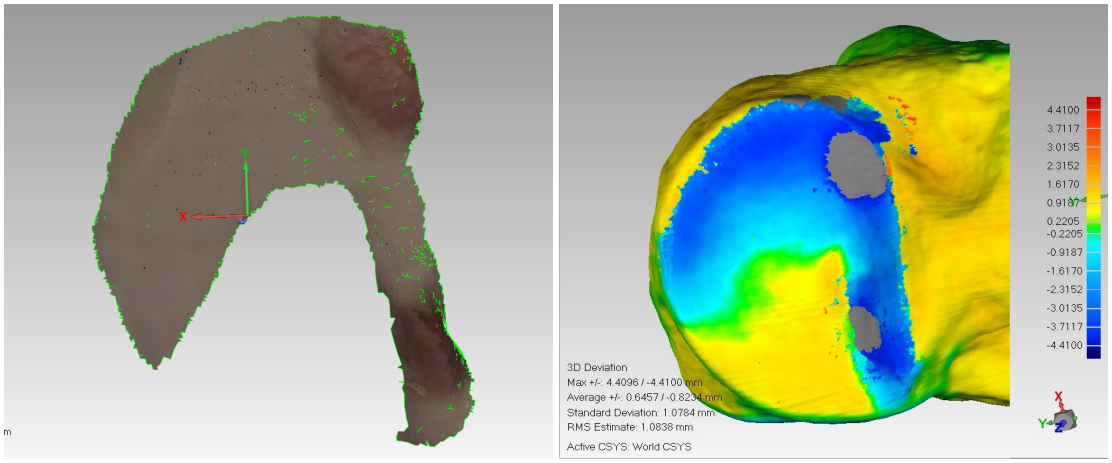


Figure 8.28: Tibia Bone 6 - 3D Comparison View

The comparison (Figure 8.28) again shows most green with error of $\pm 0.4mm$. Some parts show light blue and yellow with error less than -0.7 and less than 1.03 mm respectively.

Figure 8.29 shows the extracted flat surface for tibia 7.



(a) Tibia Bone 7 - Cut Only View (b) Tibia 7 3D Comparison

Figure 8.29: Tibia 7

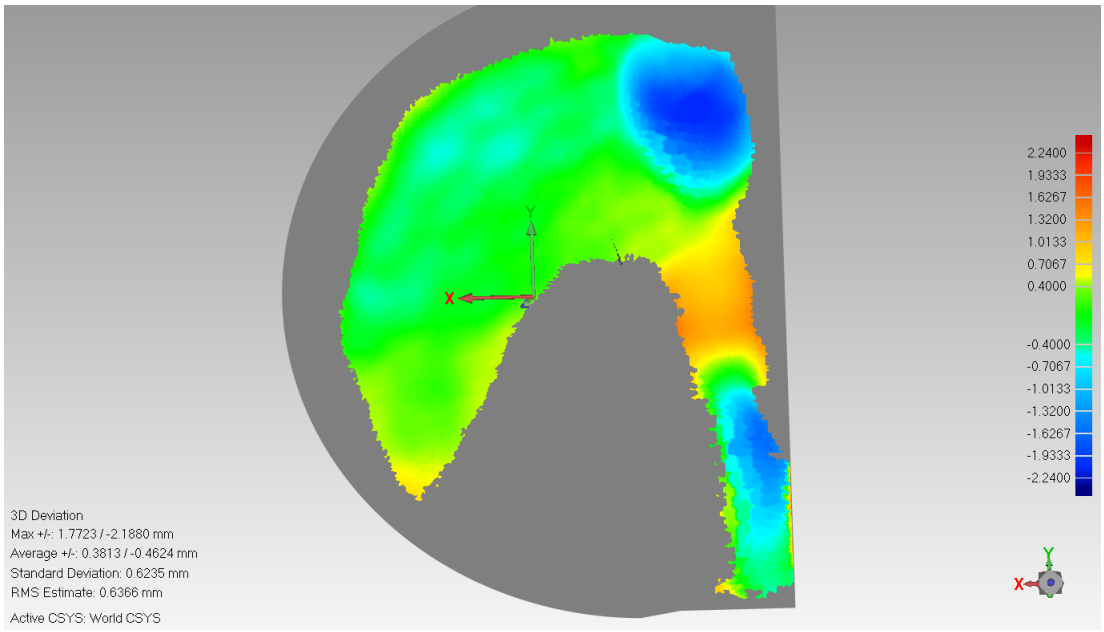
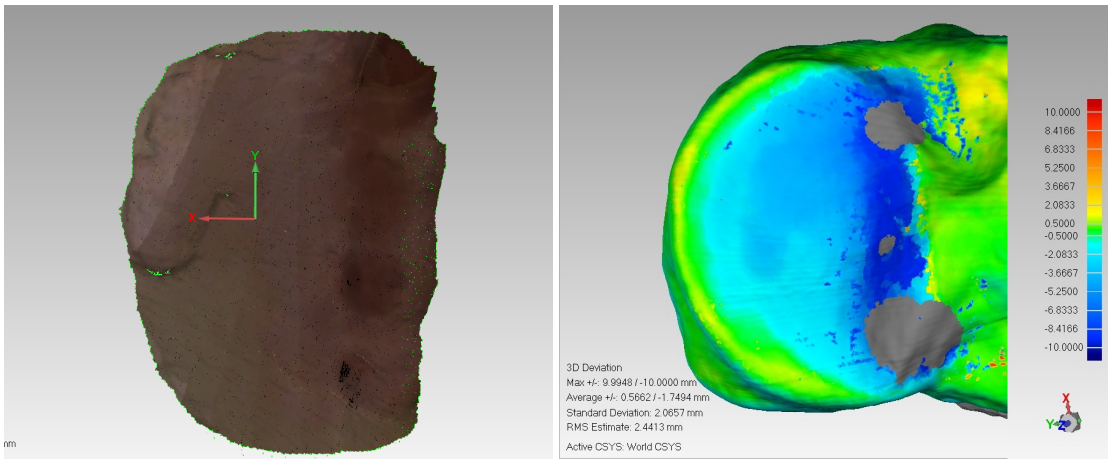


Figure 8.30: Tibia Bone 7 - 3D Comparison View

The comparison (Figure 8.30) again shows green as dominant with error of $\pm 0.4mm$. Some areas were coloured with light blue and blue less than -0.7 mm, less than -1.3 mm respectively and others was coloured with canary with error of less than 1.3 mm.

Figure 8.31 shows the extracted area for tibia 8.



(a) Tibia Bone 8 - Cut Only View (b) Tibia 8 3D Comparison

Figure 8.31: Tibia 8

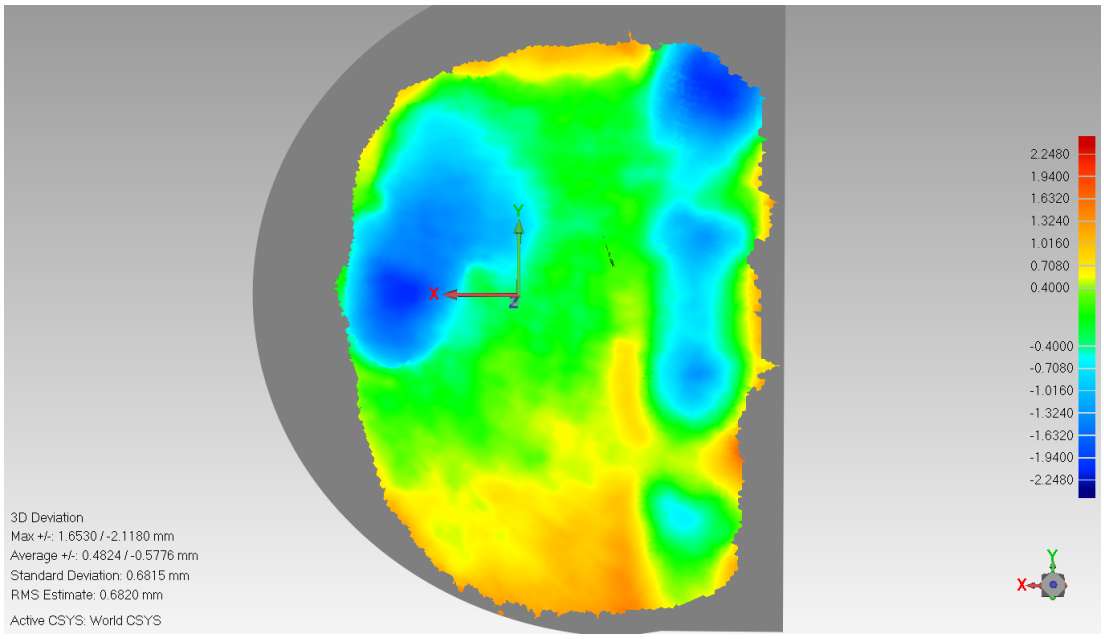
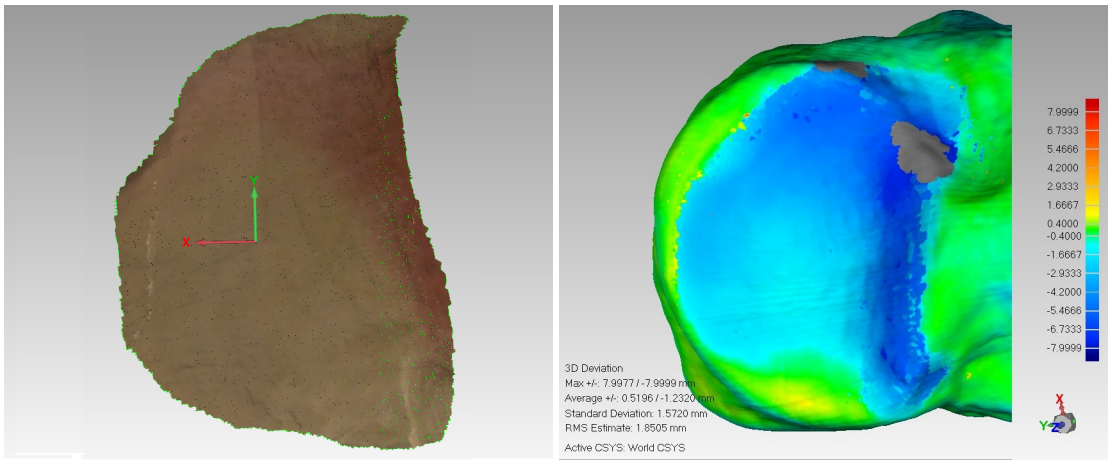


Figure 8.32: Tibia Bone 8 - 3D Comparison View

In the comparison (Figure 8.32) green is again dominant with error of $\pm 0.4mm$, while other parts shows light blue and blue with error less than -0.7 mm and less than -1.6 mm respectively while other parts shows canary with error less than 1.3 mm.

Finally Figure 8.33 shows the extracted area for tibia 9.



(a) Tibia Bone 9 - Cut Only View (b) Tibia 9 3D Comparison

Figure 8.33: Tibia 9

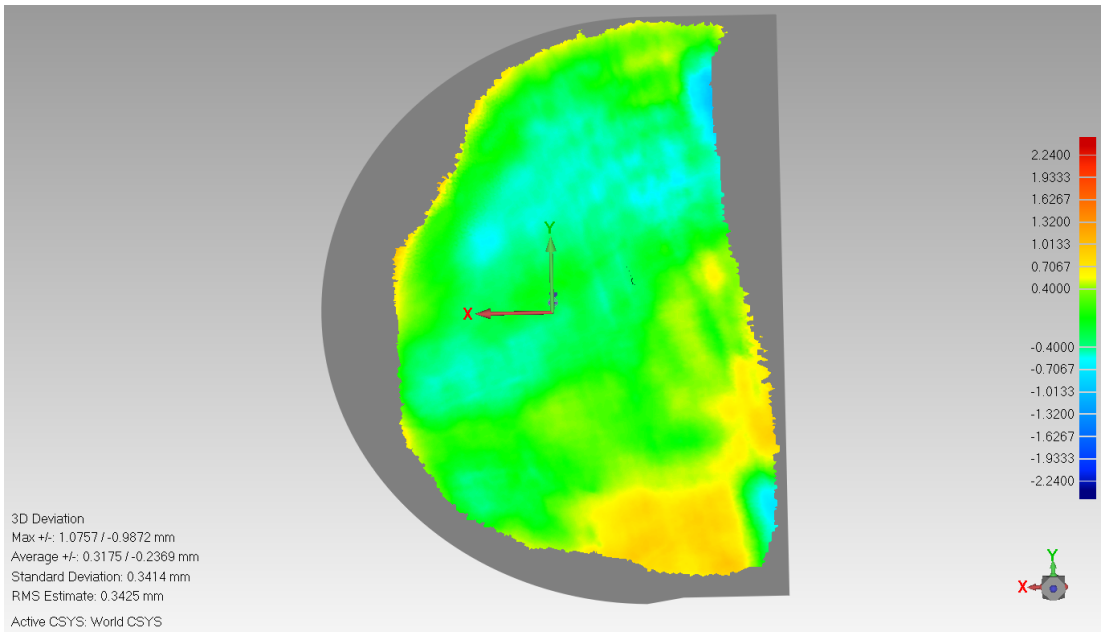


Figure 8.34: Tibia Bone 9 - 3D Comparison View

In the comparison (Figure 8.34) green is again dominant for most of the cut while little parts shows light blue with error less than -0.7 and others a yellow with error less than 0.7 mm.

8.3.2 Mako Tibia Cuts Analysis

This section will include analysis of nine tibia cuts using the Mako Rio system as a comparison. Each bone will have two images for the cut bone, the first for the extracted cut and the second was the 3D comparison by Geomagic. In the Extracted cut images there will be holes for fixation. They were drilled for the Mako implant fixation. These areas were left out of the comparison and as seen as Gray circles.

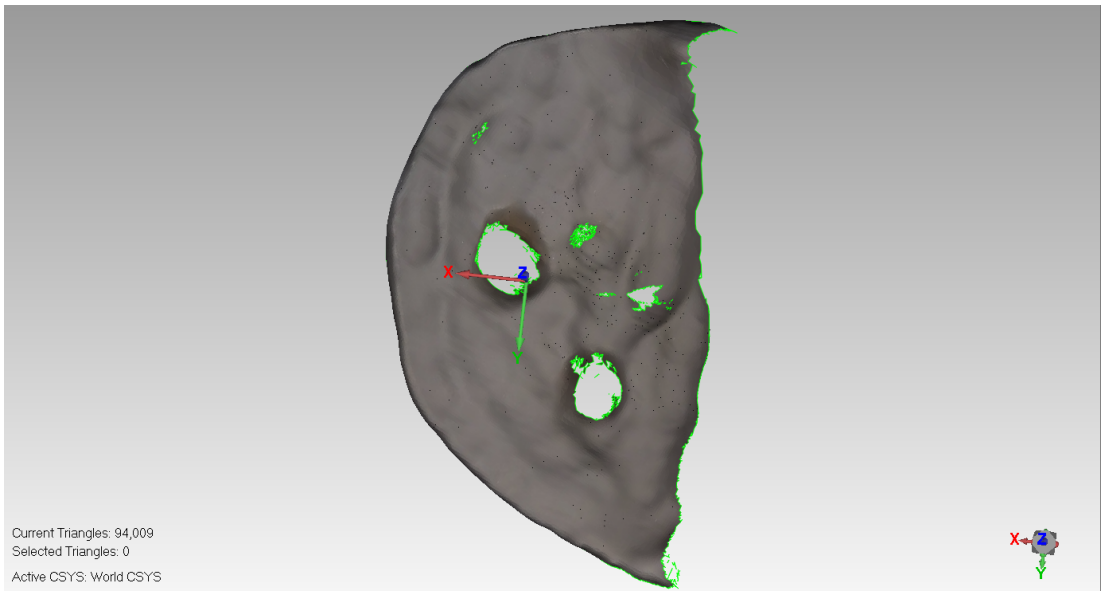


Figure 8.35: Mako Tibia Bone 1 - Cut Only View

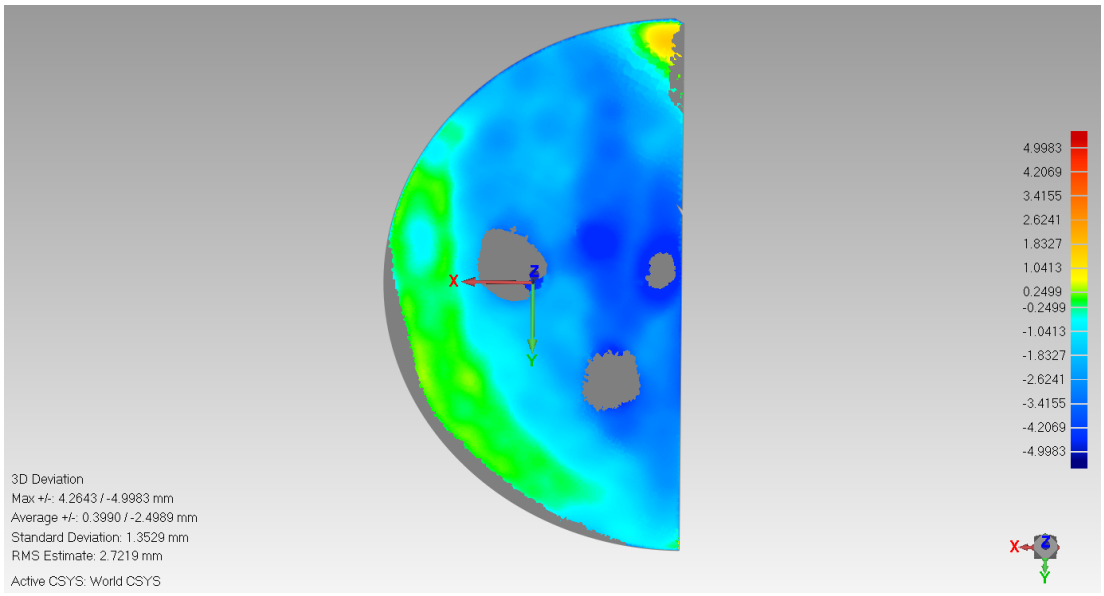


Figure 8.36: Mako Tibia Bone 1 - 3D Comparison View

Figure 8.36 shows some green but dominant colours was light, sky blue and blue with errors of -1.04 mm, -3.4 mm and -4.99 mm. Some parts on the border showed errors larger but these will be neglected as they represent the bone lip remaining on edge. The Mako implant was an inlay.

The rest of the nine sawbone cuts are presented in Appendix D.1

8.3.3 Blue Belt Tibia Cuts Analysis

For further comparison tibial sawbones cut using the Blue Belt Navio system were analysed. This section will present the Blue Belt system cut sawbones and analysis. Again, each bone will have two images for the cut bone, the first for the extracted cut and the second was the 3D comparison by Geomagic.

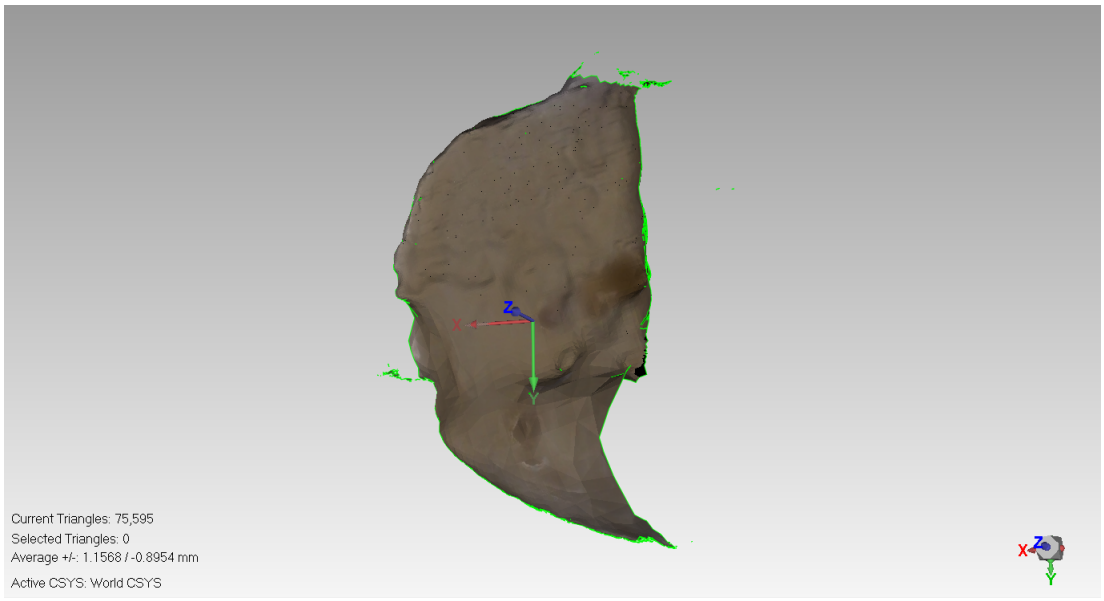


Figure 8.37: Blue Belt Tibia Bone 1 - Cut Only View

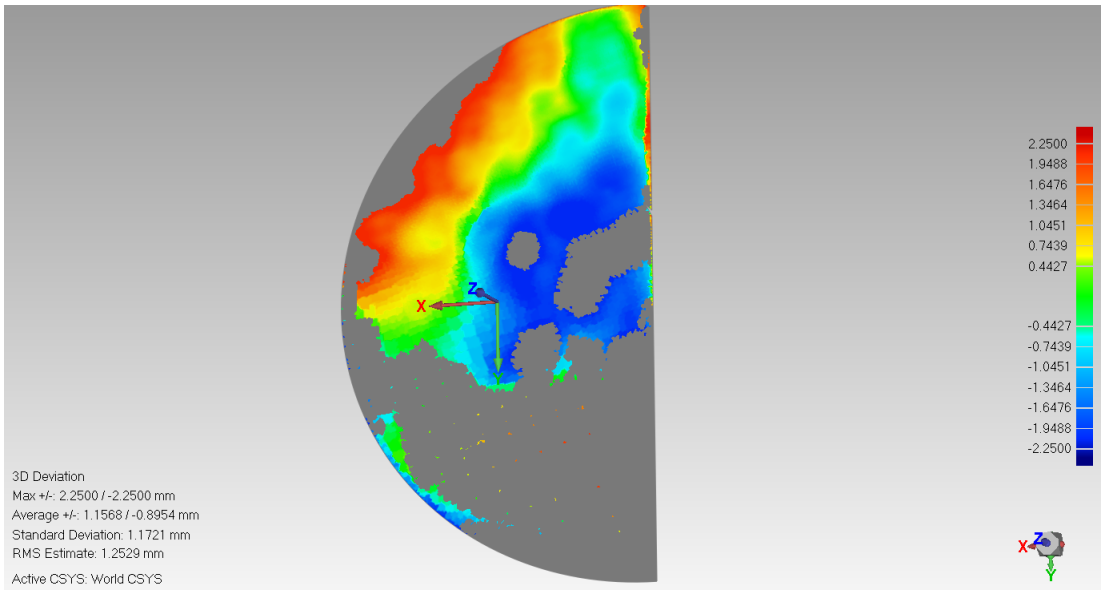


Figure 8.38: Blue Belt Tibia Bone 1 - 3D Comparison View

Figure 8.38 shows blue, powder blue, light blue, green, orange and red colours with errors of -2.25 *mm*, -1.64 *mm*, -0.74 *mm*, 0 for green, 1.64 *mm* and 2.25 *mm* respectively. Further Gray areas are due to excess bone cutting which can be visualised at the lateral part in Figure 8.37.
The rest of the Blue Belt sawbone cuts are presented in Appendix D.2

8.3.4 Summery

The results of the tibial plateau cuts are presented in summary form in Table 8.1 for the developed system, in Table 8.2 for the Mako system and in Table 8.3 for the Blue Belt system. Given that the fitting of the implantation would be dependant on the maximum error across the cut surface, the range of errors is reported.

Table 8.1: Tibia Cut By our designed model Error range

Bone	**Min. Error**	**Max. Error**	**Range**	**Unit**
Bone 1	-0.76	0.76	1.52	*mm*
Bone 2	-0.7	0.4	1.1	*mm*
Bone 3	-0.7	1	1.7	*mm*
Bone 4	-1.3	0.7	2	*mm*
Bone 5	-1.3	1	2.3	*mm*
Bone 6	-0.7	1	1.7	*mm*
Bone 7	-1.3	1.3	2.6	*mm*
Bone 8	-1.6	1.3	2.9	*mm*
Bone 9	-0.7	0.7	1.4	*mm*
Mean			1.9	*mm*
SD			0.55	*mm*
Range			2.9	*mm*

Table 8.2: Tibia Cut By Mako Error range

Bone	**Min. Error**	**Max. Error**	**Range**	**Unit**
Bone 1	-4.99	0	4.99	*mm*
Bone 2	-0.82	1.18	2	*mm*
Bone 3	-1.89	1.18	3.07	*mm*
Bone 4	-1.3	1.6	2.9	*mm*
Bone 5	-1.53	1.53	3.06	*mm*
Bone 6	-2.24	0.46	2.7	*mm*
Bone 7	-1.89	0.46	2.35	*mm*
Bone 8	-1.2	1.61	2.81	*mm*
Bone 9	-1.18	0.82	2	*mm*
Mean			2.87	*mm*
SD			0.84	*mm*
Range			4.99	*mm*

Table 8.3: Tibia Cut By Blue Belt Error range

Bone	**Min. Error**	**Max. Error**	**Range**	**Unit**
Bone 1	-2.25	2.25	4.5	*mm*
Bone 2	-2.24	1.18	3.42	*mm*
Bone 3	-1.89	0.82	2.71	*mm*
Bone 4	-2.24	2.24	4.48	*mm*
Bone 5	-2.24	0.46	2.7	*mm*
Bone 6	-1.21	1.9	3.11	*mm*
Bone 7	-0.82	0.46	1.28	*mm*
Bone 8	-0.99	1.93	2.92	*mm*
Bone 9	-1.89	0.62	2.51	*mm*
Mean			3.07	*mm*
SD			0.94	*mm*
Range			4.5	*mm*

The mean error range of the cuts performed by the developed system was 1.9 mm with standard deviation of 0.55 mm. The Mako set have a mean average of 2.87 mm with standard deviation of 0.84 mm and the Blue Belt set have 3.07 mm with 0.94 mm standard deviation. The range in values in the developed system was approximately half that from commercial systems. This system has proven more capability in cutting and more stability in the repeatability of the cutting comparing to the other systems. There are big differences in the surface finish when comparing between the images of the cuts between the three systems, the designed system provides the smoothest and less spiky surface.

8.4 Femur Surface cut Analysis

A similar cutting procedure was performed on ten femoral sawbones. In this case the implant surface was not flat but a 3-dimensional shape. This made the scanning of the cut volume difficult and as different sizes and implant designs are used in the Mako and Blue Belt systems, direct comparison of cut shapes was not possible. Hence the shape cut and the smoothness achieved by the system are presented, as well as a sawbone cut by the Mako and another by the Blue Belt.

8.4.1 System Femur Cuts

The First photo is for a femur sawbone, followed by another photo of the femur implant shape used by this system, its clear that the shape of this implant is more complex than the tibia implant.

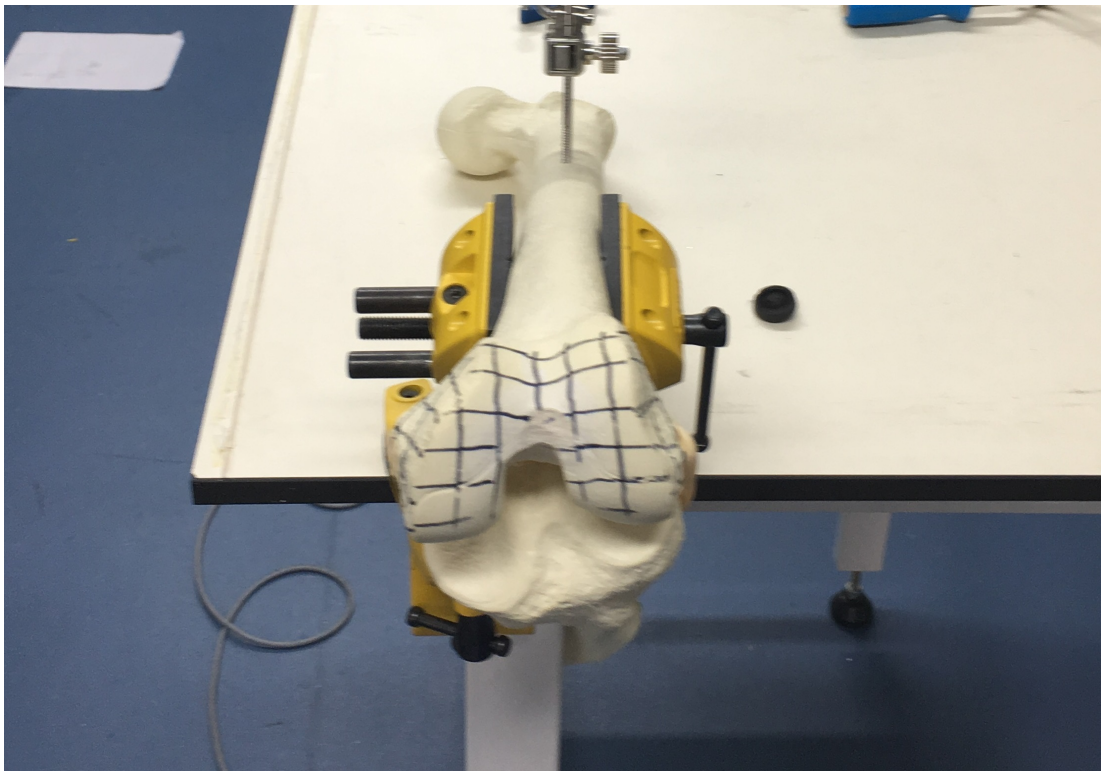


Figure 8.39: Femur Uncut Sawbone

Figure 8.40: Femur Implant

The following three Photos were captured also by camera. The first two photos

are for two sawbones as they were cut by the system, the third photo is another cut sawbone but with the implant fixed in place.

Figure 8.41: System Femur Cut Photo A

Figure 8.42: System Femur Cut Photo B

Figure 8.43: System Femur Cut Photo C

In Photos 8.41 and 8.42 above, the surfaces again been cut very smooth. The curved shape was almost perfect but the last part at the bottom (position) had excess cutting. That was caused by the machine while trying to burr the lower, deeper area (the lower part of the implant in Figure 8.40 above). The problem here was that the CNC machine prototype only had three degrees of freedom and the burr tip was spherical and so has volume. The cutting algorithm was made to move the burr tip centre to the correct positions but as the machine had only 3 DOF as the burr followed the shape deeper into the sawbone, more bone was cut than planned due to cutting by the back edge of the burr. This problem can be more easily visualized from the photos below.

The images below are for a set of femur sawbones that were cut by the system and scanned by Matter and form 3D scanner. As the photos have deep cuts, some areas were unable to be scanned, this areas was represented as black by the software.

Figure 8.44: System Femur Cut Bone 1

Figure 8.44 shows the implant shape was curved smoothly but with wrong orientation.

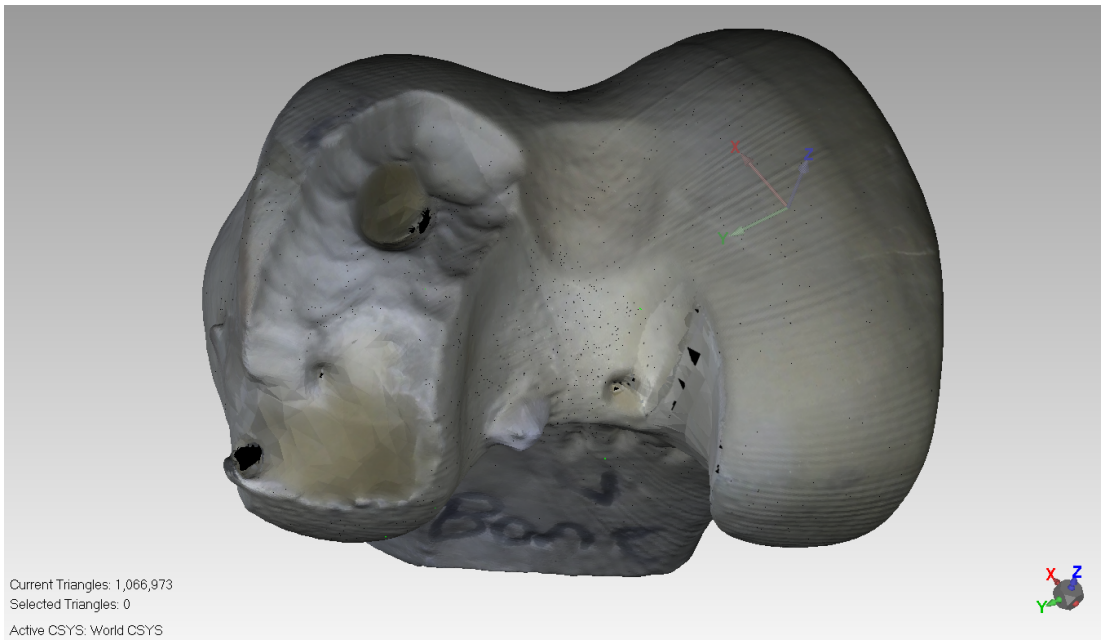


Figure 8.45: System Femur Cut Bone 2

Figure 8.45 shows the implant shape was curved again very smoothly but again with wrong orientation, the bottom part was too deep.

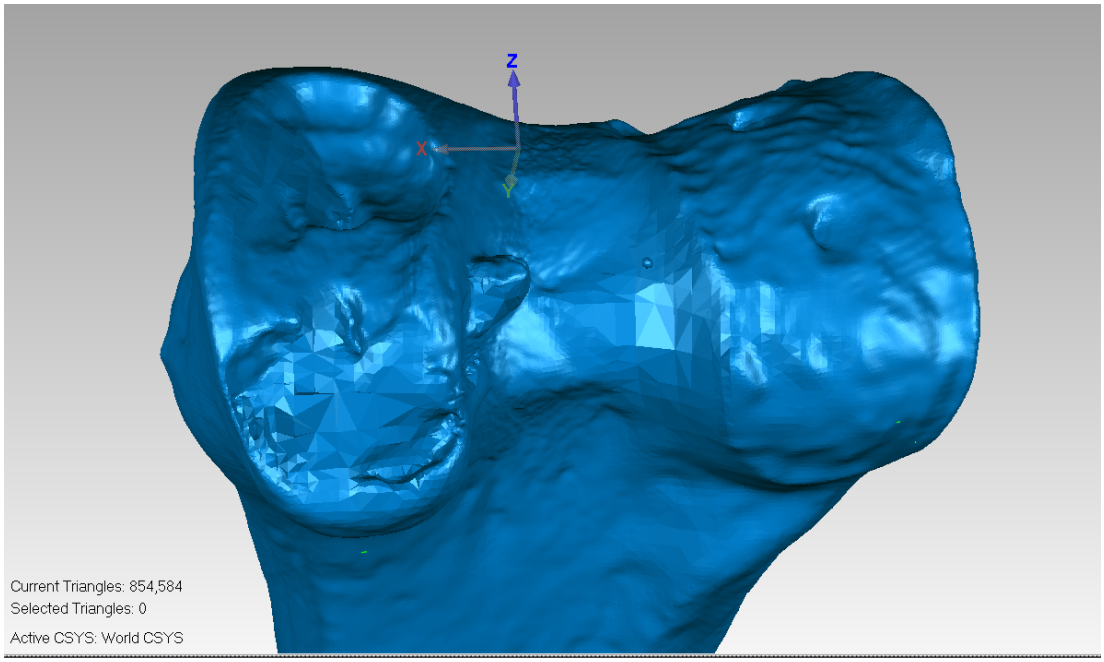


Figure 8.46: System Femur Cut Bone 3

Figure 8.46 shows the implant shape was curved smoothly and with correct orientation.

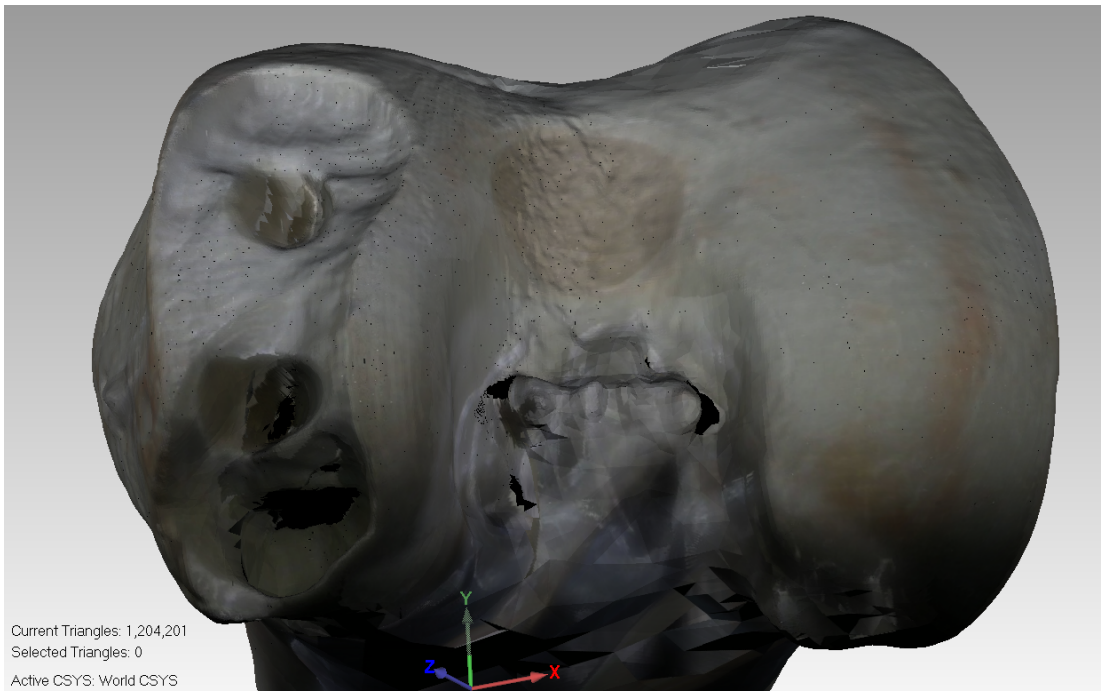


Figure 8.47: System Femur Cut Bone 4

Figure 8.47 shows the implant shape was curved with smooth finish but the lower part was deep because of orientation problem.

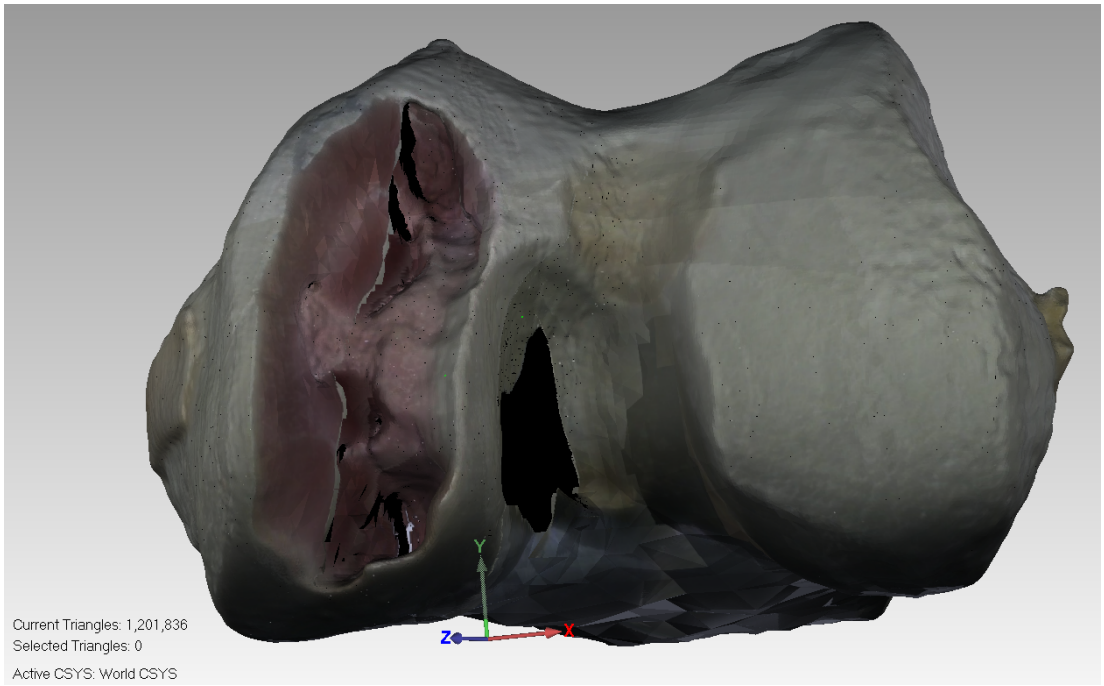


Figure 8.48: System Femur Cut Bone 5

Figure 8.48 shows the implant shape was curved with good finish and correct orientation.

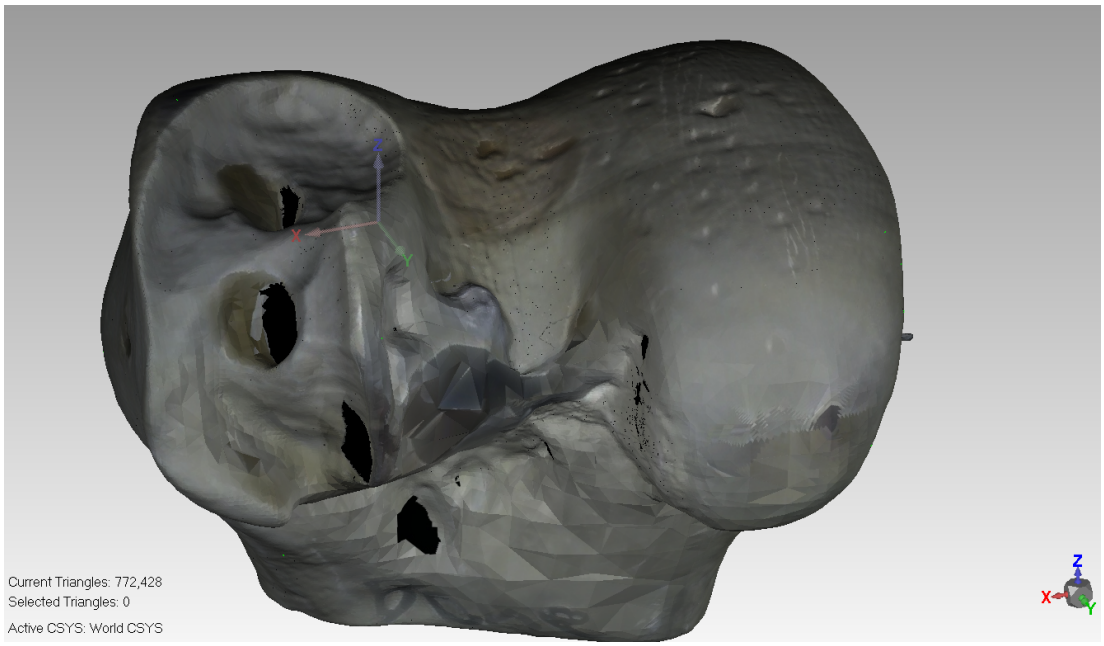


Figure 8.49: System Femur Cut Bone 6

Figure 8.49 shows the implant shape was curved smoothly.

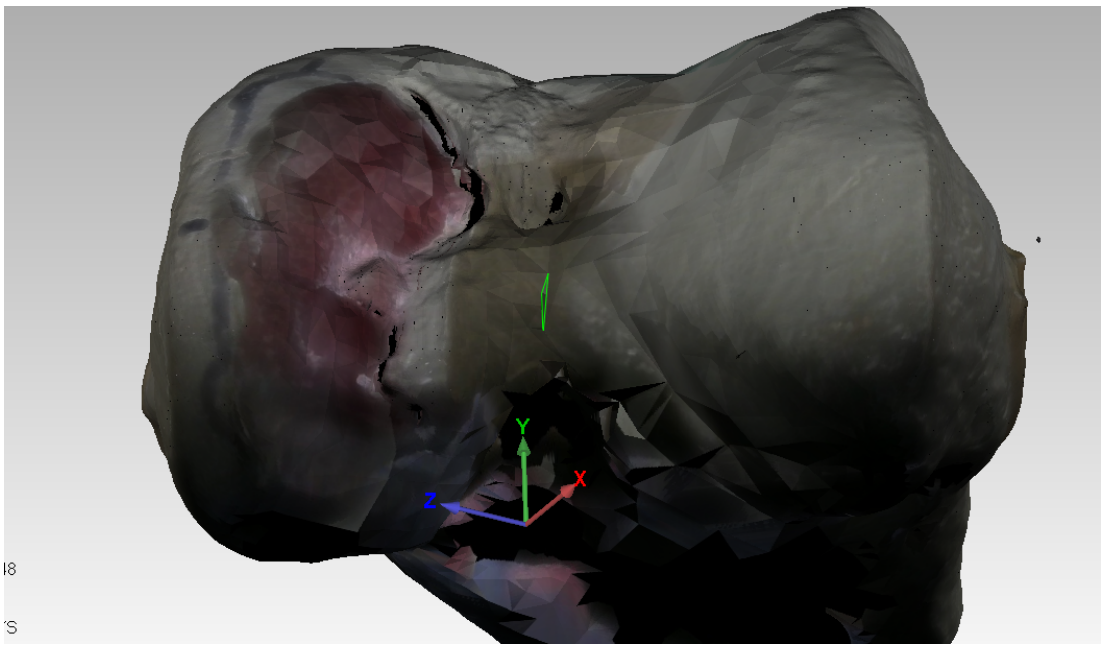


Figure 8.50: System Femur Cut Bone 7

Figure 8.50 shows the implant shape was curved again smoothly.

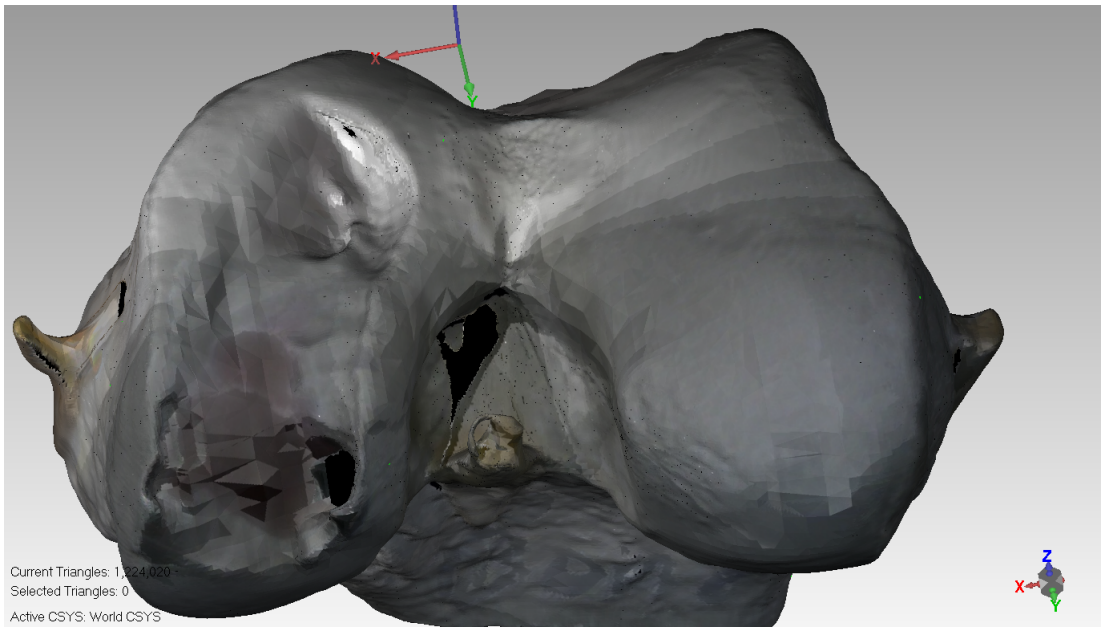


Figure 8.51: System Femur Cut Bone 8

Figure 8.51 shows the implant shape was curved again smoothly.

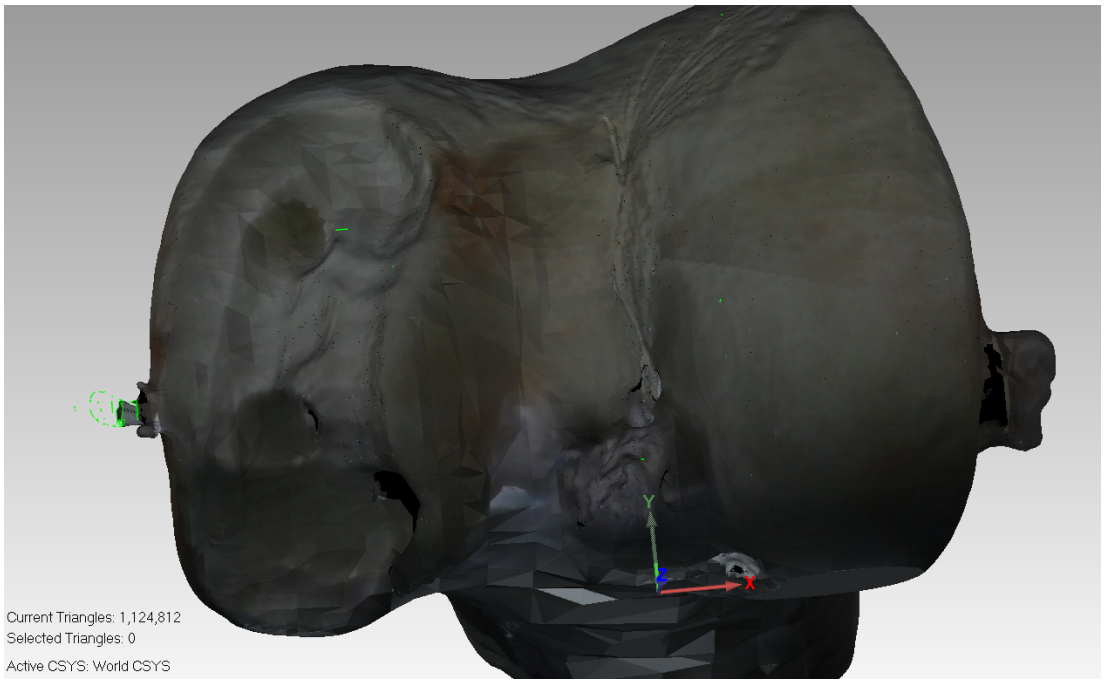


Figure 8.52: System Femur Cut Bone 9

Figure 8.52 shows the implant shape was curved again smooth and fine.

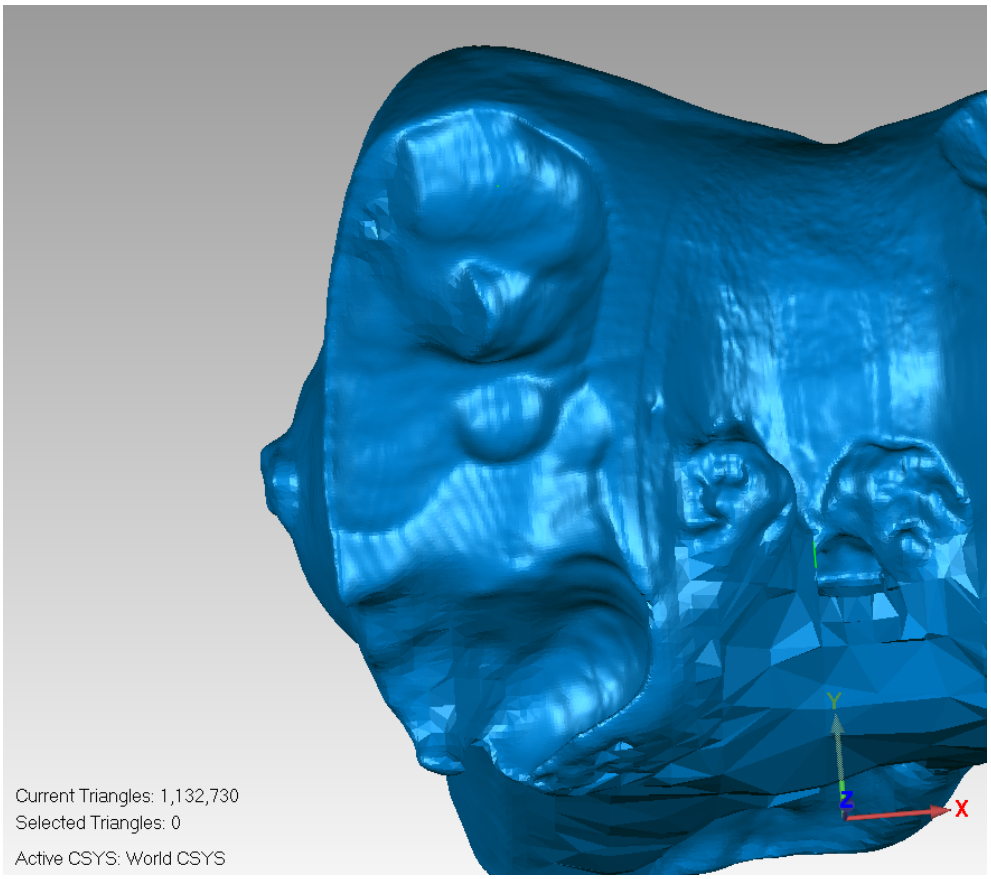


Figure 8.53: System Femur Cut Bone 10

Figure 8.53 shows the implant shape was curved smoothly with good orientation. Some images weren't clear as the scanner failed to scan deep areas. The bones were sawed from the cut centre in half to be able to scan missing areas but parts of the bone were fragile that came off and the surface cut lost its shape.

8.4.2 Mako System Femur Cut

Figure 8.54 below shows a sawbone cut by the Mako system. The uneven surface is very noticeable.

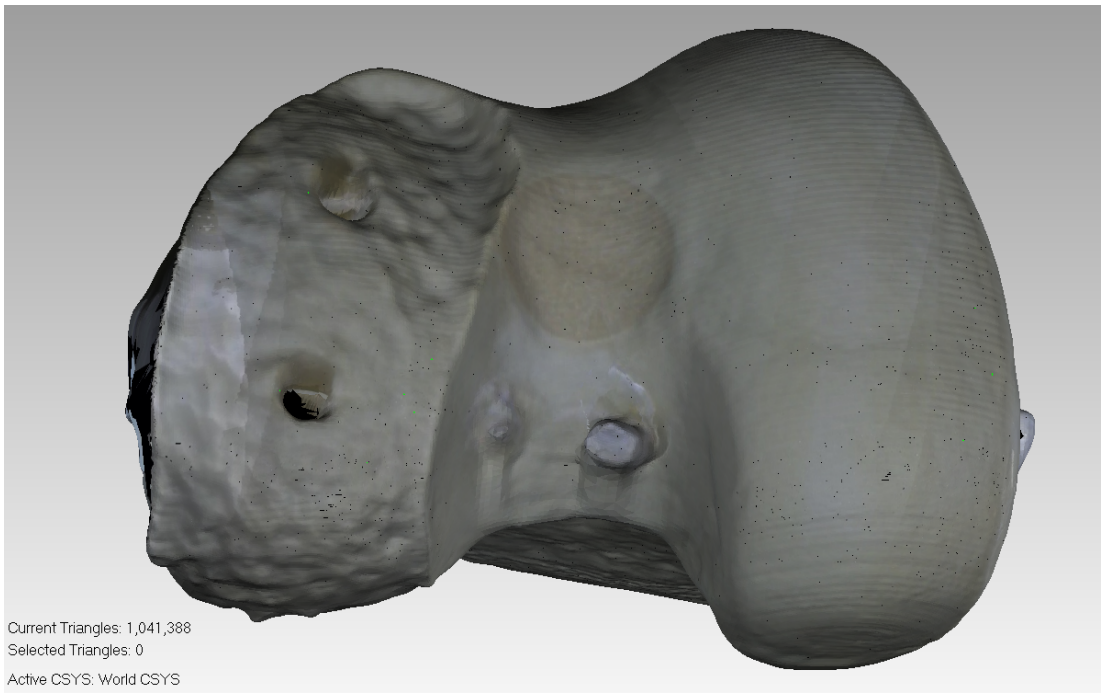


Figure 8.54: Mako Femur Cut Bone

8.4.3 Blue Belt System Femur Cut

Figure 8.55 below shows a sawbone cut by the Blue Belt system. The surface also is very rough.

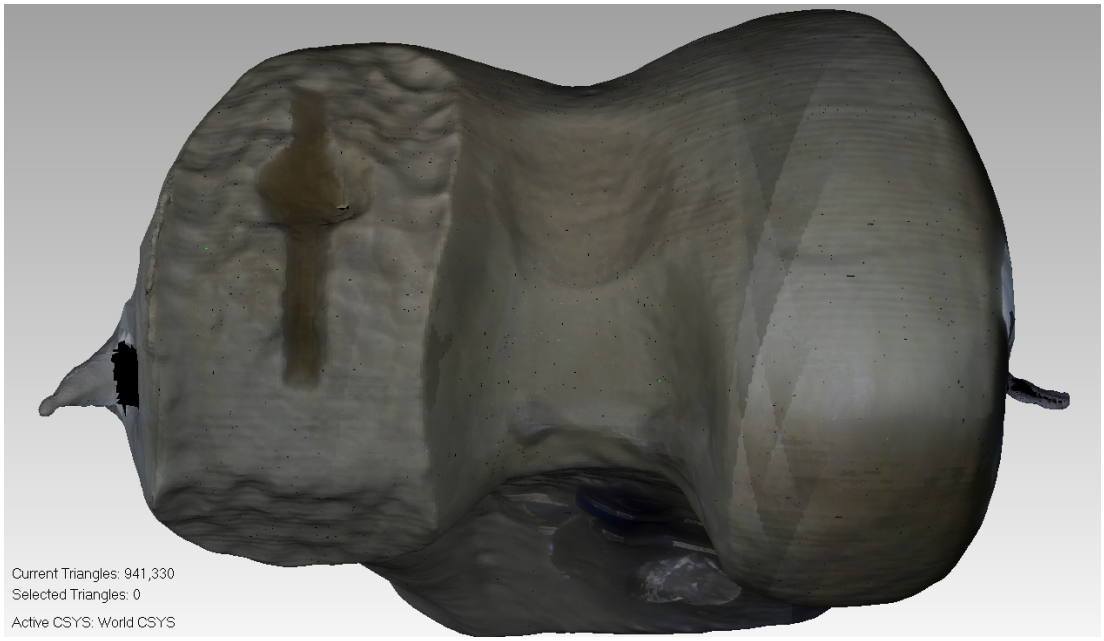


Figure 8.55: Blue Belt Femur Cut Bone

8.5 Conclusion

The developed system has made significant improvement in cut finish when compared to the other two world leading commercial systems. The tibia sawbones cuts showed a mean error of 1.9 mm with standard deviation of 0.55 mm for the cuts performed by this system, where the Mako set had 2.87 mm with standard deviation of 0.84 mm and the Blue Belt set had 3.07 mm with 0.94 mm standard deviation.

The Femur cuts were much more complex but as the Mako and the Blue Belt systems used different implants 3D comparison wasn't performed. Still the images showed that the developed system can perform complex cut with improved surface finish when compared to the other systems. If an extra degree of freedom is included to allow the burr to approach the surface of the cut perpendicularly (or near), so across the entire section of the bone.

The developed system showed improved and more consist surface finish comparing to the other systems.

Chapter 9

Discussion

9.1 General Discussion

The OptiTrack system was one of the factors that made this system cheaper when compared to the Vicon but was the accuracy of this system enough for the procedure?

Three experiments were held to test the OptiTrack system accuracy and efficiency. The first experiment was to measure a known cluster dimensions using the OptiTrack system, the used cluster was the Blunt Probe from the Mako system. The Mako company didn't publish any information about the Blunt Probe dimensions, so another motion capture system (Vicon Bonita) was used to measure the Blunt probe dimensions. There was a 0.3 mm difference between both measures. The second experiment was made to test the accuracy of the system field of view limits and possible errors. The experiment was achieved by scanning known point arrays (segments) with known positions relative to each other, multiple segments were scanned around the borders of the assumed working field area (by setting up a cuboid frame with sides lengths to cover this volume) and the error was $(0.6 - 1.2)mm$. The third experiment was performed to test the motion capture system ability to track moving clusters. The lowest error was 0.16 mm and the highest was 1.05 mm. These results have shown that the OptiTrack system was reliable and accurate enough to be used for navigation in the system. Even particularly given that the clusters were used in static state throughout the

procedures. The clusters could have moved and accuracy would have been retained. Also, the clusters were mainly used around the centre of the field of view to register the burr location and the register the bone surface where the cut will take place, both burr and bone facing each other around the centre of the field of view. In this zone the effective error was 0.3 mm from the first experiment (as the working clusters will be in the centre of the field of view) and the designed cutting resolution was $1mm$, so this was acceptable, for more information about the motion capture experiments refer to Chapter 4.

Later in the development phase, a real-time tracking application was implemented inside the used D-Flow application using the OptiTrack System. D-Flow is a valuable software that provides support for different motion capture systems. A simple drag and drop MoCap module provided all the needed support for development, such as full communication with the motion capture system, direct output from the module to other scripts/modules with the markers coordinates. The MoCap module was also useful in the testing and debugging phase as it has a record feature which enables the developer to record a data file which saves the time-line of moving markers. This is very helpful when there is a case that require testing, for example if the application needed to test the ability to detect moving clusters outside the centre of the cameras field of view, the developer can simply press record then move the cluster where it was required, the recorded file can now be loaded for playback and also looped, this feature saves a lot of time in testing.

D-Flow also provides support for Phidgets hardware parts. The module provides full communication for input and output from Phidgets parts via USB connection. The module also supports analog communication. This module made some parts more like plug and play for developers, such as the used foot switch.

D-Flow was also light on computer resources, when installed. It ran on a low specification computer (only 4GB of RAM, core 2 DUO processor and windows 7). The same computer had Motive software installed and running in the background and yet the application ran smoothly and the clusters were detected in real-time. The disadvantage of the D-Flow is that it doesn't support networking. The

first design for the created system was a real-time application (which can track clusters and communicate with the CNC machine in real-time) but the lack of wireless communication caused by D-Flow forced the creation of an intermediate program (the Java application) that can read the cutting file that D-Flow writes then send it over the network to the Arduino controller. To solve this problem the patient's knee had to be fixed so that the data of the cutting file (the burr path) was implemented with no change. It is recognised that this was a considerable drawback that previous robotic orthopaedic systems haven't reached the market when this was required due to its invasiveness and cumbersomeness, however in our system a free knee would be possible in future prototypes provided the movement was limited and didn't exceed the movement capability of the CNC machine. To enable real-time cutting of a free limb the Phidgets module could be hard-wired to the micro-controller and used to send the movements of burr relative to the limb. Allowing the burr position to adapt to movement of the limb.

Enabling the real-time feature will not lead to a completely free bone, the patients leg still require to be held in a leg holder. If the knee joint was set to have more movement capabilities, then the CNC unit would have to be upgraded to have an extra degree of freedom (pitch) –from the Pitch, Roll and Yaw – this could be accomplished by many different approaches:

1. By mounting the CNC on a robotic arm, the arm has to be stiff and can support the machine weight and keep steady from the forces facing the burr while performing the procedure.

2. By fixing the CNC base on a steward platform.

3. By fixing the base on a spherical joint with control.

The network communication proved problematic this was caused by the Wi-Fi shield silent mode as discussed in Chapter 7. The protocol was however successful and gave the working resolution of the cut. All the bones cut by this system were fully cut remotely using the remote control feature. The CNC machine was set underway with no wired connection to the running computer. Path stability was

achieved by the created handshake network transmission control protocol. This protocol ensured safe transfer of the data from the file on the computer (produced by the D-Flow application) to the CNC machine controller.
If further versions of D-Flow enabled a network module then the real-time feature would be active alongside the remote control feature. If for some reason the working procedure needed to be extended to perform different procedures or task that would require the transferring of larger chunks of data, then the Arduino and the Wi-Fi shield could be replaced by a more robust wireless network hardware module such as the Raspberry Pi or the new Intel controllers and Wi-Fi modules. These controllers have earned recently strong positive feedback in the field of communication.

9.2 Discussion of Cutting Results

The developed system prototype had some problems with cut orientation. The Mako system registration process is performed by registering 40 points, it takes about 20 minutes to complete the process. The current system registration process had only three points. At early development three points seemed to be enough, theoretically three points are enough to work out the correct orientation but human error in locating the points and recording noise in storing these points can lead to inaccuracy. If the orientation is incorrect then in some areas more bone will be removed than planned while in others less or no bone will be removed. In early trials (when the procedure was performed on tibia bone 1 and tibia bone 2, refer back to Section 8.2.1 for more details) the Blunt probe was used, this probe has a hemi sphere as a tip, see Figures 6.10, 6.14 for more details. Some times during the registration process the probe gets tilted away from being perpendicular to the bone surface causing the tip to be up to 1 mm away from the marked point and touching the bone with the side of the probe tip. After the first two bone cuts the Blunt probe was replace by the Sharp probe to be more accurate as the tip is sharper like a needle. The factor of human location error still exists. To eliminate this error the registration process should be completed

by a tool rather than registering each point by probe based on human sight.
The Mako system cutting mechanism is based on an operator holding a robotic arm, see the results chapter in Section 8.3.2 for the Mako results, while the Blue Belt system cutting mechanism was based on an operator holding a robotic cutting machine by his/her arm, see Section 8.3.3. Both systems failed to accomplish a sufficiently smooth surface and it is believed that's because of human error (The Operator). Three approaches could be made to improve the registration process:

1. By using the existing Motion capture system along with the created 3D scanner application. In this method the location of the tip of the sharp probe would be continuously monitored while it traces the surface of the bone to be cut, this method can improve the registration process from orientating using three scanned points to hundreds of points. However, the probe must stay in contact with the bone surface at all times.

2. Using laser scanner to scan thousands of points of the bone surface.

3. Using a special designed tool for registration, see Figure 9.1 below, along with MYKNEE surgical tailored instruments.

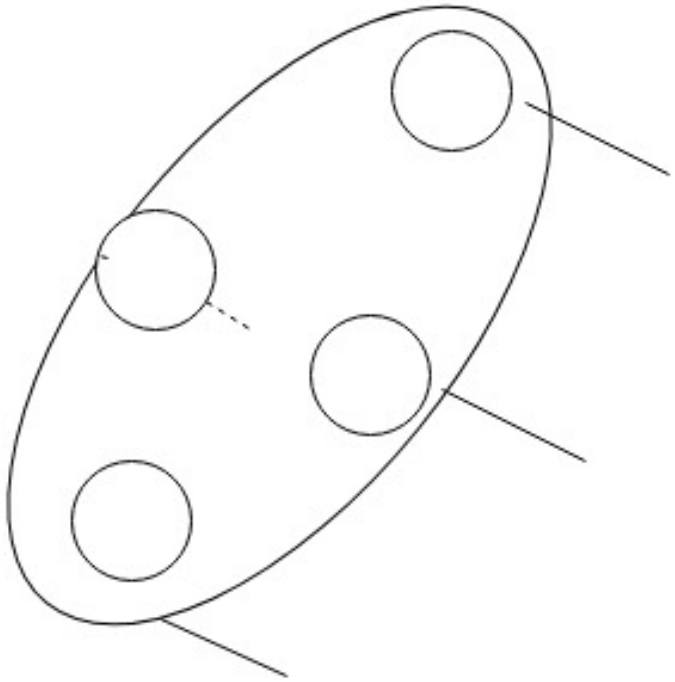


Figure 9.1: Proposed Tool for registration process

The third approach is to use the proposed tool along with the MYKNEE instrument. Figure 9.1 shows the proposed tool which is a simple track-able cluster with four needles sticking out with different lengths to fit the bone surface. The MYKNEE tool is a personalized surgical instrument created from a plastic 3d model of the patient's knee using the image from the diagnostic scan. This instrument can fit precisely on the knee allowing the operator to perform the registration process more accurately. All alternatives should be tested and the method with best orientation will be implemented.

In addition to orientation errors the cut quality is affected by the surface roughness. Both extremely expensive systems (Mako + Blue Belt) showed surface roughness cutting errors in the tibia cuts (See Section 8.3.4). The maximum error range in the developed system tibia cuts was 2.9 mm, while the average error was 1.9 mm and the standard deviation was 0.55 mm when performing the procedure on 9 tibia's. For the Mako system the maximum error was 4.99 mm with 2.87 mm average error and 0.84 mm standard deviation, also on a 9 tibia's

set. The Blue Belt system had maximum error range of 4.5 mm while the average error was 3.07 mm and standard deviation of 0.94 mm, the set was also 9 tibia's. Almost all The Mako and Blue Belt bones showed a spiked and non-smooth surface. This is caused by the controlling mechanism both systems used for cutting. The Blue Belt was a simple robotic hand-held tool with burr that is retrieved once the its tip exits the marked cutting area. Blue Belt cuts (in bone 4 Figure D.22 and bone 7 Figure D.28 in the Appendix and bone 8 Figure 8.38 in the results chapter) showed excessive cuts on the border of the tibial surface which indicates that the burr retrieval safety mechanism isn't optimal. the maximum range between the tibial cuts performed by the Blue Belt system was 4.5mm. The Mako systems shows better performance than the Blue Belt but all Mako bones showed spiked surface, still the maximum error range in the tibial cuts performed by the Mako system was 4.99mm which is more than the Blue Belt system. In conclusion the developed system in this showed a much-improved surface finish. The Femur cuts were more complex than the tibial as the femur implant required the machine to cut curves (three-dimensional shape). The results couldn't be compared with the Mako or Blue Belt cuts as both systems used different implants, however the system cuts showed remarkable surface finish comparing to the other systems which showed a bubbly uneven surface (see Figures 8.54 and 8.55). However, the femur cuts performed by the developed system showed problem with the cut shape. The designed CNC machine had only 3 degrees of freedom, with only 3 degrees of freedom the machine will have a constant attack angle relative to the bone surface. Each femural condyle surface has about a 90-degree curve from one end to the other (2 surfaces almost perpendicular on each other). Figure 8.40 of the femur implant shows the shape implant correct curve that needed to be cut. As proposed earlier adding one more degree of freedom would solve the problem and enable the attack angle to change continuously so that the burr tip remains perpendicular to the femur surface all the times. There was also a problem for the femur cuts in the orientation. This problem is more obvious in the femur as it has a more complex surface than the tibia.

The surface finish smoothness and less bubbly bone cut surface produced by

our developed system has some very important benefits. First to close the gab between the bone surface and the implant and hence reduce the healing time. Second, it was reported that the cement mantle thickness in the cement TKA prosthesis should be $1.4mm$ for the femur and $0.8mm$ for the tibia (Ko et al., 2017). If the same thickness is used in UKA then it's hard to imagine how this thickness is maintained if the Mako and Blue Belt systems tibial cuts have uneven surface with peaks of $4.99mm$ and $4.5mm$ respectively (see Section 8.3). Given that these bone cuts were undertaken by senior robotically trained orthopaedic consultants it is likely that these results represent the smallest maximal errors that are to be expected from the Mako and blue belt systems. Third, having smoother surface can help improve cement-less prosthesis as the ideal gap between the bone and the implant in TKA was reported to be less than $1mm$ (Bonnin et al., 2013). Again, if the same numbers are applied for the UKA, then the developed system is the closest to achieving this surface finish. The developed system is promising with the numbers and it is believed that after fixing the orientation problem the cutting procedure will show improved cut and surface finish.

9.3 Cost

Robotic-assisted knee arthroplasty systems may increase the efficiency and effectiveness of the procedure. However, these systems are expensive, the Mako system costs nearly $\$1,000,000$ and the Blue Belt system costs nearly $\$300,000$, see the literature chapter for more information.

In this section the cost for the developed system will be considered in relation to the other available systems. The OptiTrack motion capture system was chosen because its much cheaper comparing to other used systems. The Polaris cameras used in the Mako system costs $\$35,000$ for (Spectra + Vicra) and the Vicon Bonita system costs £40000 and they get mounted on a $\$15,000$ stand but the OptiTrack Flex V100:R2 twelve camera system costs $\$11,346$, refer back to Chapter 4 for more details.

The CNC unit and the controller box main components are five motors, five

motor drivers, five rails, Arduino, Arduino Wi-Fi shield, small control board and 15A power supply. The cost of one rail, one motor and one motor driver costed $135.5 multiply it by five hence the CNC costed $677.5, where the Arduino Duo costed $35.5, the Wi-Fi shield costed $81 and the power supply costed $48. Add a few more dollars for the power cords, box and non-costly components like the regulator, resistances, vero board $5, etc., $10 will be added as the cost for these components, that makes the hardware total cost around $852 plus the motion capture system hardware the total will be $12,198.

For the software cost, the Java and Arduino application are free to use but Motive software tracker license costs $999 (*Motive:Tracker - Motion capture and 6 DOF object tracking*, 2018b).

So, the capital cost of the system is $13,197, equivalent of the cost Mako camera stand only. The total cost will include both manufacturing and company costs, so the estimated total cost for the system as a product would likely be $131,970,0. So, the estimated price costs 12% of the Mako price and 40% of the Blue Belt system. It is nearly impossible to afford systems like the Mako or the Blue belt in third world countries. Hopefully this system will make the expected improvement to bone burring in knee arthroplasty offered by robotics available in non-industrial countries like my own.

9.4 Mobility

The Mako Rio system is a large robot that was designed and built with an arm to perform the procedure, see Figure 2.3 in the Literature. This robot has a computer and terminal inside and is extremely heavy, there isn't any published information about its size or weight from the manufacturer but it is estimated to be least $200kg$, it's mobile but its mobility isn't easy because of its size, weight and bulky power cords. These factors are one of the reasons that makes the robot highly expensive. The robot has four wheels two fixed and two rotational with a base dimensions about 100 X 50 mm, that made the mobility of the system even harder in narrow place or in a place with many objects like the surgery theatre.

Moving the robot round in the theatre would give the operator a very hard time. First you need to step on a pedal a few times to lift the robot up and make the wheels functional, then the robot base has to be set moving – which isn't easy because of the weight– while changing the angle of the base at the same time and avoiding cables or someone's feet. The Mako system isn't just the robot, there are two other stands, the camera stand and the operator stand. The Camera stand has a terminal, two mounted cameras on top and a bulky wire to the robot. The Operator stand has a terminal, keyboard and mouse for the operator and it's also wired to the robot with a bulky wire. All three are in different locations but connected with these wires. On the other hand, the system developed in this project was wireless with a remotely controlled CNC machine. The CNC unit and the controller box main components are five motors, five motor drivers, five rails, Arduino and Arduino Wi-Fi shield, small control board, 15A power supply. The controller box and the CNC are light, they can be carried easily by a single person at the same time, the controller box can be mounted on a wall or placed on a table, the unit and the box weights around $25kg$. The main program can be worked on a computer not only separate from the machine but it could be in another room, or another country. The OptiTrack was mounted on the walls but generally can be fixed anywhere, the cameras didn't occupy any space in the working theatre.

9.5 Summary

Robotic-assisted surgery is becoming more mainstream. The existing systems (Mako and Blue Belt) are very expensive and to third world countries their existence might be fictional. They are manually functioned; hence human error factor exists. On the cost comparison the developed system costed 0.01% of the Mako Rio system and 0.04% of the Blue Belt system, the system cost was evaluated based on the licenses and hardware cost and for the final version to become a product manufacturing process took place then extra expenses might be added, if the total cost was tripled then the ratio of the comparison will be 0.039% and

0.13197%, the cost still didn't reach a single Vicon camera used by the other systems. One of the major approaches to cut down the cost was using cheaper navigation system. OptiTrack motion capture system has proven to be fit to replace the Vicon system in the designed procedure as several experiments were performed to acquire this result.

The designed system is mobile and light. The CNC unit with the controller box weight around $25kg$. With the remote controlled feature it's much easier to install and move around in the theatre of operation with only ordinary power cord coming out of the controller box. The whole system fitted in one large travelling bag with a full OptiTrack system. On the other hand, the Mako system is at least $200kg$ has three components and they are all wired together, it's not easy to move around.

The tibia cuts by the developed system had an average error of 1.9 mm and the standard deviation of 0.55 mm, comparing with the Mako system which had 2.87 mm average error and 0.84 mm standard deviation, while the Blue Belt system had an average error was 3.07 mm and standard deviation of 0.94 mm. The developed system also showed better quality and smoothness of surface finish of the tibia and femur cut bones when compared to the tibia's cuts by the Mako and Blue Belt systems.

9.6 Limitations

One of the limitations of this prototype was that the bone had to be fixed to be cut. This was necessary because the cutting plan could not be updated in real-time as the cut was occurring because of the limitations of the wireless network. Fixing the bone rigidly was tried in early robotic systems such as Robodoc but proved problematic. Therefore it would be hard to replicate in real practice. It was originally intended that the bone would be constrained by a leg holder but the bone would be able to move under the limits of the soft tissue. The software was designed to allow this, but this facility could not be used due to the WiFi problems. In the future the CNC cutting machine will be directly wired to the

computer, allowing the cutting plan to be updated based on the motion capture data for the tibial and femoral bones and hence a semi constrained leg would be allowable.

The designed CNC device is an active robotic machine, however all it does is cut the bone to the design of the surgeon. A surgeon is still needed to plan the surgery, modify the soft tissue and fix the implants. it is difficult to see how these aspects of the surgery could be automated. One area were the current system could play a role would be automatically suggesting the position and orientation of the implants and therefore the bone cuts. However even in this limited application the suggested implant positions would need to be checked and modified if necessary by the surgeon. It is likely therefore that the use of robotics in arthroplasty surgery will remain limited to burring of the bone.

The CNC machine showed limitation when cutting the curved shape of the femoral implant because the machine only had three degrees of freedom. This meant that the angle of attack of the burr could not be perpendicular to the bone surface when the burr cut a curve. This was particularity so when the curvature of the implant approached or exceeded 90 degrees. If this happened the shallow angle of attack of the burr meant the top surface of the burr cut the desired shape but the back surface of the burr undercut the bone and hence removed bone that should have been preserved. It was intended that the CNC machine would be mounted on positioning arm which would allow the burr to be positioned at different points relative to the joint and hence avoid this problem. This didn't prove possible within the project timescales.

9.7 Future Work

The CNC unit should be upgraded to have an extra degree of freedom (pitch) –from the Pitch, Roll and Yaw– to enable cutting a free limb. This can be accomplished by different approaches:

1. By mounting the CNC on a robotic arm, the arm has to be stiff and can support the machine weight and keep steady from the forces facing the burr

while performing the procedure.

2. By fixing the CNC base on a steward platform.

3. By fixing the base on a spherical joint with control.

The burr length is $50mm$, this caused the burr to bend due to cutting forces after around 14 cutting trials. A support system will be designed and attached to the burr so that it can rotate without any resistance from the support.
The three-point registration process showed orientation problems. Three approaches could be made to improve the registration process:

1. By using the existing Motion capture system along with the created 3D scanner application. In this method the location of the tip of the sharp probe would be continuously monitored while it traces the surface of the bone to be cut, this method can improve the registration process from orientating using three scanned points to hundreds of points. However, the probe must stay in contact with the bone surface at all times.

2. Using laser scanner to scan thousands of points of the bone surface.

3. Using a special designed tool for registration, see Figure 9.1 below, along with patient specific surgical instruments.

The network communication proved problematic this was caused by the Wi-Fi shield silent mode as discussed in Chapter 7. The protocol was however successful and gave the working resolution of the cut. To solve the wireless communication problem with the Arduino Wi-Fi shield module if larger files needed to be used, then it is recommended to either use wired communication or the usage of more stable Wi-Fi modules.
After testing and implementing all proposed methods, the new prototype should be retested, then the best solution will be applied. First the upgraded system should be tested on a sawbones. Second the system should be tested on animal bones. Third the system should be tested on human cadavers. Applications could then be made for first in man human trials. provided these stages are successful,

the system would then begin the long and costly journey to becoming a licensed medical product. In this thesis we have taken the first steps on this journey to a semi-automatic robotic orthopaedic surgical system.

Chapter 10

Conclusion

Robotic assisted surgery is becoming mainstream. These systems bring better accuracy and efficiency for the medical world. With all the positive features that have been achieved by the robotic assisted systems, comes higher cost for the patients and the healthcare provider.

The main component of the robotic system was the navigation system (Motion Capture System). The OptiTrack motion capture system was tested and selected to be more affordable choice when compared to the Vicon and Polaris motion capture systems. The OptiTrack experimental testing are as below:

1. The first experiment showed that the navigation system accuracy was $0.3mm$ different than the Vicon system, which is acceptable within the functionality of the full system as the cutting process was set to perform a $1mm$ cut.

2. The second experiment showed accuracy at the origin of the system was ± 1 mm, while at the extreme ends of the field of view ± 2.5 mm.

3. The third experiment showed the robustness of tracking the objects while moving in field of the cameras view.

These experiments have proven the OptiTrack system to be valid as a more affordable motion capture system that can perform as the system navigator.

In addition, a Wireless CNC unit with three degrees of freedom was designed and built to perform the cutting procedure. The machine features were:

1. The machine is guided by the navigation through an application that sends the machine movements coordinates over wireless network to enable remote control feature, as the running computer and the CNC machine have no wired connection.

2. The machine hardware components costed around $852.

3. The CNC machine has a resolution of 200 step/mm, which means that the burr can be driven by $1/200mm$ accuracy.

4. The machine was lighter, around $25kg$ with only the power cord coming out which makes it easy to be portable when mounted on a trolley.

After experimenting on sawbones, it was clear that the machine needs one more degree of freedom to fully perform the required cut.
Moreover, the system's main application responsible for creating the control movements of the burr mounted on the CNC machine was created by D-Flow software. The application tasks are:

1. The main application was successfully able to detect moving clusters in real-time.

2. The main application was able to register the machine and knee-joint location.

3. The main application uses the registered locations then creates the burr path cutting (movements file) successfully.

Also, a separate D-Flow application was created to scan three-dimensional surfaces. This application was used to scan the implants shape and insert them into the main application to assist planning the path route for the burr.
Furthermore, in order to safely transfer the data from the running PC to the CNC machine a series of operations was established:

1. A Java-written application was created to act as a server side in order to transfer the burr path coordinates.

2. Another application was created on the CNC machine micro-controller to act as the client side.

3. The communication between both applications was wireless. A transfer network protocol was created in insure lossless transfer of data between server and client.

The micro-controller application sends required control to the CNC motors drivers according to the received coordinates.
In conclusion, the autonomous developed system has an estimated product cost 12% of the Mako price and 40% of the Blue Belt system price. The system also showed better cutting accuracy. The tibia cuts performed by the developed system had an average error of 1.9 mm and the standard deviation of 0.55 mm, compared to the Mako system which had 2.87 mm average error and 0.84 mm standard deviation, while the Blue Belt system had an average error was 3.07 mm and standard deviation of 0.94 mm. The developed system also showed better quality and smoothness of surface finish of the tibia and femur cut bones when compared to the tibia's cuts by the Mako and Blue Belt systems. With much more affordable system and better cutting quality this system should provide the needed improvement to the knee arthroplasty all around the world.

References

3D Systems. (2018). *Software — 3d systems.* Retrieved 2018-04-26, from https://www.3dsystems.com/software

About cnc machining. (2017). Retrieved 2017-10-31, from https://www.thomasnet.com/about/cnc-machining-45330503.html

Allard, P.-H., & Lavoie, J.-A. (2014). Differentiation of 3d scanners and their positioning method when applied to pipeline integrity. In *9th pipeline technology conference 2014.*

Altintas, Y. (2012). *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and cnc design.* Cambridge university press.

Argenson, J., & O'Connor, J. J. (1992). Polyethylene wear in meniscal knee replacement. a one to nine-year retrieval analysis of the oxford knee. *Bone & Joint Journal*, *74*(2), 228–232.

Bell, S. W., Anthony, I., Jones, B., MacLean, A., Rowe, P., & Blyth, M. (2016). Improved accuracy of component positioning with robotic-assisted unicompartmental knee arthroplasty: data from a prospective, randomized controlled study. *JBJS*, *98*(8), 627–635.

Blyth, M., Smith, J., Jones, B., MacLean III, A., Anthony, I., & Rowe, P. (2013). Does robotic surgical assistance improve the accuracy of implant placement in unicompartmental knee arthroplasty. In *Aaos 2013 annual meeting, chicago, il.*

Bonnin, M., Amendola, N. A., Bellemans, J., MacDonald, S. J., & Menetrey, J. (2013). *The knee joint: surgical techniques and strategies.* Springer Science & Business Media.

Bourne, R. B., Chesworth, B. M., Davis, A. M., Mahomed, N. N., & Charron, K. D. (2010). Patient satisfaction after total knee arthroplasty: who is satisfied and who is not? *Clinical Orthopaedics and Related Research®*, *468*(1), 57–63.

Brasseler usa - medical. (2018). Retrieved 2018-02-18, from https://brasselerusamedical.com/products/burs/

Braun, M. (1994). *Picturing time: the work of etienne-jules marey (1830-1904).* University of Chicago Press.

Build your own motion capture system. (2018). Retrieved 2018-03-23, from http://optitrack.com/systems/\#movement/flex-3/12

Chawla, H., Pearle, A., et al. (2016). Robotic-assisted knee arthroplasty: An overview. *American journal of orthopedics (Belle Mead, NJ)*, *45*(4), 202–211.

Chen, Z., Wang, C., Jiang, W., Tang, N., & Chen, B. (2017). A review on surgical instruments of knee arthroscopic debridement and total hip arthroplasty. *Procedia CIRP*, *65*, 291–298.

Cobb, J., Henckel, J., Gomes, P., Harris, S., Jakopec, M., Rodriguez, F., ... Davies, B. (2006). Hands-on robotic unicompartmental knee replacement: a prospective, randomised controlled study of the acrobot system. *The Journal of bone and joint surgery. British volume*, *88*(2), 188–197.

Delta Surgical. (2018). *Burrs - delta surgical.* Retrieved 2018-02-18, from http://www.deltasurgical.co.uk/drill-systems/burrs

D-flow - motekforce link. (2018). Retrieved 2018-01-27, from https://www.motekforcelink.com/product/d-flow/

Diaz Novo, C., Alharbi, S., Fox, M., Ouellette, E., Biden, E., Tingley, M., & Chester, V. (2014). The impact of technical parameters such as video sensor technology, system configuration, marker size and speed on the accuracy of motion analysis systems. *Ingeniería mecánica, tecnología y desarrollo*, *5*(1), 265–271.

Dillon, N. P., Fichera, L., Wellborn, P. S., Labadie, R. F., & Webster, R. J. (2016). Making robots mill bone more like human surgeons: Using bone density and anatomic information to mill safely and efficiently. In *Intelligent robots and systems (iros), 2016 ieee/rsj international conference on* (pp. 1837–1843).

Dillon, N. P., Kratchman, L. B., Dietrich, M. S., Labadie, R. F., Webster III, R. J., & Withrow, T. J. (2013). An experimental evaluation of the force requirements for robotic mastoidectomy. *Otology & neurotology: official publication of the American Otological Society, American Neurotology So-*

ciety [and] European Academy of Otology and Neurotology, *34*(7), e93.

Dockter, R. L. I. (2013). *A fast, low-cost, computer vision-based approach for tracking surgical tools*. University of Minnesota.

Dunbar, N. J., Roche, M. W., Park, B. H., Branch, S. H., Conditt, M. A., & Banks, S. A. (2012). Accuracy of dynamic tactile-guided unicompartmental knee arthroplasty. *The Journal of arthroplasty*, *27*(5), 803–808.

Eder, M., Brockmann, G., Zimmermann, A., Papadopoulos, M. A., Schwenzer-Zimmerer, K., Zeilhofer, H. F., ... Kovacs, L. (2013). Evaluation of precision and accuracy assessment of different 3-d surface imaging systems for biomedical purposes. *Journal of digital imaging*, *26*(2), 163–172.

FARO Technologies. (2018a). *Faroarm: The global standard for arm technology.* Retrieved 2018-04-24, from `https://www.faro.com/products/factory-metrology/faroarm/`

FARO Technologies. (2018b). *Faro quantumm arm & faro quantumm scanarm hd tech sheet.* Retrieved 2018-04-24, from `https://www.faro.com/resource/faro-quantum-m-arm-faro-quantum-m-scanarm-hd-tech-sheet/`

Faugeras, O., & Robert, L. (1996). What can two images tell us about a third one? *International Journal of Computer Vision*, *18*(1), 5–19.

Felson, D. T. (2006). Osteoarthritis of the knee. *New England Journal of Medicine*, *354*(8), 841–848.

Foran, J. (2016). *Unicompartmental knee replacement - orthoinfo - aaos.* Retrieved 2018-04-18, from `https://orthoinfo.aaos.org/en/treatment/unicompartmental-knee-replacement/`

Gage, J. R. (1993). Gait analysis. an essential tool in the treatment of cerebral palsy. *Clinical orthopaedics and related research*(288), 126–134.

Geijtenbeek, T., Steenbrink, F., Otten, B., & Even-Zohar, O. (2011). D-flow: immersive virtual reality and real-time feedback for rehabilitation. In *Proceedings of the 10th international conference on virtual reality continuum and its applications in industry* (pp. 201–208).

Goodfellow, J., Kershaw, C., Benson, M., & O'Connor, J. (1988). The oxford

knee for unicompartmental osteoarthritis. the first 103 cases. *Bone & Joint Journal*, *70*(5), 692–701.

Goodfellow, J., & O'Connor, J. (1978). The mechanics of the knee and prosthesis design. *Bone & Joint Journal*, *60*(3), 358–369.

Goodfellow, J., O'Connor, J., Dodd, C., & Murray, D. (2011). *Unicompartmental arthroplasty with the oxford knee*. Goodfellow Publishers Limited.

Goodfellow, J., Tibrewal, S., Sherman, K., & O'Connor, J. (1987). Unicompartmental oxford meniscal knee arthroplasty. *The Journal of arthroplasty*, *2*(1), 1–9.

Guerra-Filho, G. (2005a). Optical motion capture: Theory and implementation. *RITA*, *12*(2), 61–90.

Guerra-Filho, G. (2005b). Optical motion capture: Theory and implementation. *RITA*, *12*(2), 61–90.

Harris-Love, M. O., Siegel, K. L., Paul, S. M., & Benson, K. (2004). Rehabilitation mnagement of friedreich ataxia: Lower extremity force-control variability and gait performance. *Neurorehabilitation and neural repair*, *18*(2), 117–124.

Harwin, S. F. (2003). Complications of unicompartmental knee arthroplasty. In *Seminars in arthroplasty* (Vol. 14, pp. 232–244).

Heisel, U., & Krondorfer, H. (1997). Application of the surface method for vibration analysis to cnc-routers. In *Proc. of the 13th int. wood machining seminar* (pp. 253–264).

Hill, C., El-Bash, R., Johnson, L., & Coustasse, A. (2015). Robotic joint replacement surgery: does technology improve outcomes? *The health care manager*, *34*(2), 128–136.

Hodt-Billington, C., Helbostad, J. L., & Moe-Nilssen, R. (2008). Should trunk movement or footfall parameters quantify gait asymmetry in chronic stroke patients? *Gait & posture*, *27*(4), 552–558.

Hood-Daniel, P., & Kelly, J. F. (2009). *Build your own cnc machine*. Springer.

Iosub, A., Nagit, G., & Negoescu, F. (2008). Plasma cutting of composite materials. *International Journal of Material Forming*, *1*(1), 1347–1350.

Jain, V., Dixit, P., & Pandey, P. (1999). On the analysis of the electrochemical spark machining process. *International Journal of Machine Tools and Manufacture*, *39*(1), 165–186.

Kazarian, G. S., Lonner, J. H., Maltenfort, M. G., Ghomrawi, H. M., & Chen, A. F. (2018). Cost-effectiveness of surgical and nonsurgical treatments for unicompartmental knee arthritis: A markov model. *JBJS*, *100*(19), 1653–1660.

Keraita, J. N., & Kim, K.-H. (2007). Pc-based low-cost cnc automation of plasma profile cutting of pipes. *ARPN Journal of Engineering and Applied Sciences*, *2*(5), 1–7.

Ko, D. O., Lee, S., Kim, K. T., Lee, J. I., Kim, J. W., & Yi, S. M. (2017). Cement mantle thickness at the bone cement interface in total knee arthroplasty: Comparison of ps150 rp and lps-flex knee implants. *Knee surgery & related research*, *29*(2), 115.

Kolarevic, B. (2001). Digital fabrication: manufacturing architecture in the information age. *ACADIA*.

Koshino, T., Saito, T., Wada, J., & Akamatsu, Y. (1997). Unicompartmental arthroplasty for osteoarthritis of the knee using the ceramic ymck model. In *Reconstruction of the knee joint* (pp. 200–206). Springer.

Leardini, A., Chiari, L., Della Croce, U., & Cappozzo, A. (2005). Human movement analysis using stereophotogrammetry: Part 3. soft tissue artifact assessment and compensation. *Gait & posture*, *21*(2), 212–225.

Lofterød, B., Terjesen, T., Skaaret, I., Huse, A.-B., & Jahnsen, R. (2007). Preoperative gait analysis has a substantial effect on orthopedic decision making in children with cerebral palsy: comparison between clinical evaluation and gait analysis in 60 patients. *Acta orthopaedica*, *78*(1), 74–80.

Lonner, J. H. (2016). Robotically assisted unicompartmental knee arthroplasty with a handheld image-free sculpting tool. *Orthopedic Clinics*, *47*(1), 29–40.

Lonner, J. H., & Kerr, G. J. (2012). Robotically assisted unicompartmental knee arthroplasty. *Operative Techniques in Orthopaedics*, *22*(4), 182–188.

Lonner, J. H., Smith, J. R., Picard, F., Hamlin, B., Rowe, P. J., & Riches, P. E. (2015). High degree of accuracy of a novel image-free handheld robot for unicondylar knee arthroplasty in a cadaveric study. *Clinical Orthopaedics and Related Research®*, *473*(1), 206–212.

Lynch, M. (2017). *The fundamentals of cnc.* Retrieved 2017-10-31, from https://www.mmsonline.com/articles/key-cnc-concept-1the-fundamentals-of-cnc

Magee, T. (2018). *Conventional vs. robot-assisted knee replacement — iasis.* Retrieved 2018-02-18, from http://www.utahorthopediccenters.com/conventional-vs-robot-assisted-knee-replacement/

Manaster, B. (1995). Total knee arthroplasty: postoperative radiologic findings. *AJR. American journal of roentgenology*, *165*(4), 899–904.

Maxwell, R., Johnston, A., Lees, D., & Walker, C. (2017). Knee outcome study: A comparison of the patient perceived outcome between high tibial osteotomy, unicompartmental and total knee arthroplasty for medial compartment osteoarthitis in men under age 55. *Orthopaedic journal of sports medicine*, *5*(5_suppl5), 2325967117S00165.

McPherson, A. L., Berry, J. D., Bates, N. A., & Hewett, T. E. (2017). Validity of athletic task performance measures collected with a single-camera motion analysis system as compared to standard clinical measurements. *International journal of sports physical therapy*, *12*(4), 527.

Menache, A. (2000). *Understanding motion capture for computer animation and video games*. Morgan kaufmann.

Moon, Y.-W., Ha, C.-W., Do, K.-H., Kim, C.-Y., Han, J.-H., Na, S.-E., ... PArk, Y.-S. (2012). Comparison of robot-assisted and conventional total knee arthroplasty: A controlled cadaver study using multiparameter quantitative three-dimensional ct assessment of alignment. *Computer Aided Surgery*.

Motesharei, A. (2014). *Investigating the biomechanical outcomes of a robotic-assisted versus conventional unicompartmental knee arthroplasty* (Unpublished doctoral dissertation). University of Strathclyde.

Motive documentation. (2016). Retrieved 2017-10-15, from http://v110.wiki.optitrack.com/index.php?title=Motive_Documentation

Motive:tracker - motion capture and 6 dof object tracking. (2018a). Retrieved from http://optitrack.com/products/motive/tracker

Motive:tracker - motion capture and 6 dof object tracking. (2018b). Retrieved 3-7-2018, from http://optitrack.com/products/motive/tracker/

Mündermann, L., Corazza, S., & Andriacchi, T. P. (2006). The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. *Journal of NeuroEngineering and Rehabilitation*, *3*(1), 6.

Murray, D., Goodfellow, J., & Oconnor, J. (1998). The oxford mediasl unicompartmental arthroplasty: a ten-year survival study. *J Bone Joint Surg Br*, *80*(6), 983–989.

Nanne Kort, J. V. R., Marcus Romanowski. (2018). *Unicompartmental knee arthroplasty: Overview, periprocedural care, technique.* Retrieved 2018-02-18, from http://emedicine.medscape.com/article/1252912-overview

NaturalPoint. (2016). *Devices pane.* Retrieved 2018-04-01, from http://wiki.optitrack.com/index.php?title=Devices_pane\&redirect=no\#Exposure_.28EXP.29

NaturalPoint. (2018a). *Flex 3 an affordable motion capture camera.* Retrieved 2018-03-24, from http://optitrack.com/products/flex-3/

NaturalPoint. (2018b). *Optitrack prime 17w motion capture camera - tracklab.* Retrieved 2018-03-24, from http://tracklab.com.au/products/hardware/optitrack-prime-17w/

NaturalPoint. (2018c). *Prime 13w.* Retrieved 2018-03-24, from http://optitrack.com/products/prime-13w/

NaturalPoint. (2018d). *Slim 13e - a board level camera for computer vision, multi-touch, and more.* Retrieved 2018-03-24, from http://optitrack.com/products/slim-13e/

Nogueira, P. (2011). Motion capture fundamentals: A critical and comparative analysis on real-world applications. In *v zborniku: 4th international conference on information society and technology* (pp. 1–12).

NorthernDigital. (2018). *Polaris series - measurement sciencester vision, multi-touch, and more.* Retrieved 2018-04-9, from `https://www.ndigital.com/msci/products/polaris-series/`

Parvizi, J., Zmistowski, B., Berbari, E. F., Bauer, T. W., Springer, B. D., Della Valle, C. J., ... Zalavras, C. G. (2011). New definition for periprosthetic joint infection: from the workgroup of the musculoskeletal infection society. *Clinical Orthopaedics and Related Research®*, *469*(11), 2992.

Payne, C. (2015). *Ungrounded haptic-feedback for hand-held surgical robots* (Doctoral dissertation, Imperial College London). Retrieved from `http://hdl.handle.net/10044/1/26587`

Pearle, A. D., O'Loughlin, P. F., & Kendoff, D. O. (2010). Robot-assisted unicompartmental knee arthroplasty. *The Journal of arthroplasty*, *25*(2), 230–237.

Perrott, M. A., Pizzari, T., Cook, J., & McClelland, J. A. (2017, feb). Comparison of lower limb and trunk kinematics between markerless and marker-based motion capture systems. *Gait & Posture*, *52*, 57–61. doi: 10.1016/j.gaitpost.2016.10.020

Pfister, A., West, A. M., Bronner, S., & Noah, J. A. (2014). Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis. *Journal of medical engineering & technology*, *38*(5), 274–280.

Picard, F., Moody, J., DiGioia III, A. M., & Jaramaz, B. (2004). Clinical classifications of CAOS systems. *Computer and Robotic Assisted Hip and Knee Surgery*, 43–48.

Piva, S. R., & Klatt, B. A. (2017). An editorial on outcome of unicondylar knee arthroplasty vs total knee arthroplasty for early medial compartment arthritis: a randomized study. *Annals of Joint*, *2*(7).

Plate, J. F., Mofidi, A., Mannava, S., Smith, B. P., Lang, J. E., Poehling, G. G., ... Jinnah, R. H. (2013). Achieving accurate ligament balancing using

robotic-assisted unicompartmental knee arthroplasty. *Advances in orthopedics*, *2013*.

Price, A., Svard, U., Murray, D., & Goodfellow, J. (1999). Ten year survival results of oxford mobile bearing unicompartmental knee arthroplasty in young patients. *ISTA Chicago*.

Psychoyios, V., Crawford, R., Murray, D., & OConnor, J. (1998). Wear of congruent meniscal bearings in unicompartmental knee arthroplasty: a retrieval study of 16 specimens. *J Bone Joint Surg Br*, *80*(6), 976–982.

Purcell, R. L., Cody, J. P., Ammeen, D. J., Goyal, N., & Engh, G. A. (2018). Elimination of preoperative flexion contracture as a contraindication for unicompartmental knee arthroplasty. *JAAOS-Journal of the American Academy of Orthopaedic Surgeons*, *26*(7), e158–e163.

Rahimian, P., & Kearney, J. K. (2017). Optimal camera placement for motion capture systems. *IEEE transactions on visualization and computer graphics*, *23*(3), 1209–1221.

Repicci, J. (2003). Mini-invasive knee unicompartmental arthroplasty: bone-sparing technique. *Surgical technology international*, *11*, 282–286.

Rojas-Lertxundi, S., Fernández-López, J. R., Huerta, S., & García Bringas, P. (2017). Motion capture systems for jump analysis. *Logic Journal of the IGPL*, *25*(6), 890–901.

Sharkey, P. F., Hozack, W. J., Rothman, R. H., Shastri, S., & Jacoby, S. M. (2002). Why are total knee arthroplasties failing today? *Clinical Orthopaedics and Related Research®*, *404*, 7–13.

Sharkey, P. F., Lichstein, P. M., Shen, C., Tokarski, A. T., & Parvizi, J. (2014). Why are total knee arthroplasties failing todayhas anything changed after 10 years? *The Journal of arthroplasty*, *29*(9), 1774–1778.

Sharma, A., Agarwal, M., Sharma, A., & Dhuria, P. (2013). Motion capture process, techniques and applications. *Int. J. Recent Innov. Trends Comput. Commun*, *1*, 251–257.

Shiratori, T., Park, H. S., Sigal, L., Sheikh, Y., & Hodgins, J. K. (2011, jul). Motion capture from body-mounted cameras. *ACM Transactions on Graphics*,

30(4), 1. doi: 10.1145/2010324.1964926

SMARTTECH. (2016a). *scan3dmed - smarttech 3d scanner.* Retrieved 2018-03-24, from http://smarttech3dscanner.com/3d-scanners/for-medicine/scan3dmed/

SMARTTECH. (2016b). *Smarttech 3d — optical measurement systems — portable 3d scanner.* Retrieved 2018-03-24, from http://smarttech3dscanner.com/3d-scanners/smarttech-3d-portable-3d-scanner/

SMARTTECH_Co.Ltd. (2018). *Scan3d med - 3d scanners for medical application and much more.* Retrieved 2018-12-02, from http://smarttech3dscanner.com/wp-content/uploads/2016/06/SMARTTECH_scan3Dmed_web.pdf

Smith, J. R., Riches, P. E., & Rowe, P. J. (2014). Accuracy of a freehand sculpting tool for unicondylar knee replacement. *The International Journal of Medical Robotics and Computer Assisted Surgery*, *10*(2), 162–169. Retrieved from http://dx.doi.org/10.1002/rcs.1522 doi: 10.1002/rcs.1522

Specht, L. M., & Koval, K. J. (2001). Robotics and computer-assisted orthopaedic surgery. *Bulletin of the Hospital for Joint Diseases Orthopaedic Institute*, *60*(3-4), 168-172.

Stryker.com. (2018). *Mako partial knee.* Retrieved 2018-02-18, from https://www.stryker.com/us/en/joint-replacement/systems/mako-partial-knee.html

Surgical burs and drill system. (2016). Retrieved 2018-02-18, from https://www.merciansurgical.com/products/surgical-burs-and-drill-system/

Thewlis, D., Bishop, C., Daniell, N., & Paul, G. (2011). *A comparison of two commercially available motion capture systems for gait analysis: High end vs low-cost.*

Treleaven, P., & Wells, J. (2007). 3d body scanning and healthcare applications. *Computer*, *40*(7).

Unal, G., Yezzi, A., Soatto, S., & Slabaugh, G. (2007). A variational approach to

problems in calibration of multiple cameras. *IEEE transactions on pattern analysis and machine intelligence*, *29*(8), 1322–1338.

USA, S. E. M. (2017). Nema size 23 1.8 2-phase stepper motor [Computer software manual]. Retrieved 28-08-2017, from `https://motion.schneider-electric.com/hybrid-stepper-motor/m-23-nema-23-3-0-1-8-stepper-motor/`

van der Esch, M., Knol, D. L., Schaffers, I. C., Reiding, D. J., van Schaardenburg, D., Knoop, J., ... Dekker, J. (2013). Osteoarthritis of the knee: multicompartmental or compartmental disease? *Rheumatology*, *53*(3), 540–546.

Van der List, J. P., Chawla, H., & Pearle, A. D. (2016). Robotic-assisted knee arthroplasty: an overview. *Am J Orthop*, *45*(4), 202.

Vicon. (2016). *About vicon motion systems.* Retrieved 2018-03-24, from `https://www.vicon.com/vicon/about`

Wawro, A. (2016). *Yu suzuki recalls using military tech to make virtua fighter 2.* Gamasutra.

Weyrich, T., Pauly, M., Keiser, R., Heinzle, S., Scandella, S., & Gross, M. H. (2004). Post-processing of scanned 3d surface data. *SPBG*, *4*, 85–94.

What is motion capture. (2017). VICON. Retrieved 2017-09-13, from `https://www.vicon.com/what-is-motion-capture`

Windolf, M., Götzen, N., & Morlock, M. (2008). Systematic accuracy and precision analysis of video motion capturing systemsexemplified on the vicon-460 system. *Journal of biomechanics*, *41*(12), 2776–2780.

Yabukami, S., Kikuchi, H., Yamaguchi, M., Arai, K., Takahashi, K., Itagaki, A., & Wako, N. (2000). Motion capture system of magnetic markers using three-axial magnetic field sensor. *IEEE transactions on magnetics*, *36*(5), 3646–3648.

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, *19*(2), 4–10.

ZimmerBiomet. (2018). *Oxford partial knee.* Retrieved 2018-02-18, from `http://www.oxfordpartialknee.net/emea`

Appendix A

Codes

A.1 Getting Cluster Definition Data Application

```
--Functions
--Euclidean Distance
function getEuclideanDistance(p1, p2)
return math.sqrt((p1["x"]-p2["x"])^2 + (p1["y"]-p2["y"])
   ↪ ^2 +
(p1["z"]-p2["z"])^2)
end
--Sorting
function sortArrayAssending(A)
for i = 1, expectedNoOfCombinations do
for j = i+1, expectedNoOfCombinations do
if A[i][1] > A[j][1] then
temp = A[j]
A[j] = A[i]
A[i] = temp
end
end
```

```
end
return A
end
--Printing:
function printArray(A)
--print(table.getn(A))
x = 30
if table.getn(A) < x then
x = table.getn(A)
end
for i = 1, x do--table.getn(A) do
print(A[i][1], A[i][2], A[i][3])
end
end

function printMarkersArray(A)
--print(table.getn(A))
for i = 1, table.getn(A) do
print(A[i]["x"], A[i]["y"], A[i]["z"])
end
end

function printSimpleArray(A)
--print(table.getn(A))
for i = 1, table.getn(A) do
print(A[i])
end
end
-----------------------------------------------------

--ScalingFactor variable controls all the inputs scaling
```

```
   ↪ factor
toleranceFactor = toleranceFactor or 0
--init variables
--Set this variable with the total number of markers in
   ↪ all segments
noOfMarkers = noOfMarkers or 3
init = init or 0
allinputs = allinputs or {}
allOuts = allOuts or {}
--Scalling Factor for the distance unit control
--The forbidden number is the number that indicats nil
   ↪ value in a channel
theForbiddenNumber = theForbiddenNumber or 0
expectedNoOfCombinations = expectedNoOfCombinations or
noOfMarkers * (noOfMarkers - 1) / 2 --fact(noOfMarkers)/(
   ↪ fact(noOfMarkers-2) * 2)  --Combinational Logic
eculideanDistanceArray = eculideanDistanceArray or {}
input = input or {}
markers = markers or {}
--DIM's in CM
TibialArrayDims = TibialArrayDims or {}
FemoralArrayDims = FemoralArrayDims or {}
EndEffectorArrayDims = EndEffectorArrayDims or {}
BaseArrayDims = BaseArrayDims or {}
BlueProbeDims = BlueProbeDims or {}
GreenProbeDims = GreenProbeDims or {}
tibialArray = tibialArray or {}
femoralArray = femoralArray or {}
endEffectorArray = endEffectorArray or {}
baseArray = baseArray or {}
blueProbe = blueProbe or {}
```

```
greenProbe = greenProbe or {}
--Separating segments from each other by steps, each is
   ↪ the expected dims
tibial = table.getn(TibialArrayDims)
femoral = tibial + table.getn(FemoralArrayDims)
endEffector = femoral + table.getn(EndEffectorArrayDims)
base = endEffector + table.getn(BaseArrayDims)
blue = base + table.getn(BlueProbeDims)
green = blue + table.getn(GreenProbeDims)

--init the code
if init == 0 then
for i = 1, (noOfMarkers * 3) do
allinputs[i] = "Channel"..i
end
counter = 1
i = 1
while i < (tibial * 3) + 1 do
allOuts[i] = "Tibial␣Array␣"..counter.."␣X"
allOuts[i+1] = "Tibial␣Array␣"..counter.."␣Y"
allOuts[i+2] = "Tibial␣Array␣"..counter.."␣Z"
i = i+3
counter = counter + 1
end
counter = 1
i = (tibial * 3) + 1
while i < (femoral * 3) + 1 do
allOuts[i] = "Femoral␣Array␣"..counter.."␣X"
allOuts[i+1] = "Femoral␣Array␣"..counter.."␣Y"
allOuts[i+2] = "Femoral␣Array␣"..counter.."␣Z"
i = i+3
```

```
counter = counter + 1
end
counter = 1
i = (femoral * 3) + 1
while i < (endEffector * 3) + 1 do
allOuts[i] = "EndEffector Array "..counter.." X"
allOuts[i+1] = "EndEffector Array "..counter.." Y"
allOuts[i+2] = "EndEffector Array "..counter.." Z"
i = i+3
counter = counter + 1
end
counter = 1
i = (endEffector * 3) + 1
while i < (base * 3) + 1 do
allOuts[i] = "Base Array "..counter.." X"
allOuts[i+1] = "Base Array "..counter.." Y"
allOuts[i+2] = "Base Array "..counter.." Z"
i = i+3
counter = counter + 1
end
counter = 1
i = (base * 3) + 1
while i < (blue * 3) + 1 do
allOuts[i] = "Blue Probe "..counter.." X"
allOuts[i+1] = "Blue Probe "..counter.." Y"
allOuts[i+2] = "Blue Probe "..counter.." Z"
i = i+3
counter = counter + 1
end
counter = 1
i = (blue * 3) + 1
```

```
while i < (green * 3) + 1 do
allOuts[i] = "Green Probe "..counter.." X"
allOuts[i+1] = "Green Probe "..counter.." Y"
allOuts[i+2] = "Green Probe "..counter.." Z"
i = i+3
counter = counter + 1
end

inputs.setchannels(unpack(allinputs))
outputs.setchannels(unpack(allOuts))
init = 1
end
i = 1
j = 1
while i < (noOfMarkers * 3) + 1 do
markers[j] = {}
markers[j]["x"] = inputs.get("Channel"..i)
markers[j]["y"] = inputs.get("Channel"..i+1)
markers[j]["z"] = inputs.get("Channel"..i+2)
i = i + 3
j = j + 1
end



--processing
x = 1
y = 2
for i = 1, expectedNoOfCombinations do
--print(i, x, y)
eculideanDistanceArray[i] = {getEuclideanDistance(markers
   ↪ [x], markers[y]), x , y}
```

```
if y == noOfMarkers then
x = x+1
y = x+1
else
y = y+1
end
end

eculideanDistanceArray =
sortArrayAssending(eculideanDistanceArray)

---[[
print("Start")
print(expectedNoOfCombinations)
printArray(eculideanDistanceArray)
print("Finish.....")
```

A.2 Translating Script from the Scanner Application

```
--Euclidean Distance
function getEuclideanDistance(p1, p2)
return math.sqrt((p1["x"]-p2["x"])^2 + (p1["y"]-p2["y"])
   ↪ ^2 + (p1["z"]-p2["z"])^2)
end
--Sorting
function sortArrayAssending(A)
for i = 1, expectedNoOfCombinations do
for j = i+1, expectedNoOfCombinations do
if A[i][1] > A[j][1] then
temp = A[j]
```

```
A[j] = A[i]
A[i] = temp
end
end
end
return A
end
--Printing:
function printArray(A)
--print(table.getn(A))
x = 30
if table.getn(A) < x then
x = table.getn(A)
end
for i = 1, x do
print(i,A[i][1], A[i][2], A[i][3])
end
print("-------------------------------------")
end
function getAVG(point1, point2)
a = {}
a["x"] = (point1["x"] + point2["x"]) / 2.0
a["y"] = (point1["y"] + point2["y"]) / 2.0
a["z"] = (point1["z"] + point2["z"]) / 2.0
return a
end
function getVector(point1, point2)
vector = {}
vector["x"] = point1["x"] - point2["x"]
vector["y"] = point1["y"] - point2["y"]
vector["z"] = point1["z"] - point2["z"]
```

```
return vector
end
function getUNIVector(vector, magV)
vector["x"] = vector["x"]/magV
vector["y"] = vector["y"]/magV
vector["z"] = vector["z"]/magV
return vector
end
function crossProduct(vector1, vector2)
--[[
cx = aybz - azby
cy = azbx - axbz
cz = axby - aybx
--]]
vector = {}
vector["x"] = vector1["y"] * vector2["z"] - vector1["z"]
   ↪ * vector2["y"]
vector["y"] = vector1["z"] * vector2["x"] - vector1["x"]
   ↪ * vector2["z"]
vector["z"] = vector1["x"] * vector2["y"] - vector1["y"]
   ↪ * vector2["x"]
return vector
end
function getMagnitude(point)
vector = (point["x"]^2 + point["y"]^2 + point["z"]^2)^0.5
return vector
end
function transform(aP, bP, cP, fourthMarker)
tfm = {}
tfm["x"] = aP["x"] * fourthMarker["x"] + aP["y"] *
   ↪ fourthMarker["y"] + aP["z"] * fourthMarker["z"]
```

```
tfm["y"] = bP["x"] * fourthMarker["x"] + bP["y"] *
   ↪ fourthMarker["y"] + bP["z"] * fourthMarker["z"]
tfm["z"] = cP["x"] * fourthMarker["x"] + cP["y"] *
   ↪ fourthMarker["y"] + cP["z"] * fourthMarker["z"]
return tfm
end
function transformTranspose(aP, bP, cP, fourthMarker)
tfm = {}
tfm["x"] = aP["x"] * fourthMarker["x"] + bP["x"] *
   ↪ fourthMarker["y"] + cP["x"] * fourthMarker["z"]
tfm["y"] = aP["y"] * fourthMarker["x"] + bP["y"] *
   ↪ fourthMarker["y"] + cP["y"] * fourthMarker["z"]
tfm["z"] = aP["z"] * fourthMarker["x"] + bP["z"] *
   ↪ fourthMarker["y"] + cP["z"] * fourthMarker["z"]
return tfm
end
function setMarkers()  --to number each marker in the
   ↪ segment
--printArray(eculideanDistanceArray)
if (eculideanDistanceArray[1][1] + toleranceFactor >=
   ↪ FemoralArrayDims[1] and (eculideanDistanceArray
   ↪ [1][1] - toleranceFactor <= FemoralArrayDims[1]))
   ↪ and
(eculideanDistanceArray[3][1] + toleranceFactor >=
   ↪ FemoralArrayDims[3] and (eculideanDistanceArray
   ↪ [3][1] - toleranceFactor <= FemoralArrayDims[3]))
   ↪ and
(eculideanDistanceArray[5][1] + toleranceFactor >=
   ↪ FemoralArrayDims[5] and (eculideanDistanceArray
   ↪ [5][1] - toleranceFactor <= FemoralArrayDims[5]))
   ↪ then
```

```
if femoralArray[eculideanDistanceArray[1][2]] ==
   ↪ femoralArray[eculideanDistanceArray[3][2]] then
noTwo = femoralArray[eculideanDistanceArray[1][2]]
noOne = femoralArray[eculideanDistanceArray[1][3]]
noThree = femoralArray[eculideanDistanceArray[3][3]]
elseif femoralArray[eculideanDistanceArray[1][2]] ==
   ↪ femoralArray[eculideanDistanceArray[3][3]] then
noTwo = femoralArray[eculideanDistanceArray[1][2]]
noOne = femoralArray[eculideanDistanceArray[1][3]]
noThree = femoralArray[eculideanDistanceArray[3][2]]
elseif femoralArray[eculideanDistanceArray[1][3]] ==
   ↪ femoralArray[eculideanDistanceArray[3][2]] then
noTwo = femoralArray[eculideanDistanceArray[1][3]]
noOne = femoralArray[eculideanDistanceArray[1][2]]
noThree = femoralArray[eculideanDistanceArray[3][3]]
elseif femoralArray[eculideanDistanceArray[1][2]] ==
   ↪ femoralArray[eculideanDistanceArray[3][2]] then
noTwo = femoralArray[eculideanDistanceArray[1][2]]
noOne = femoralArray[eculideanDistanceArray[1][3]]
noThree = femoralArray[eculideanDistanceArray[3][3]]
end
else
print("SETTING␣MARKERS␣NOT␣VALID!!!")
return
end
print("Markers␣Set...")
end
function transformTipLocation()
setMarkers()
localOrigin = getAVG(noOne, noTwo)
a = getVector(noThree, localOrigin)
```

```
bt = getVector(noOne, localOrigin)
c = crossProduct(a, bt)
b = crossProduct(a, c)

magA = getMagnitude(a)
magB = getMagnitude(b)
magC = getMagnitude(c)
--printVector(localOrigin)
--[[
printVector(a)
printVector(b)
printVector(c)
print(magA)
print(magC)
print(magB)
--]]
aUNI = getUNIVector(a, magA)
bUNI = getUNIVector(b, magB)
cUNI = getUNIVector(c, magC)
--tipLocation = transformTranspose(aUNI, bUNI, cUNI,
   ↪ tipLocation)
tipLocation = transform(aUNI, bUNI, cUNI, tipLocation)
tipLocation["x"] = tipLocation["x"] + localOrigin["x"]
tipLocation["y"] = tipLocation["y"] + localOrigin["y"]
tipLocation["z"] = tipLocation["z"] + localOrigin["z"]
return tipLocation

end
--Variables------------------------------
noOfMarkers = noOfMarkers or 0
expectedNoOfCombinations = expectedNoOfCombinations or 0
```

```
eculideanDistanceArray = eculideanDistanceArray or {}
toleranceFactor = toleranceFactor or 0
theAvoidNumber = theAvoidNumber or 0
femoralArray = femoralArray or {}
FemoralArrayDims = FemoralArrayDims or {}
expectedNoOfCombinations = expectedNoOfCombinations or 0
eculideanDistanceArray = eculideanDistanceArray or {}
drawMarkers = drawMarkers or {}
markerSize = markerSize or 0.05
allIns = allIns or {}
allOuts = allOuts or {}
fileCount = fileCount or 1
noOne = noOne or 0
noTwo = noTwo or 0
noThree = noThree or 0
previousFootSwitchState = previousFootSwitchState or 0.5
init = init or 0
--Initialization-------------------------
if init == 0 then
expectedNoOfCombinations = 4 * (4 - 1) / 2
allIns[1] = "Number␣of␣Markers"
allIns[2] = "Tolerance␣Factor"
allIns[3] = "The␣Avoid␣Number"
for i = 4, 9 do
allIns[i] = "FemoralArrayDims␣"..i-3
end
inputs.setchannels(unpack(allIns))
noOfMarkers = inputs.get("Number␣of␣Markers")
toleranceFactor = inputs.get("Tolerance␣Factor")
theAvoidNumber = inputs.get("The␣Avoid␣Number")
for i = 1, 6 do
```

```
FemoralArrayDims[i] = inputs.get("FemoralArrayDims␣"..i)
end
i = 10
counter = 1
while i < (4 * 3) + 10 do
allIns[i] = "Femoral␣Marker␣"..counter.."␣X"
allIns[i+1] = "Femoral␣Marker␣"..counter.."␣Y"
allIns[i+2] = "Femoral␣Marker␣"..counter.."␣Z"
i = i+3
counter = counter + 1
end
allIns[i] = "GreenProbe␣Tip␣X"
allIns[i+1] = "GreenProbe␣Tip␣Y"
allIns[i+2] = "GreenProbe␣Tip␣Z"
i = i+3
allIns[i] = "Foot␣Switch"
allIns[i+1] = "Femur␣Sitiuation"
allIns[i+2] = "Green␣Sitiuation"
i = i+2
inputs.setchannels(unpack(allIns))
outputs.setchannels(unpack(allOuts))
outFileName='C:\\CAREN␣Resources\\data\\Pointer\\Pointer'
   ↪ ..fileCount..'.txt'
f1 = io.output(outFileName)
outFileName='C:\\CAREN␣Resources\\data\\Pointer\\Pointer␣
   ↪ Global'..fileCount..'.txt'
f3 = io.output(outFileName)
init = 1
end
print(inputs.get("Foot␣Switch"))
if inputs.get("Foot␣Switch") < 0.5 and inputs.get("Femur␣
```

```
   ↪ Sitiuation") == 1 and inputs.get("Green Sitiuation"
   ↪ ) == 1 then
for i = 1, 4 do
femoralArray[i] = {}
femoralArray[i]["x"] = inputs.get("Femoral Marker "..i.."
   ↪ X")
femoralArray[i]["y"] = inputs.get("Femoral Marker "..i.."
   ↪ Y")
femoralArray[i]["z"] = inputs.get("Femoral Marker "..i.."
   ↪ Z")
end
x = 1
y = 2
--print(table.getn(femoralArray))
for i = 1, expectedNoOfCombinations do
--print(i, x, y)
eculideanDistanceArray[i] = {getEuclideanDistance(
   ↪ femoralArray[x], femoralArray[y]), x , y}
if y == 4 then
x = x+1
y = x+1
else
y = y+1
end
end
eculideanDistanceArray = sortArrayAssending(
   ↪ eculideanDistanceArray)
tipLocation = {}
tipLocation["x"] = inputs.get("GreenProbe Tip X")
tipLocation["y"] = inputs.get("GreenProbe Tip Y")
tipLocation["z"] = inputs.get("GreenProbe Tip Z")
```

```
f3:write(tipLocation["x"],";",tipLocation["y"],";",
   ↪ tipLocation["z"],"\n")
print(inputs.get("GreenProbe Tip X"),"\t",tipLocation["y"
   ↪ ],"\t",tipLocation["z"],"\n")
tipLocation = transformTipLocation(tipLocation)
f1:write(tipLocation["x"],";",tipLocation["y"],";",
   ↪ tipLocation["z"],"\n")
print(tipLocation["x"],"\t",tipLocation["y"],"\t",
   ↪ tipLocation["z"],"\n")
elseif inputs.get("Femur Sitiuation") == 2 then
print("Femur Not Found!")
elseif inputs.get("Green Sitiuation") == 2 then
print("Green Probe Not Found!")
end
if previousFootSwitchState < 0.5 and inputs.get("Foot
   ↪ Switch") > 0.5  then
fileCount = fileCount+1
f1:close()
f3:close()
outFileName='C:\\CAREN Resources\\data\\Pointer\\Pointer'
   ↪ ..fileCount..'.txt'
f1 = io.output(outFileName)
outFileName='C:\\CAREN Resources\\data\\Pointer\\Pointer
   ↪ Global'..fileCount..'.txt'
f3 = io.output(outFileName)
end
previousFootSwitchState = inputs.get("Foot Switch")
```

Appendix B

Clusters Identification Data

B.1 Tibia Cluster

0.051857

0.060581

0.067941

0.074713

0.097268

0.127427

B.2 Blunt Probe

0.058216

0.096548

0.10534

B.3 Sharp Probe

0.052109

0.106377

0.125720

B.4 Femur Cluster

0.05609

0.065675

0.076599

0.084969

0.095304

0.106154

B.5 Testing Accuracy of the System Application

```
--Function-----------------------------------------
function setFile(segmentNumber, layer)
outfname="C:\textbackslash\{\}\textbackslash\{\}CAREN␣
   ↪ Resources\textbackslash\{\}\textbackslash\{\}Data\
   ↪ textbackslash\{\}\textbackslash\{\}My␣Saved␣Data\
   ↪ textbackslash\{\}\textbackslash\{\}Segment␣"
   ↪ ..segmentNumber.."␣-␣Layer␣"..layer..".txt"
io.output(outfname)
io.write("\#\textbackslash\{\}tX\textbackslash\{\}tY\
   ↪ textbackslash\{\}tZ\textbackslash\{\}n")

end
--init variables-----------------------------------
init = init or 0
allinputs = allinputs or \{\}
theForbiddenNumber = 99999.44444
counter = counter or 0
maxCounts = 9
segmentNumber = segmentNumber or 1
```

```
layer = layer or 1
maxLayers = 3
loops = loops or 1


--init the code
if init == 0 then
setFile(segmentNumber, layer)

notification = sound.create("C:/CAREN␣Resources/Sounds/
   ↪ beep-06.wav")
sound.setvolume(notification,100)
segComp = sound.create("C:/CAREN␣Resources/Sounds/Segment
   ↪ \_Complete.wav")
sound.setvolume(segComp,100)



switchprevious=inputs.get("Channel␣4")
switchready=1
for i = 1, 4 do
allinputs[i] = "Channel␣"..i

end
inputs.setchannels(unpack(allinputs))
init = 1
end

--[[
theTip["x"] = inputs.get("Channel 1")
theTip["y"] = inputs.get("Channel 2")
```

```
theTip["z"] = inputs.get("Channel 3")
--]]

if(inputs.get("Channel_4") == 1 and counter < maxCounts
   ↪ and switchready==1) then
counter = counter + 1
io.write(counter.."\textbackslash\{\}t"..inputs.get("
   ↪ Channel_1").."\textbackslash\{\}t"..inputs.get("
   ↪ Channel_2").."\textbackslash\{\}t"..inputs.get("
   ↪ Channel_3").."\textbackslash\{\}n")
sound.play(notification)
switchready=0
end

switchnow=inputs.get("Channel_4")

if switchprevious==1 and switchnow==0 then
switchready=1
end

switchprevious=switchnow

if counter == maxCounts then
io.flush()
counter = 0
segmentNumber = segmentNumber + 1
sound.play(segComp)
setFile(segmentNumber, layer)
if segmentNumber == 10 then
segmentNumber = 1
layer = layer + 1
```

```
end
end
```

Appendix C

Arduino Application

```
1 #include <LiquidCrystal.h>
2 #include <AccelStepper.h>
3 #include <SPI.h>
4 #include <WiFi.h>
5
6 IPAddress ip(192,168,2,100);
7 //----IPAddress ip(192,168,0,100);
8 //char ssid[] = "VM567432-2G"; //  your network SSID (
    ↪ name)
9 //char pass[] = "brjwegfu";    // your network password (
    ↪ use for WPA, or use as key for WEP)
10 char ssid[] = "CROSS"; //  your network SSID (name)
11 char pass[] = "11111111";//"********@strath.ac.uk";    //
    ↪  your network password (use for WPA, or use as key
    ↪ for WEP)
12
13
14 int keyIndex = 0;            // your network key Index
    ↪ number (needed only for WEP)
15
16 int status = WL_IDLE_STATUS;
```

```
17 WiFiServer server(80);
18 boolean alreadyConnected = false; // whether or not the
     ↪ client was connected previously
19
20 // initialize the library with the numbers of the
     ↪ interface pins
21 LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
22 const int RESET = 50;
23 const int JOYSTICK1_VRX = A0;
24 const int JOYSTICK1_VRY = A1;
25 const int JOYSTICK1_SW = A2;
26 const int JOYSTICK2_VRX = A3;
27 const int JOYSTICK2_VRY = A4;
28 const int JOYSTICK2_SW = A5;
29 const int MOTOR_M_DIRECTION = 30;
30 const int MOTOR_M_PULL = 31;
31 const int MOTOR_Y1_DIRECTION = 32;
32 const int MOTOR_Y1_PULL = 33;
33 const int MOTOR_Y2_DIRECTION = 34;
34 const int MOTOR_Y2_PULL = 35;
35 const int MOTOR_X1_DIRECTION = 36;
36 const int MOTOR_X1_PULL = 37;
37 const int MOTOR_X2_DIRECTION = 38;
38 const int MOTOR_X2_PULL = 39;
39 const int MAX_SPEED = 3000;
40 const int MAX_ACCELERATION = 2500;
41 const int COMMON_DELAY = 1;
42 String directions[6];
43 AccelStepper stepperM(1, MOTOR_M_PULL, MOTOR_M_DIRECTION)
     ↪ ;
44 AccelStepper stepperY1(1, MOTOR_Y1_PULL,
```

```
    ↪ MOTOR_Y1_DIRECTION);
45 AccelStepper stepperY2(1, MOTOR_Y2_PULL,
    ↪ MOTOR_Y2_DIRECTION);
46 AccelStepper stepperZ1(1, MOTOR_X1_PULL,
    ↪ MOTOR_X1_DIRECTION);
47 AccelStepper stepperZ2(1, MOTOR_X2_PULL,
    ↪ MOTOR_X2_DIRECTION);
48 int mode; // 0 = welcome - 1 = Network - 2 = Manual
49 float x,y,z;
50 WiFiClient client;
51 String networkMode;
52 bool printNetworkStatus;
53 void setup()
54 {
55 // initialize serial communications at 9600 bps:
56 Serial.begin(9600);
57 lcd.begin(16, 2);
58 lcd.print("Welcome...");
59 // check for the presence of the shield:
60 if (WiFi.status() == WL_NO_SHIELD)
61 {
62 lcd.setCursor(0, 1);
63 lcd.print("Shield␣problem!");
64 Serial.println("Shield␣problem!");
65 // don't continue:
66 while (true);
67 }
68 WiFi.config(ip);
69 stepperM.setMaxSpeed(MAX_SPEED);
70 stepperM.setSpeed(MAX_SPEED);
71 stepperM.setAcceleration(MAX_ACCELERATION);
```

```
72 stepperY1.setMaxSpeed(MAX_SPEED);
73 stepperY1.setSpeed(MAX_SPEED);
74 stepperY1.setAcceleration(MAX_ACCELERATION);
75 stepperY2.setMaxSpeed(MAX_SPEED);
76 stepperY2.setSpeed(MAX_SPEED);
77 stepperY2.setAcceleration(MAX_ACCELERATION);
78 stepperZ1.setMaxSpeed(MAX_SPEED);
79 stepperZ1.setSpeed(MAX_SPEED);
80 stepperZ1.setAcceleration(MAX_ACCELERATION);
81 stepperZ2.setMaxSpeed(MAX_SPEED);
82 stepperZ2.setSpeed(MAX_SPEED);
83 stepperZ2.setAcceleration(MAX_ACCELERATION);
84 mode = 0;
85 printNetworkStatus = false;
86 x = 0;
87 y = 0;
88 z = 0;
89 pinMode(RESET, INPUT_PULLUP);
90 }
91 //The Joysticks: VRX (UP = Max, Down = 0) VRY (RIGHT =
     ↪ Max, LEFT = 0) SW (PRESS = 0, ELSE !0)
92
93
94
95 void loop()
96 {
97 if(mode == 0)
98 {
99
100 // set the cursor to column 0, line 1
101 // (note: line 1 is the second row, since counting begins
```

```
    ↪  with 0):
lcd.setCursor(0, 1);
lcd.print("Manual<.>Network");
getReadings();
if(directions[2] == "PRESSED")
{
mode = 2;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("MANUAL...");
Serial.println("MANUAL...");
delay(200);
}
else if(directions[5] == "PRESSED")
{
mode = 1;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Network...");
Serial.println("Network...");
}
delay(50);
}
else if(mode == 1) //Network
{
getReadings();
if(!alreadyConnected)
{
//Networking Again:
while ( status != WL_CONNECTED)
{
```

```
132 lcd.setCursor(0, 1);
133 lcd.print("Connecting:");
134 lcd.print(ssid);
135 Serial.print("Connecting:␣");
136 Serial.println(ssid);
137 // Connect to WPA/WPA2 network. Change this line if using
      ↪ open or WEP network:
138 status = WiFi.begin(ssid, pass);
139 // wait 10 seconds for connection:
140 delay(3000);
141
142
143 // you're connected now, so print out the status:
144 printWifiStatus();
145 }
146 if(status == WL_CONNECTED && !printNetworkStatus)
147 {
148 lcd.clear();
149 lcd.setCursor(0, 0);
150 lcd.print("Network...");
151 lcd.setCursor(0, 1);
152 lcd.print("Connected");
153 printNetworkStatus = true;
154 }
155
156 // start the server:
157 server.begin();
158 client = server.available();
159 if(client.connected())
160 {
161 lcd.setCursor(0, 1);
```

```
lcd.print("Client Connected");
alreadyConnected = true;
networkMode = "starting";
}
}
if (client)
{
if(networkMode == "starting")
{
networkMode = getNetworkReading();
}
networkMode.trim();
if(networkMode.equals("Manual"))
{
String in = client.readString();
in.trim();
// echo the bytes back to the client:
if(in == "up")
{
directions[4] = "RIGHT";
}
else if(in == "down")
{
directions[4] = "LEFT";
}
else if(in == "left")
{
directions[1] = "LEFT";
}
else if(in == "right")
{
```

```
193 directions[1] = "RIGHT";
194 }
195 else if(in == "in")
196 {
197 directions[4] = "LEFT";
198 }
199 else if(in == "out")
200 {
201 directions[4] = "RIGHT";
202 }
203 else if(in == "idle")
204 {
205 directions[1] = "Nah";
206 directions[0] = "Nah";
207 directions[4] = "Nah";
208 }
209 runIt();
210 }
211 else if(networkMode == "Automatic")
212 {
213 char in[60];
214 String data = getNetworkReading();
215 data.toCharArray(in,60);
216 data.trim();
217 if(data == "HALT")
218 {
219 reset();
220 Serial.println("HALTTED!!!");
221 }
222 else
223 {
```

```
224 int i = 0;
225 for(i; i < data.length();i++)
226 {
227 if(data.substring(i,i+1) == ";")
228 {
229 x = data.substring(0,i).toFloat();
230 break;
231 }
232 }
233 i +=1;
234 int b = i;
235 for(i; i < data.length();i++)
236 {
237 if(data.substring(i,i+1) == ";")
238 {
239 y = data.substring(b,i).toFloat();
240 break;
241 }
242 }
243 i +=1;
244 z = data.substring(i,data.length()).toFloat();
245 stepperM.moveTo(cmToTurns(x*100));
246 stepperZ1.moveTo(cmToTurns(z*100));
247 stepperZ2.moveTo(cmToTurns(z*100));
248 stepperY1.moveTo(cmToTurns(y*-100));
249 stepperY2.moveTo(cmToTurns(y*-100));
250 while(stepperM.distanceToGo() != 0 || stepperZ1.
    ↪ distanceToGo() != 0 || stepperY1.distanceToGo() !=
    ↪ 0)
251 {
252 delay(COMMON_DELAY);
```

```
253 stepperM.run();
254 stepperZ1.run();
255 stepperZ2.run();
256 stepperY1.run();
257 stepperY2.run();
258 }
259 delay(10);
260 }
261 }
262 }
263 else if(alreadyConnected)
264 {
265 lcd.setCursor(0, 1);
266 lcd.print("Client␣disConnected");
267 reset();
268 }
269 }
270 else //Manual
271 {
272 getReadings();
273 runIt();
274 delay(COMMON_DELAY);
275 }
276 }
277 String getNetworkReading()
278 {
279 int counter = 0;
280 char temp[60];
281 String reading = "";
282 while(!client.available())
283 {
```

```
284 delay(5);
285 }
286 String networkMsg = client.readStringUntil('\n');
287 while(reading != "OK")
288 {
289 if(reading != "OK" && !reading.equals("\0") && reading !=
      ↪  0)
290 networkMsg = reading;
291 networkMsg.toCharArray(temp,60);
292 client.write(temp);
293 while(!client.available())
294 {
295 delay(5);
296 }
297 reading = client.readStringUntil('\n');
298 reading.trim();
299 }
300 return networkMsg;
301 }
302 void getReadings()
303 {
304 if(checkReset())
305 return;
306 //Joystick 1 >>
307 if(analogRead(JOYSTICK1_VRX) > 790)
308 directions[0] = "DOWN";
309 else if(analogRead(JOYSTICK1_VRX) < 720)
310 directions[0] = "UP";
311 else
312 directions[0] = "NAH";
313
```

```
314 if(analogRead(JOYSTICK1_VRY) > 790)
315 directions[1] = "LEFT";
316 else if(analogRead(JOYSTICK1_VRY) < 720)
317 directions[1] = "RIGHT";
318 else
319 directions[1] = "NAH";
320
321 if(analogRead(JOYSTICK1_SW) == 0)
322 directions[2] = "PRESSED";
323 else
324 directions[2] = "NAH";
325 //Joystick 2 >>
326 if(analogRead(JOYSTICK2_VRX) > 790)
327 directions[3] = "DOWN";
328 else if(analogRead(JOYSTICK2_VRX) < 720)
329 directions[3] = "UP";
330 else
331 directions[3] = "NAH";
332
333 if(analogRead(JOYSTICK2_VRY) > 790)
334 directions[4] = "LEFT";
335 else if(analogRead(JOYSTICK2_VRY) < 720)
336 directions[4] = "RIGHT";
337 else
338 directions[4] = "NAH";
339
340 if(analogRead(JOYSTICK2_SW) == 0)
341 directions[5] = "PRESSED";
342 else
343 directions[5] = "NAH";
344 if(directions[2] == "PRESSED" && directions[5] == "
```

```
    ↪ PRESSED")
345 {
346 mode = 0;
347 delay(300);
348 }
349 }
350
351 void runIt()
352 {
353 if(directions[0] == "UP")
354 {
355 y-=1;
356 stepperY1.setSpeed(-MAX_SPEED);
357 stepperY2.setSpeed(-MAX_SPEED);
358 stepperY1.runSpeed();
359 stepperY2.runSpeed();
360 }
361 else if(directions[0] == "DOWN")
362 {
363 y+=1;
364 stepperY1.setSpeed(MAX_SPEED);
365 stepperY2.setSpeed(MAX_SPEED);
366 stepperY1.runSpeed();
367 stepperY2.runSpeed();
368 }
369 if(directions[1] == "LEFT")
370 {
371 z-=1;
372 stepperZ1.setSpeed(-MAX_SPEED);
373 stepperZ2.setSpeed(-MAX_SPEED);
374 stepperZ1.runSpeed();
```

```
375 stepperZ2.runSpeed();
376 }
377 else if(directions[1] == "RIGHT")
378 {
379 z+=1;
380 stepperZ1.setSpeed(MAX_SPEED);
381 stepperZ2.setSpeed(MAX_SPEED);
382 stepperZ1.runSpeed();
383 stepperZ2.runSpeed();
384 }
385 if(directions[4] == "LEFT")
386 {
387 x+=1;
388 stepperM.setSpeed(MAX_SPEED);
389 stepperM.runSpeed();
390 }
391 else if(directions[4] == "RIGHT")
392 {
393 x-=1;
394 stepperM.setSpeed(-MAX_SPEED);
395 stepperM.runSpeed();
396 }
397 }
398
399 int cmToTurns(float x)
400 {
401   return x*2000;
402 }
403
404 void printWifiStatus()
405 {
```

```
406 // print the SSID of the network you're attached to:
407 Serial.print("SSID: ");
408 Serial.println(WiFi.SSID());
409
410 // print your WiFi shield's IP address:
411 IPAddress ip = WiFi.localIP();
412 Serial.print("IP Address: ");
413 Serial.println(ip);
414
415 // print the received signal strength:
416 long rssi = WiFi.RSSI();
417 Serial.print("signal strength (RSSI):");
418 Serial.print(rssi);
419 Serial.println(" dBm");
420 }
421 bool checkReset()
422 {
423 if(digitalRead(RESET)==HIGH)
424 {
425 reset();
426 return true;
427 }
428 return false;
429 }
430
431 void reset()
432 {
433 lcd.clear();
434 if(client.connected())
435 {
436 WiFi.disconnect();
```

```
437 alreadyConnected = false;
438 lcd.setCursor(0, 1);
439 lcd.print("Net. disConnected");
440 printNetworkStatus = false;
441 }
442 mode = 0;
443
444 lcd.setCursor(0, 0);
445 lcd.print("RESETTING...");
446 Serial.println("Resetting...");
447
448 stepperM.stop();
449 stepperZ1.stop();
450 stepperZ2.stop();
451 stepperY1.stop();
452 stepperY2.stop();
453 stepperM.moveTo(0);
454 stepperZ1.moveTo(0);
455 stepperZ2.moveTo(0);
456 stepperY1.moveTo(0);
457 stepperY2.moveTo(0);
458 delay(300);
459 while(stepperM.distanceToGo() != 0 || stepperZ1.
      ↪ distanceToGo() != 0 || stepperY1.distanceToGo() !=
      ↪ 0)
460 {
461 delay(COMMON_DELAY);
462 stepperM.run();
463 stepperZ1.run();
464 stepperZ2.run();
465 stepperY1.run();
```

```
466 stepperY2.run();
467 }
468 x = 0;
469 y = 0;
470 z = 0;
471 lcd.clear();
472 lcd.setCursor(0, 0);
473 lcd.print("Welcome...");
474 }
```

Appendix D

Mako an Blue Belt Tibia cut results

D.1 Mako

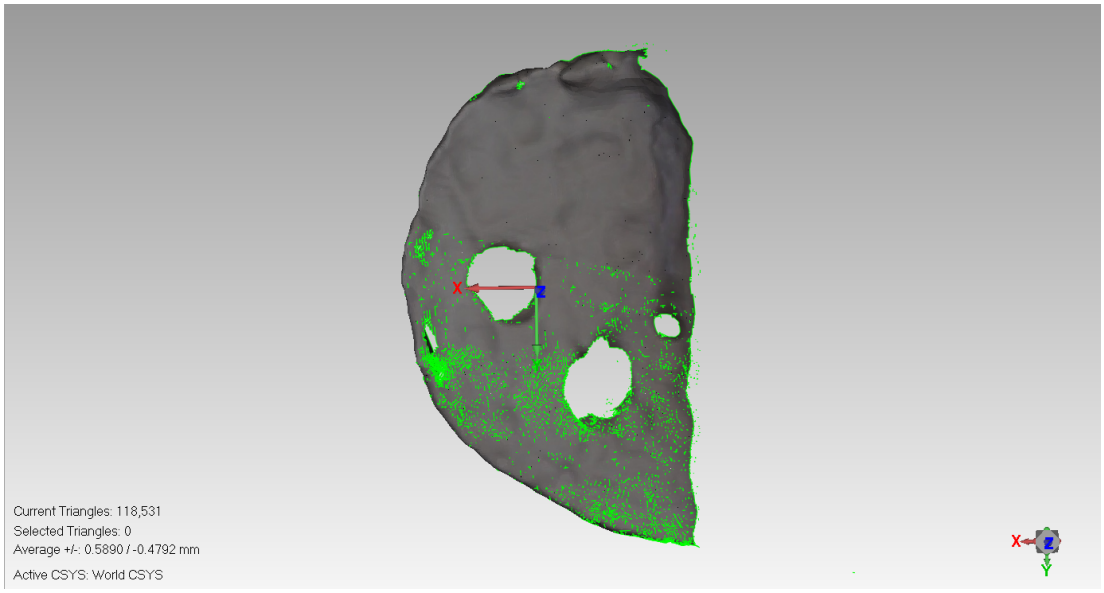


Figure D.1: Mako Tibia Bone 2 - Cut Only View

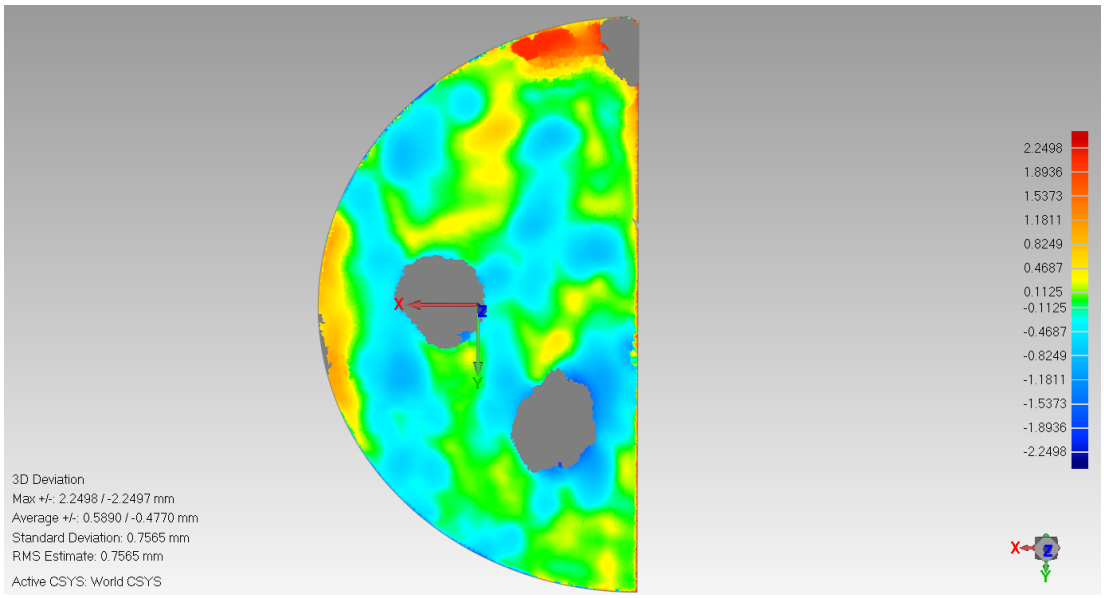


Figure D.2: Mako Tibia Bone 2 - 3D Comparison View

Figure D.2 shows powder blue, light blue, green, yellow and some orange areas with the errors of -0.82 *mm*, -0.46 *mm*, 0 for green, 0.46 *mm* and 1.18 *mm* respectively.

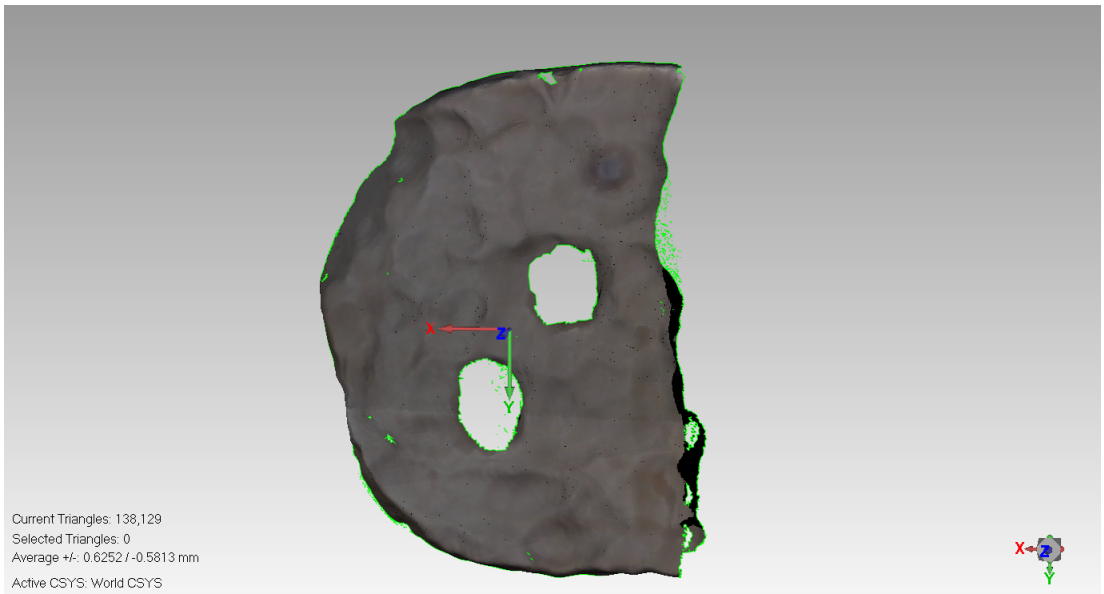


Figure D.3: Mako Tibia Bone 3 - Cut Only View

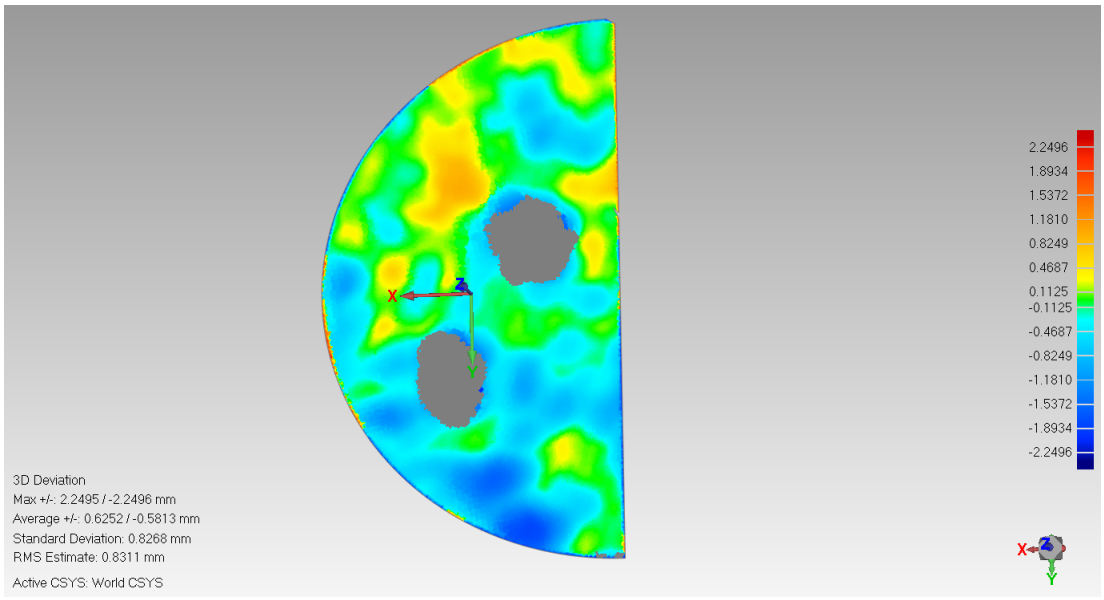


Figure D.4: Mako Tibia Bone 3 - 3D Comparison View

Figure D.4 shows blue, powder blue, light blue, green, yellow and orange colours with errors -1.89 *mm*, -0.82 *mm*, -0.46 *mm*, 0 for green, 0.46 *mm* and 1.18 *mm* respectively.

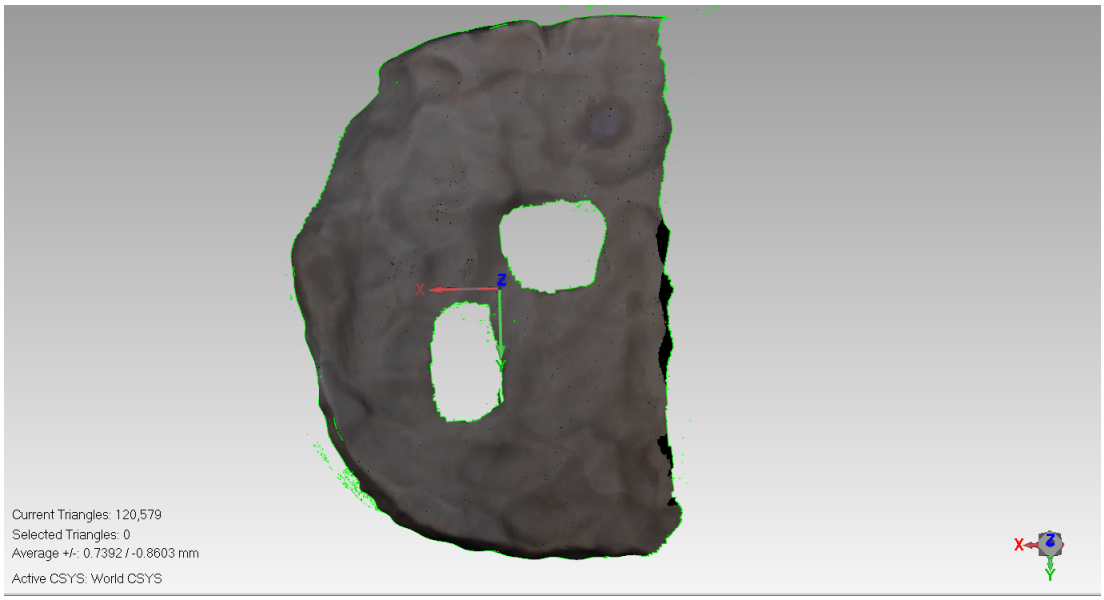


Figure D.5: Mako Tibia Bone 4 - Cut Only View

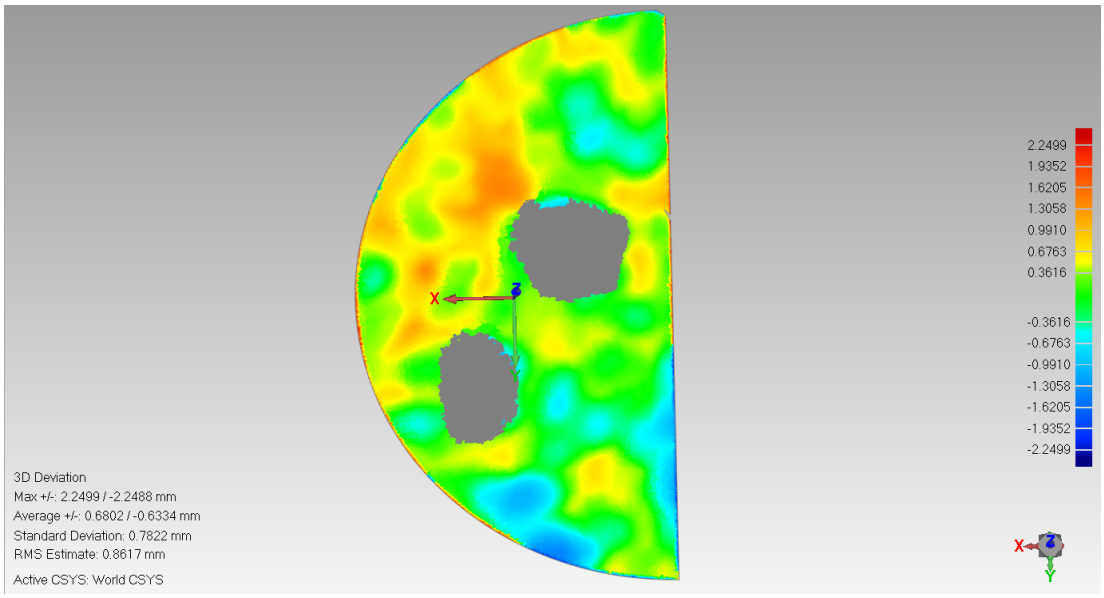


Figure D.6: Mako Tibia Bone 4 - 3D Comparison View

Figure D.6 shows powder blue, light blue, green, yellow and orange colours with errors -1.3 mm, -0.67 mm, 0 for green, 0.67 mm and 1.6 mm and respectively.

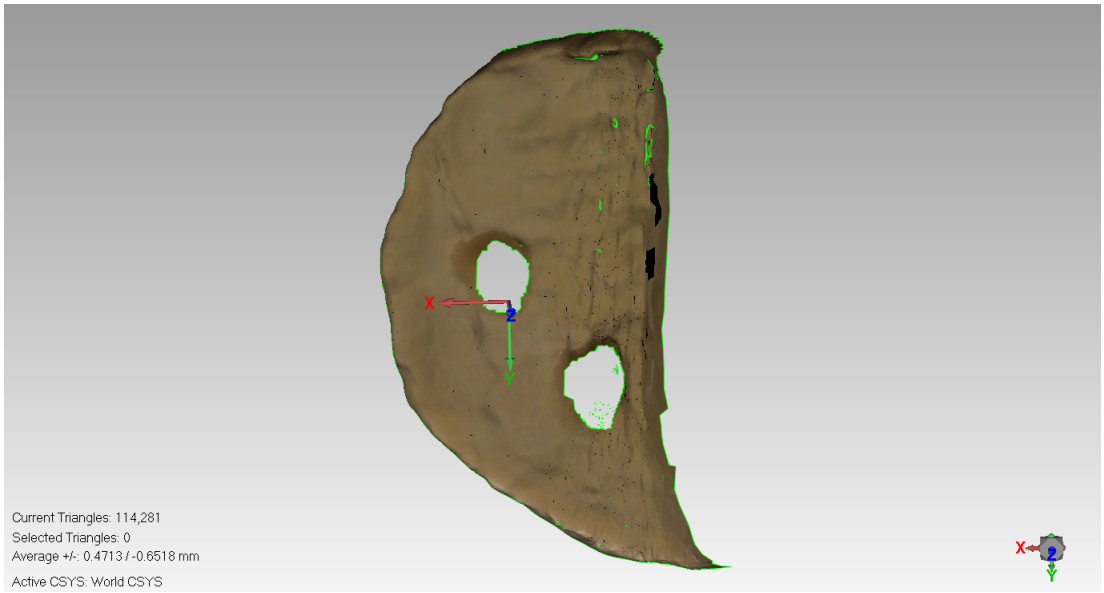


Figure D.7: Mako Tibia Bone 5 - Cut Only View

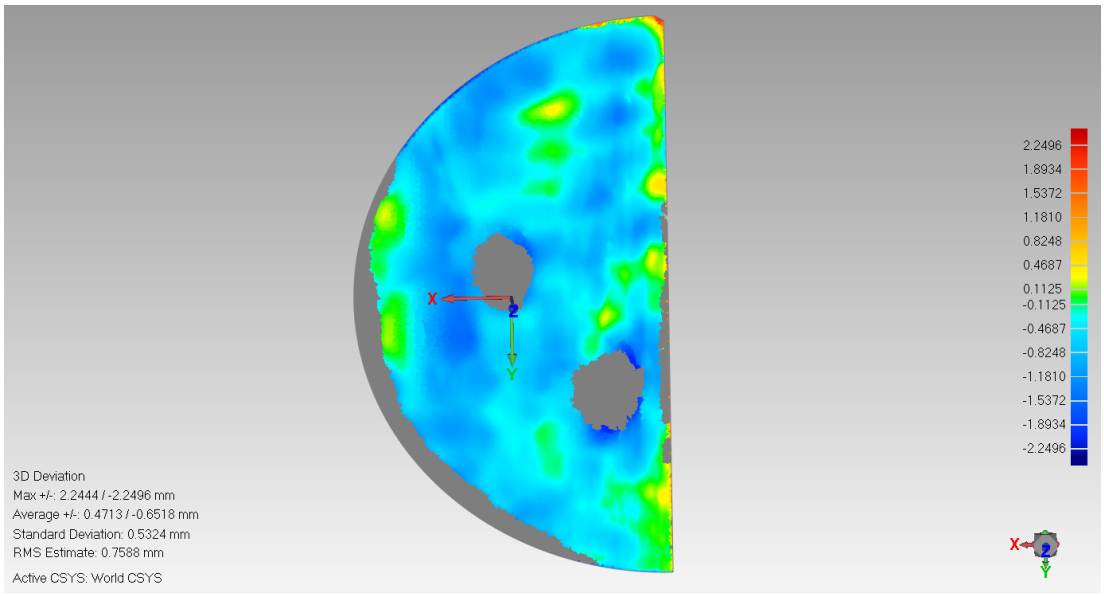


Figure D.8: Mako Tibia Bone 5 - 3D Comparison View

Figure D.8 shows powder Blue, light blue, green and very little spikes with yellow with errors of -1.53 mm, -0.46 mm, 0 for green and 0.46 mm respectively.

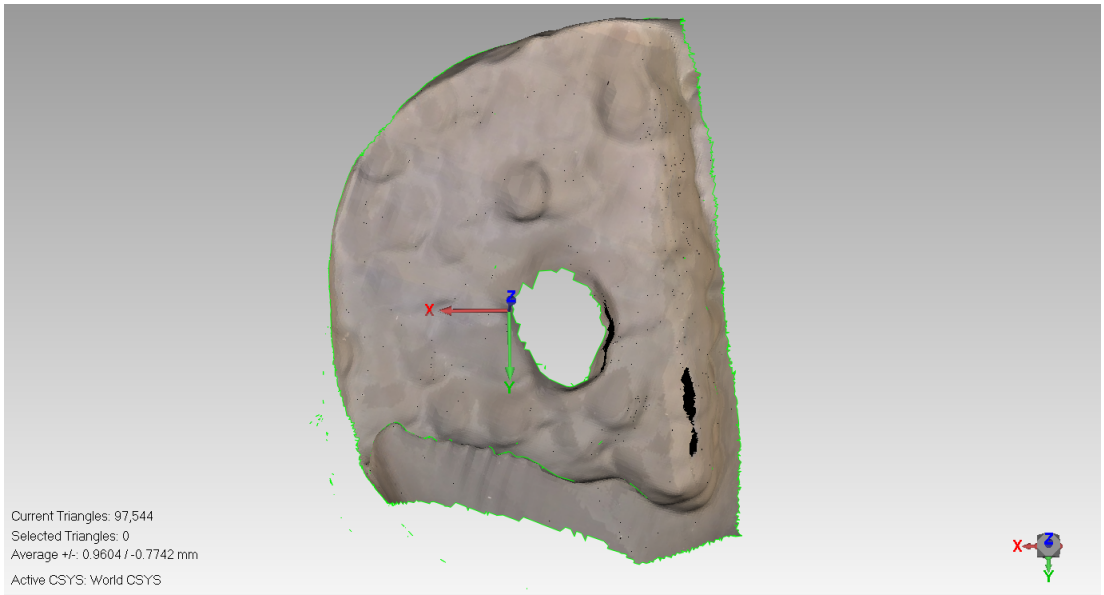


Figure D.9: Mako Tibia Bone 6 - Cut Only View

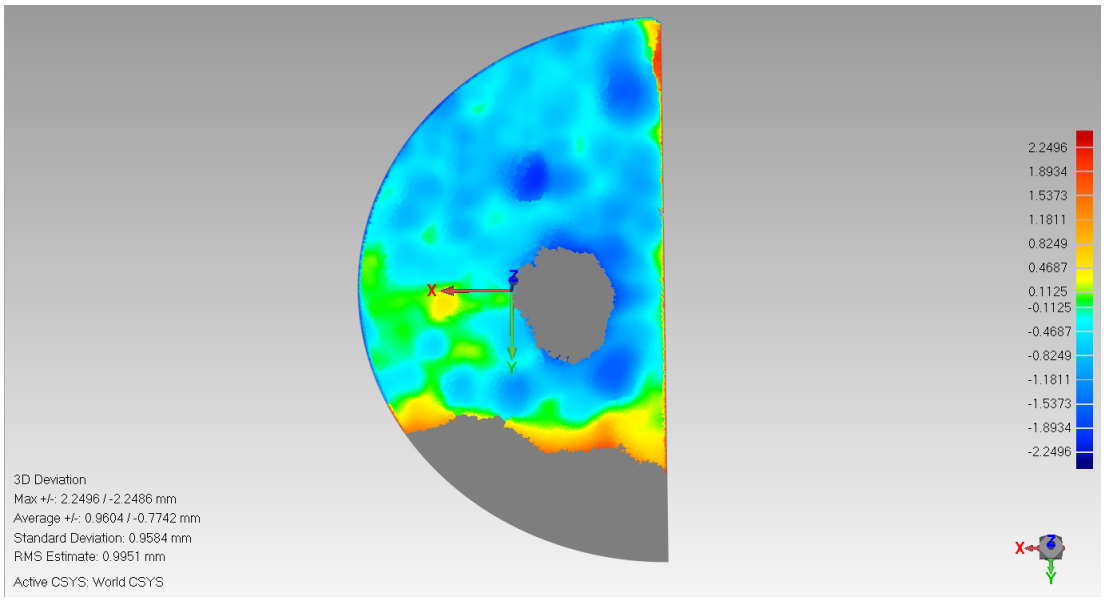


Figure D.10: Mako Tibia Bone 6 - 3D Comparison View

Figure D.10 shows blue, powder blue, light blue, green, yellow and and some orange coloured parts but it belongs to the edge of the unprocessed part and will be neglected. The error ranges will be -2.2 mm, -1.53 mm, -0.46 mm, 0 for green and 0.46 mm respectively.

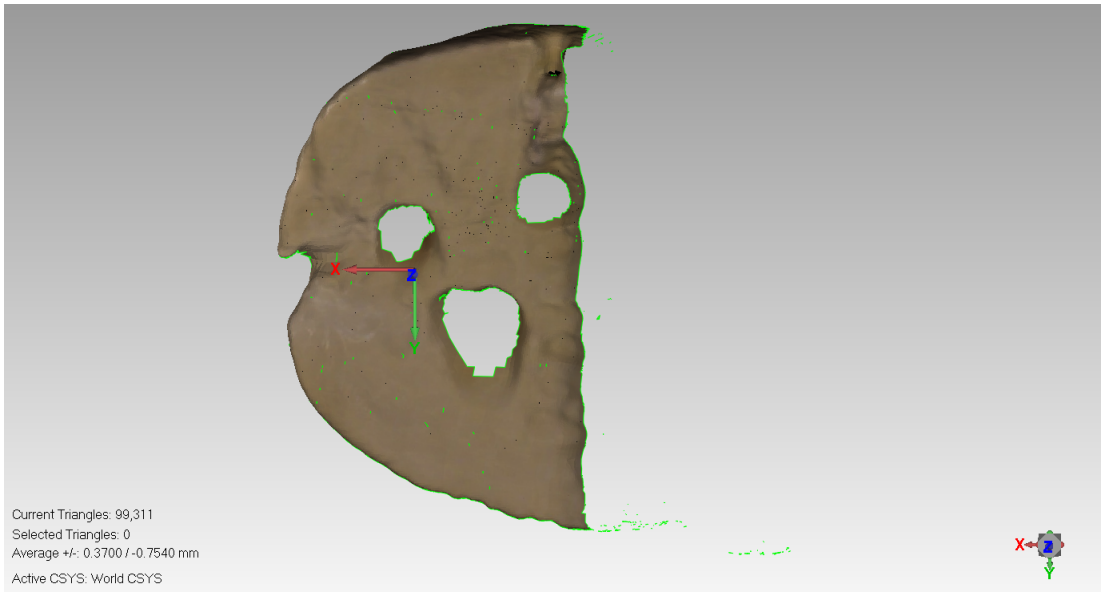


Figure D.11: Mako Tibia Bone 7 - Cut Only View

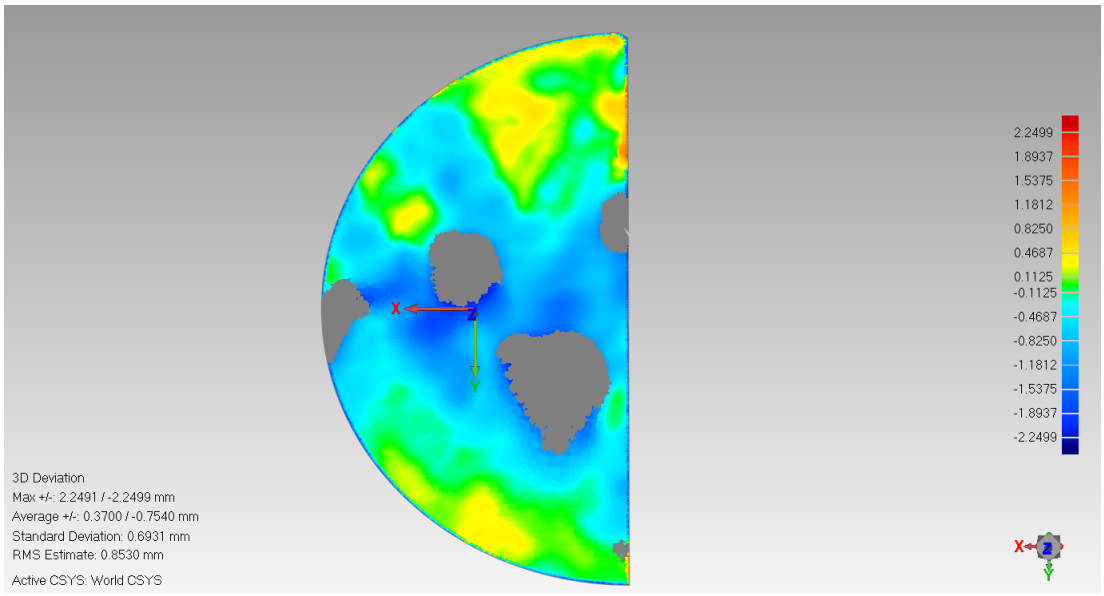


Figure D.12: Mako Tibia Bone 7 - 3D Comparison View

Figure D.12 shows blue, powder blue, light blue, green and yellow colours with errors of -1.89 *mm*, -1.18 *mm*, -0.46 *mm*, 0 for green and 0.46 *mm* respectively.

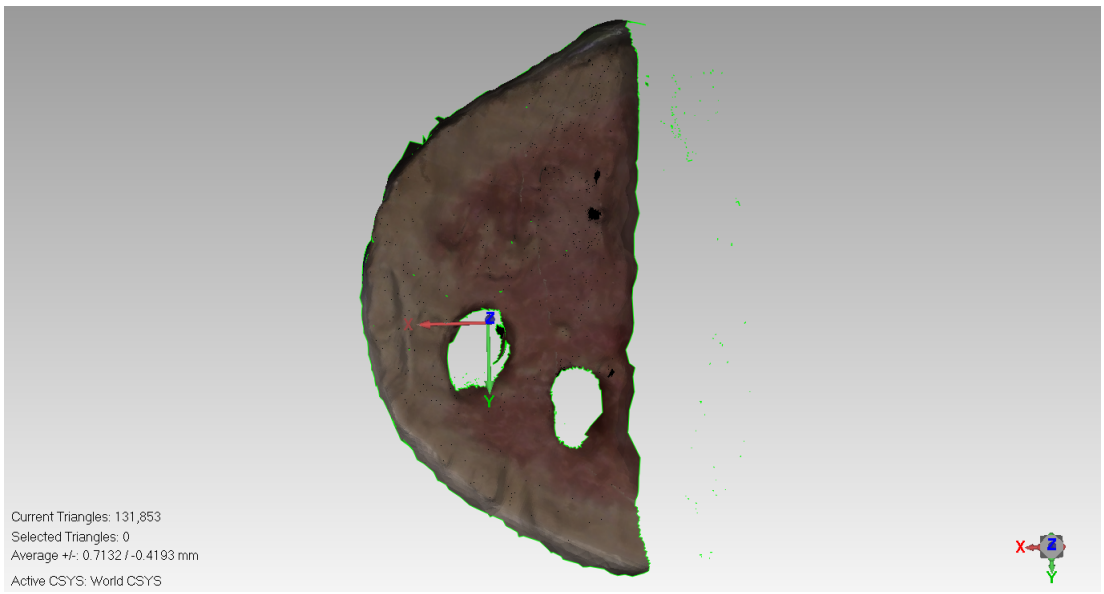


Figure D.13: Mako Tibia Bone 8 - Cut Only View

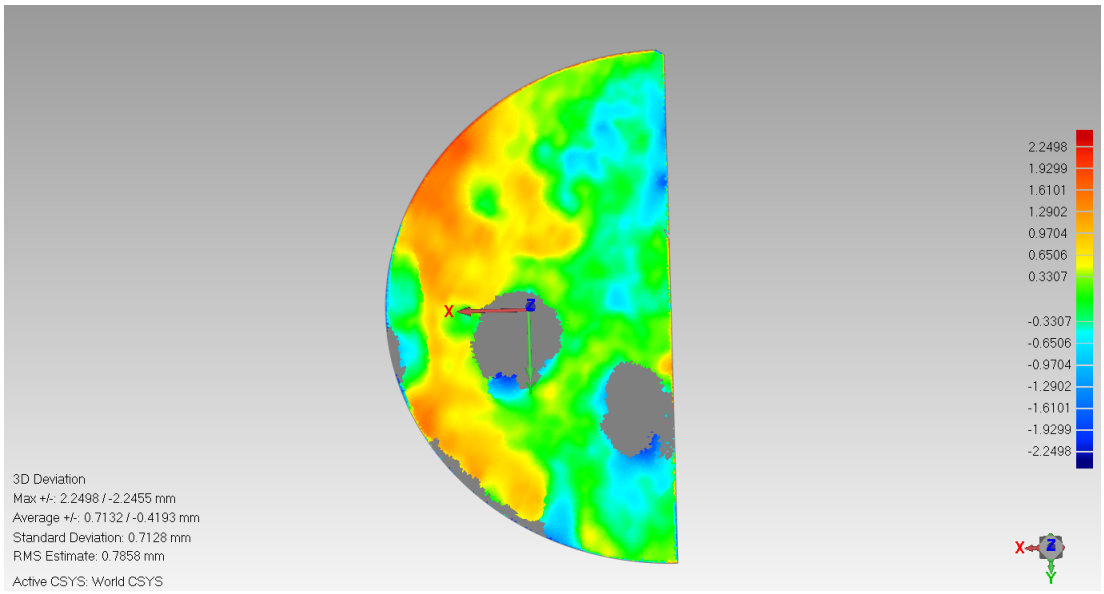


Figure D.14: Mako Tibia Bone 8 - 3D Comparison View

Figure D.14 shows powder blue, light blue, green, yellow and orange colours with errors of -1.2 mm, -0.65 mm, 0 for green, 0.65 mm and 1.61 mm respectively.

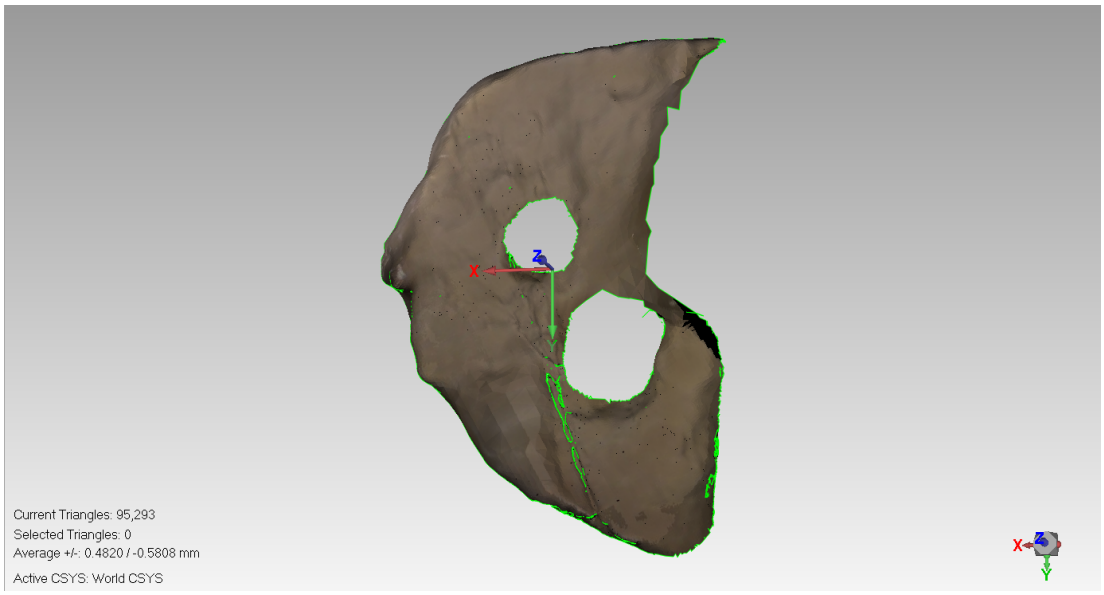


Figure D.15: Mako Tibia Bone 9 - Cut Only View

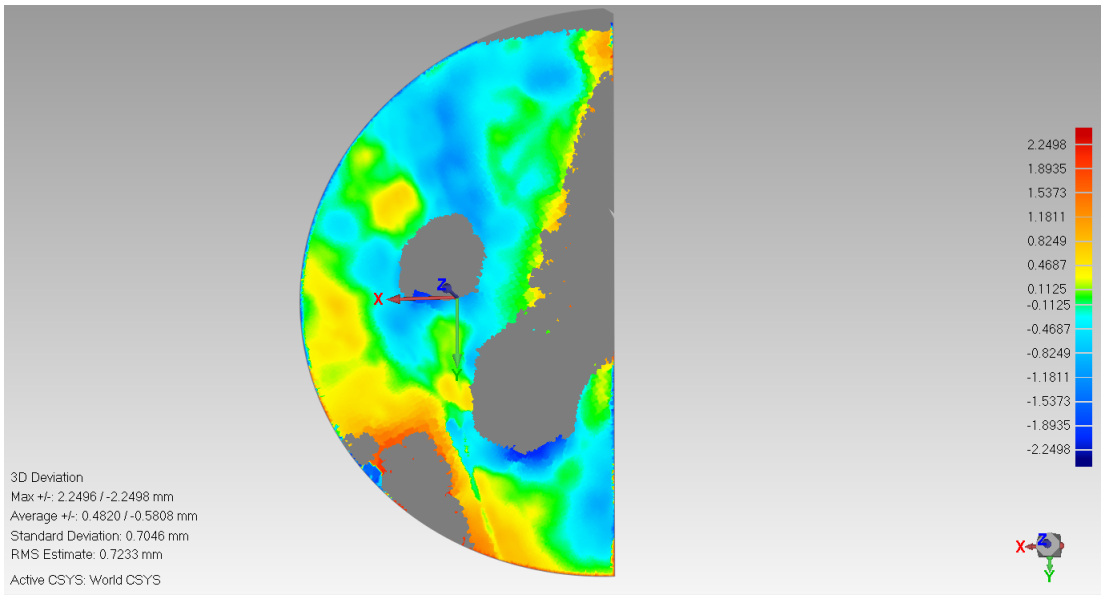


Figure D.16: Mako Tibia Bone 9 - 3D Comparison View

Figure D.16 shows powder blue, light blue, green, yellow and orange colours with errors of -1.18 mm, -0.46 mm, 0 for green, 0.46 mm and 0.82 mm respectively. The Gray areas were made due to poor cutting quality as shown in Figure D.15 which results to difference in comparison out of the scale range (2.25 mm to -2.25 mm).

D.2 Blue Belt

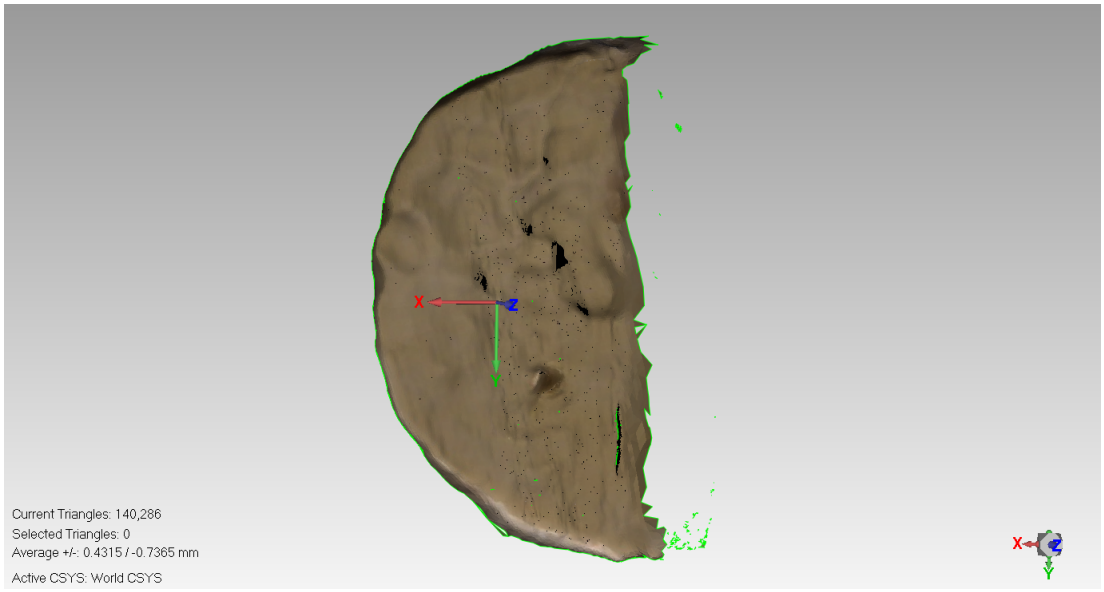


Figure D.17: Blue Belt Tibia Bone 2 - Cut Only View

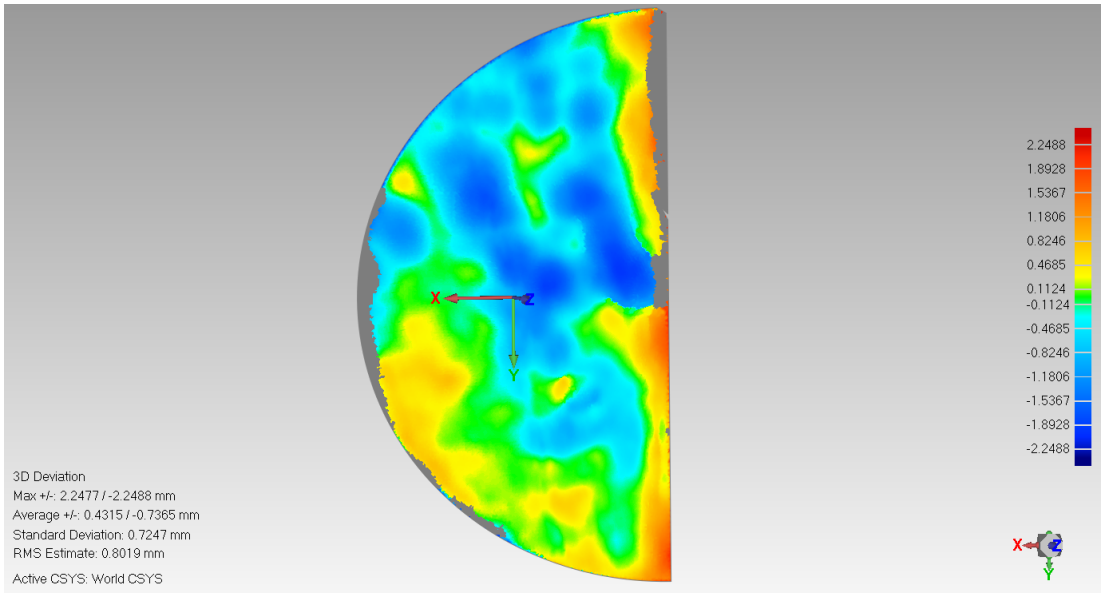


Figure D.18: Blue Belt Tibia Bone 2 - 3D Comparison View

Figure D.18 shows blue, powder blue, light blue, green and two scales of orange colours with errors of -2.24 *mm*, -0.82 *mm*, -0.46 *mm*, 0 for green, 0.82 *mm* and 1.18 *mm* respectively.

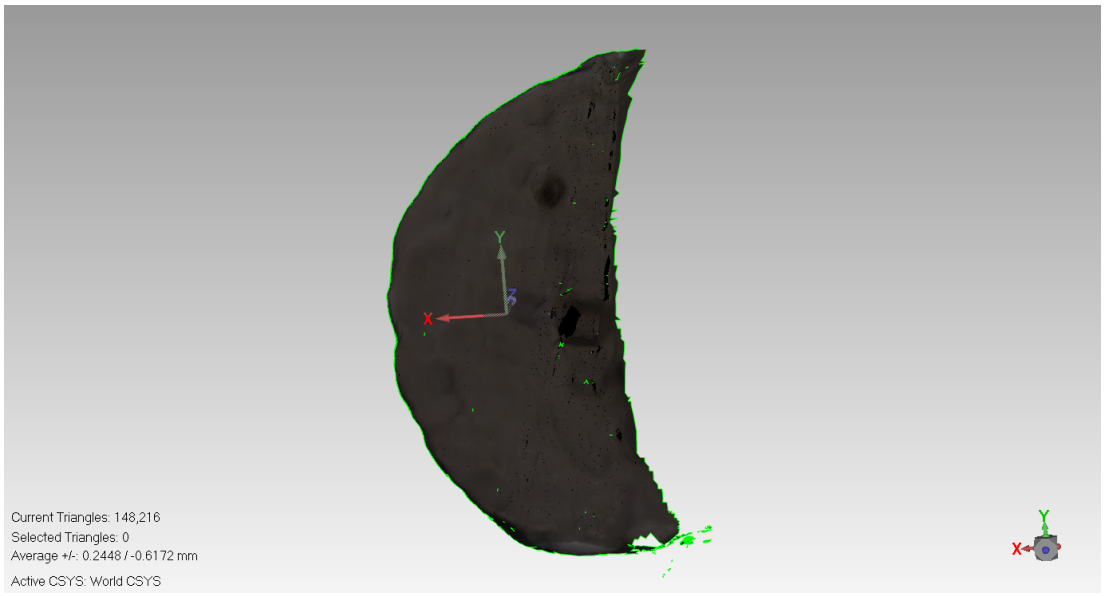


Figure D.19: Blue Belt Tibia Bone 3 - Cut Only View

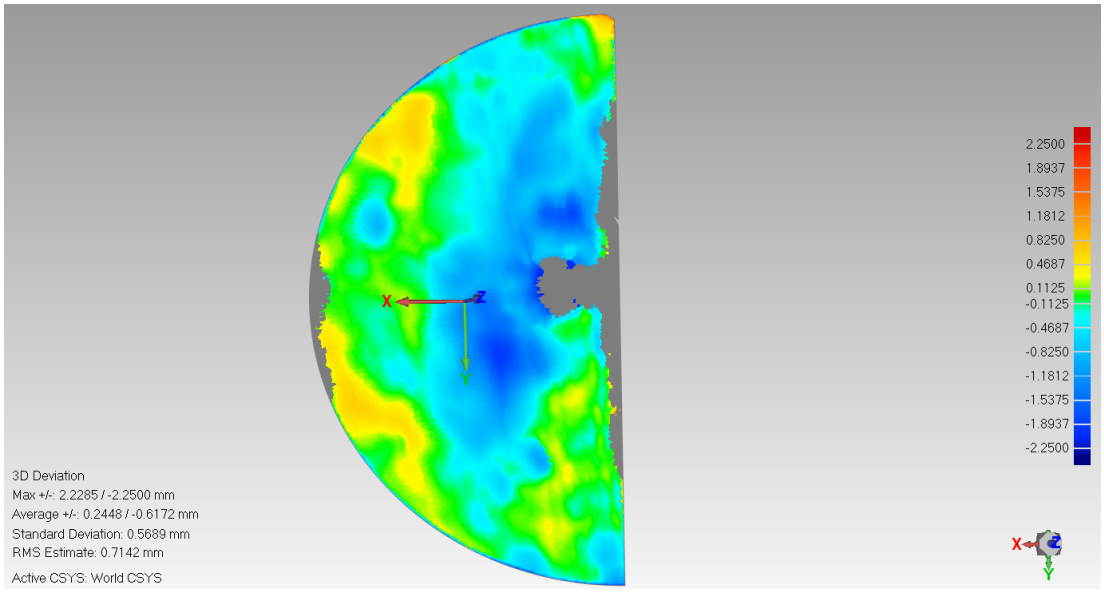


Figure D.20: Blue Belt Tibia Bone 3 - 3D Comparison View

Figure D.20 shows blue, powder blue, light blue, green, yellow and orange colours with errors of -1.89 mm, -1.18 mm, -0.46 mm, 0 for green, 0.46 mm and 0.82 mm respectively.

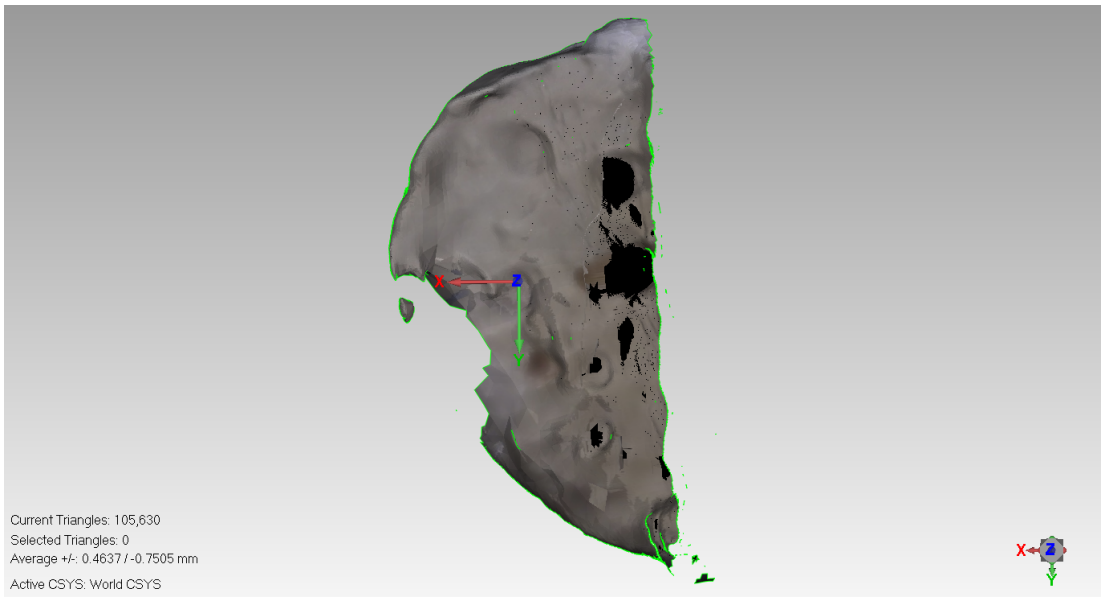


Figure D.21: Blue Belt Tibia Bone 4 - Cut Only View

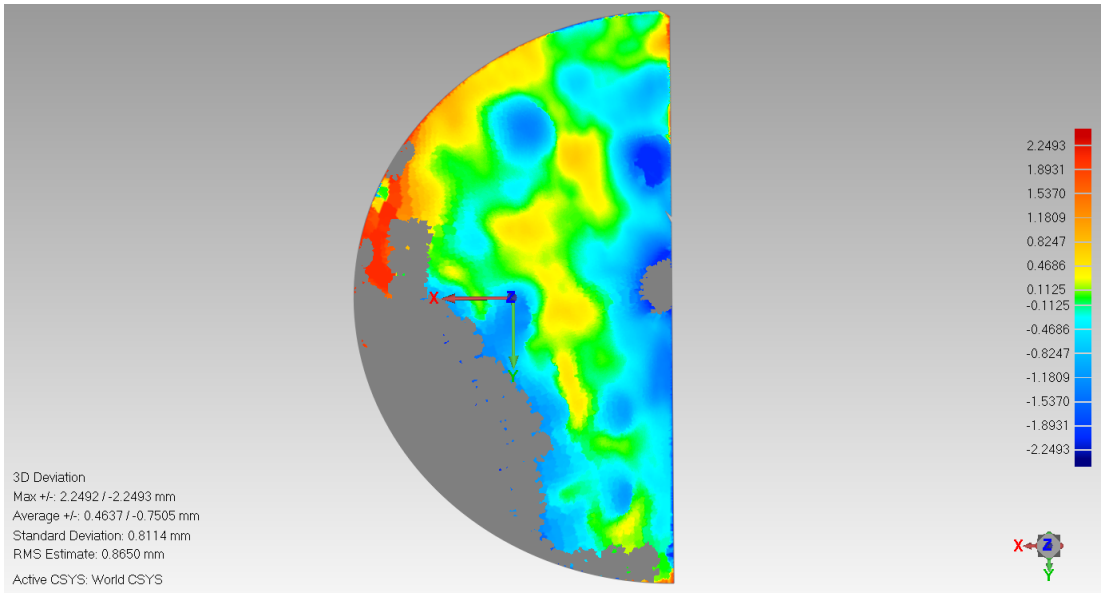


Figure D.22: Blue Belt Tibia Bone 4 - 3D Comparison View

Figure D.22 shows blue, powder blue, light blue, green, yellow and red colours with errors of -2.24 *mm*, -0.82 *mm*, -0.46 *mm*, 0 for green, 0.46 *mm* and 2.24 *mm* respectively. The missing part of the bone on the left was due to excessive burr and cutting.

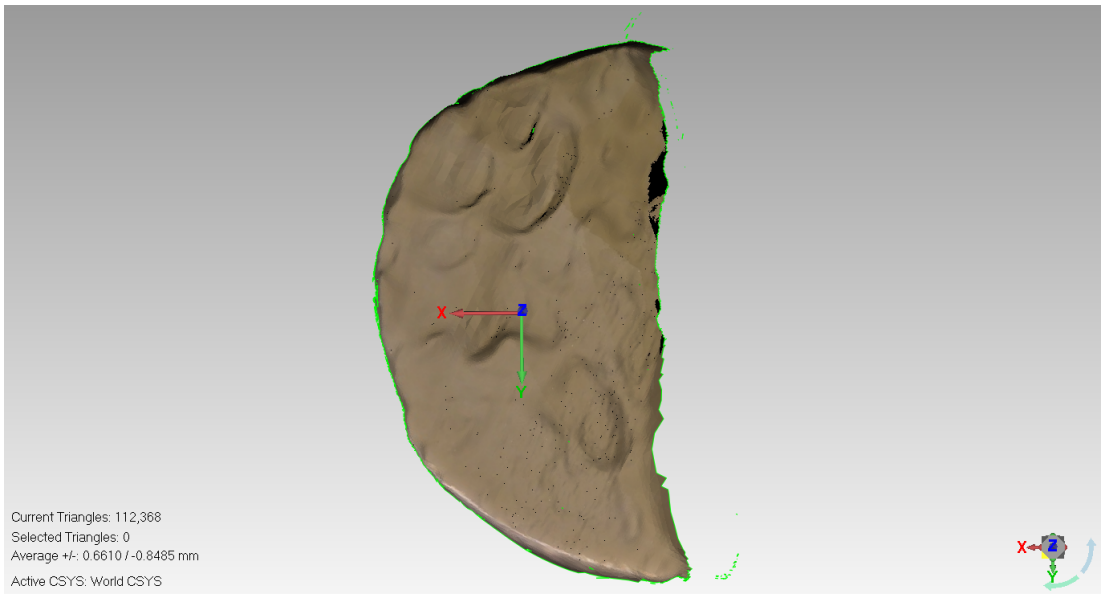


Figure D.23: Blue Belt Tibia Bone 5 - Cut Only View

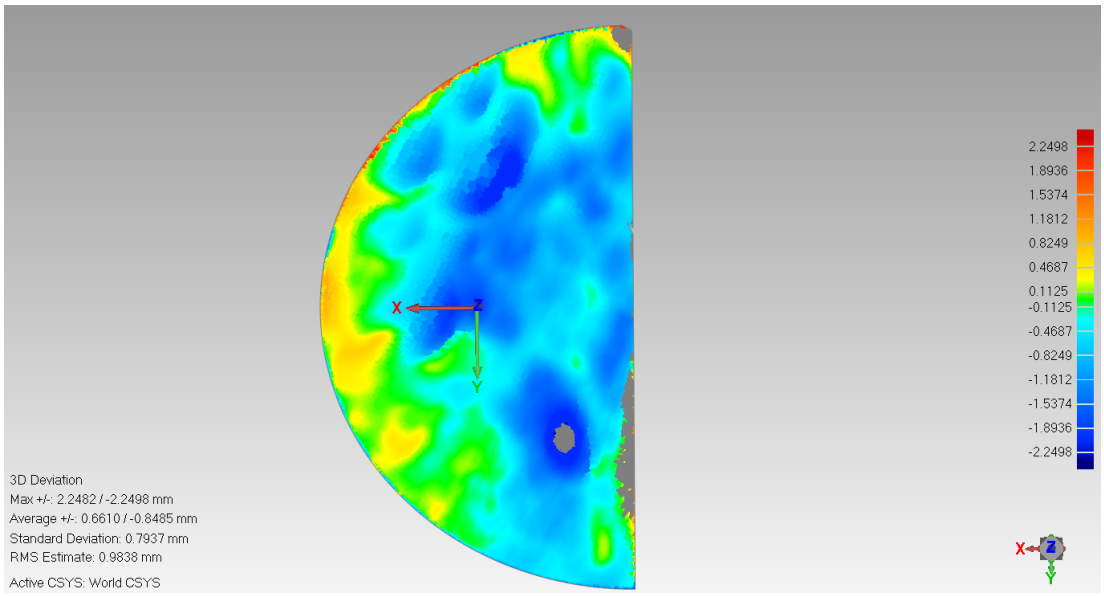


Figure D.24: Blue Belt Tibia Bone 5 - 3D Comparison View

Figure D.24 shows blue, powder blue, light blue, green, yellow and orange colours with errors of -2.24 mm, -1.53 mm, -0.46 mm, 0 for green and 0.46 mm respectively.

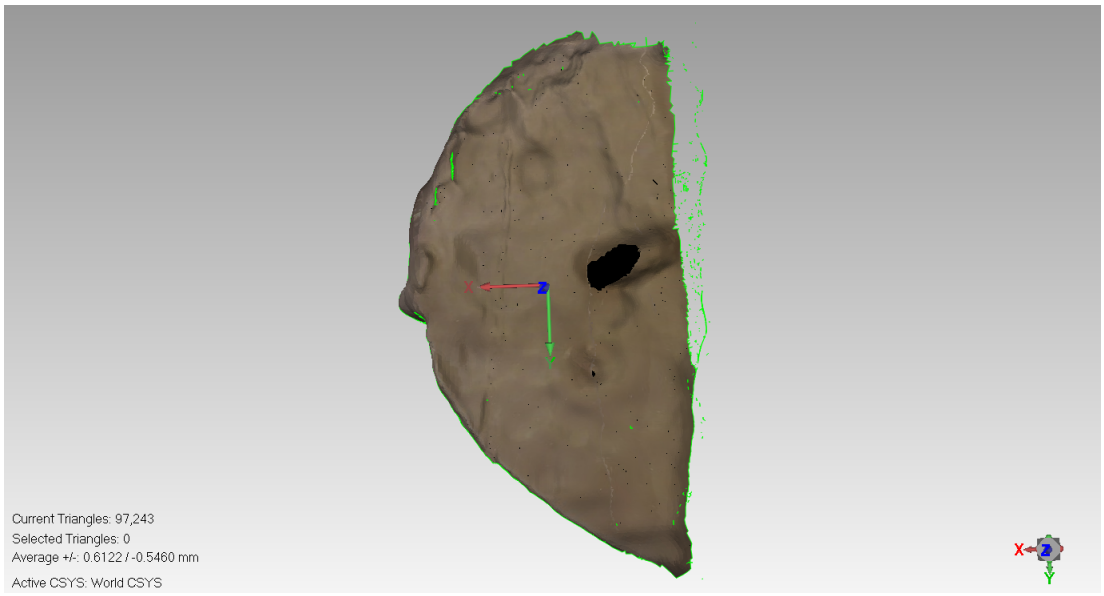


Figure D.25: Blue Belt Tibia Bone 6 - Cut Only View

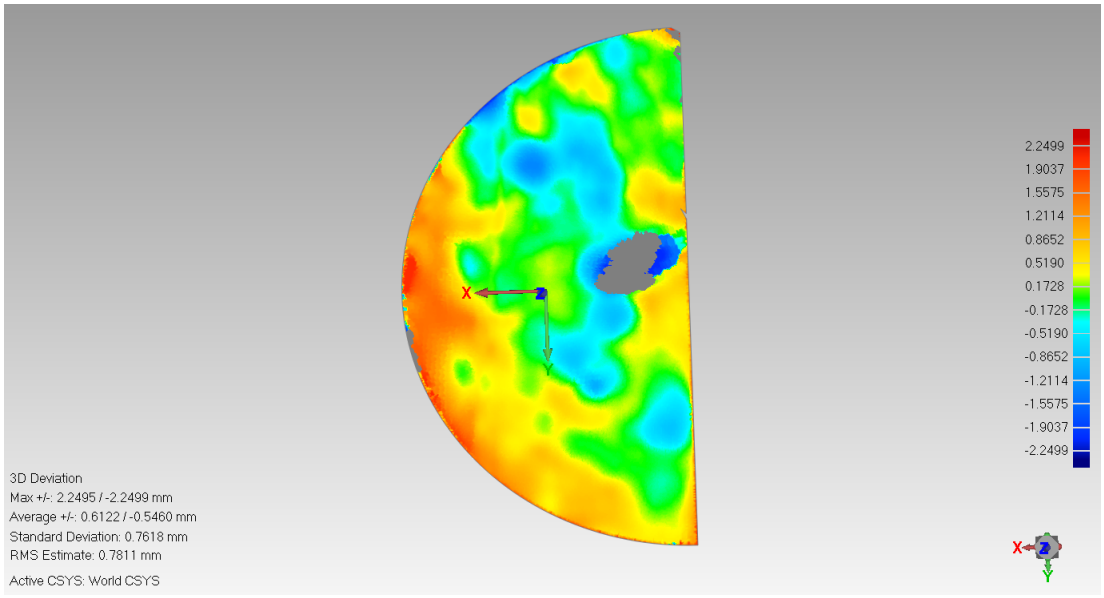


Figure D.26: Blue Belt Tibia Bone 6 - 3D Comparison View

Figure D.26 shows powder blue, light blue, green, orange and flame colours with errors of -1.21 mm, -0.51 mm, 0 for green, 0.86 mm and 1.9 mm respectively.

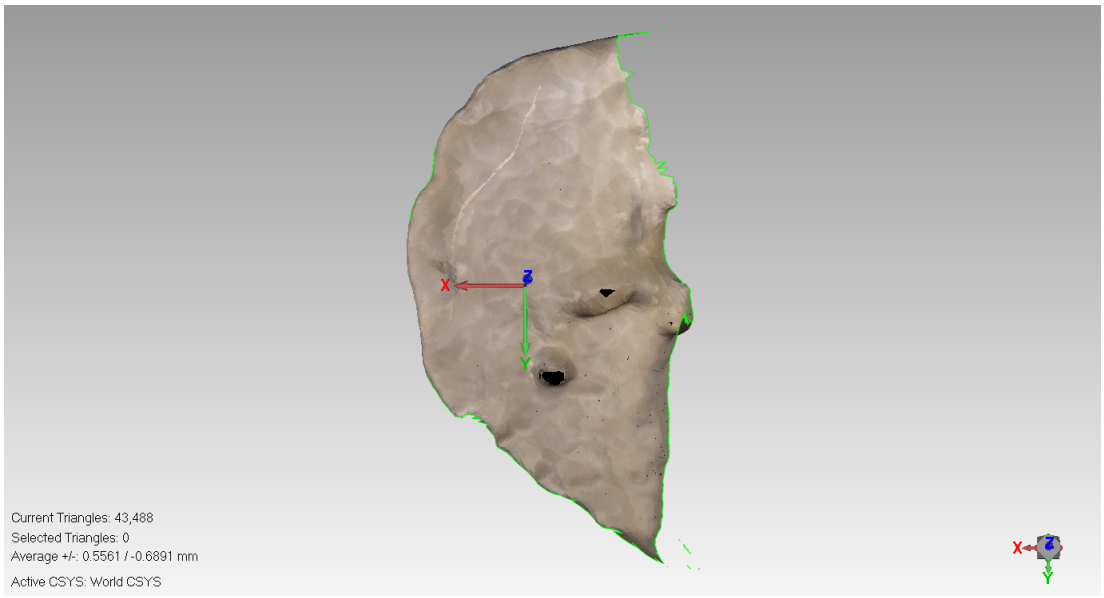


Figure D.27: Blue Belt Tibia Bone 7 - Cut Only View

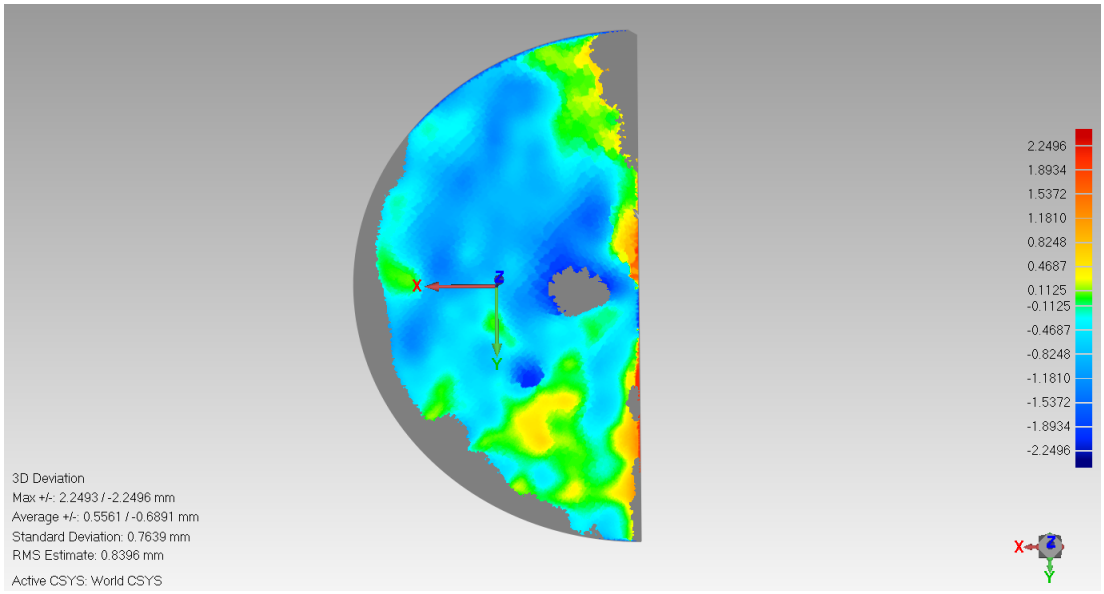


Figure D.28: Blue Belt Tibia Bone 7 - 3D Comparison View

Figure D.28 shows blue, light blue, green, yellow and orange colours with errors of -1.89 mm, -0.46 mm, 0 for green, 0.46 mm and 0.82 mm respectively.

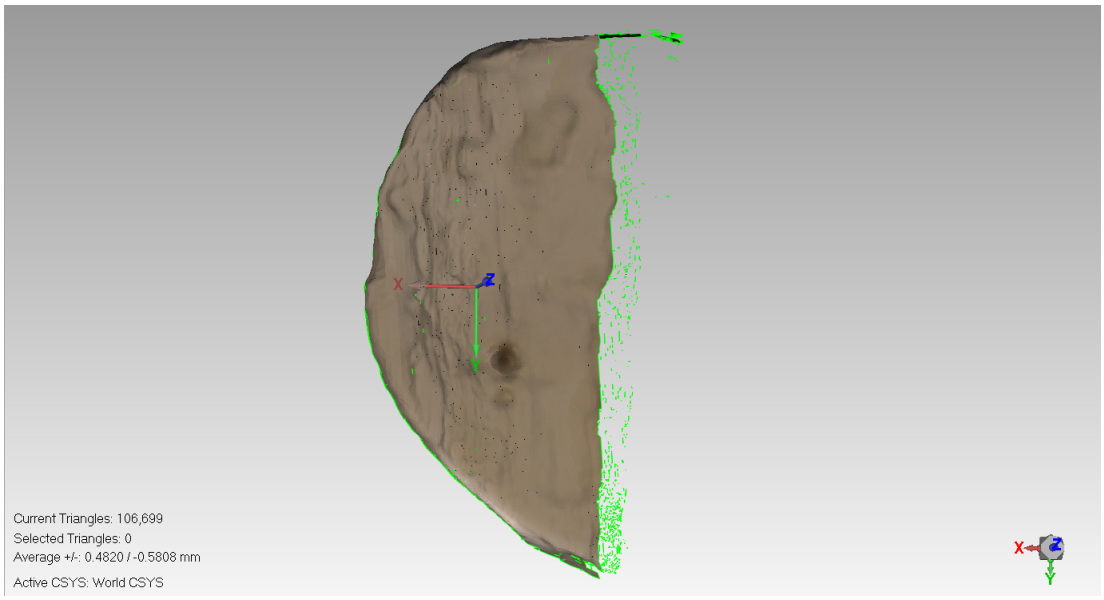


Figure D.29: Blue Belt Tibia Bone 8 - Cut Only View

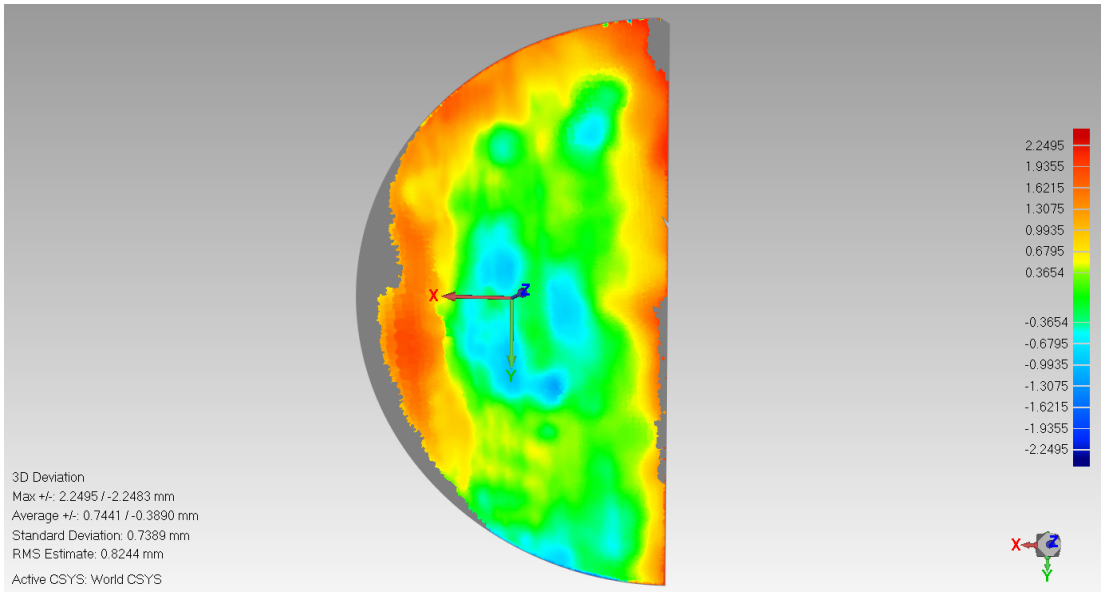


Figure D.30: Blue Belt Tibia Bone 8 - 3D Comparison View

Figure D.30 shows powder blue, light blue, green, yellow and orange colours with errors of -0.99 *mm*, -0.67 *mm*, 0 for green, 0.67 *mm* and 1.93 *mm* respectively. The red areas on the right side because of the bone lib and will be neglected.

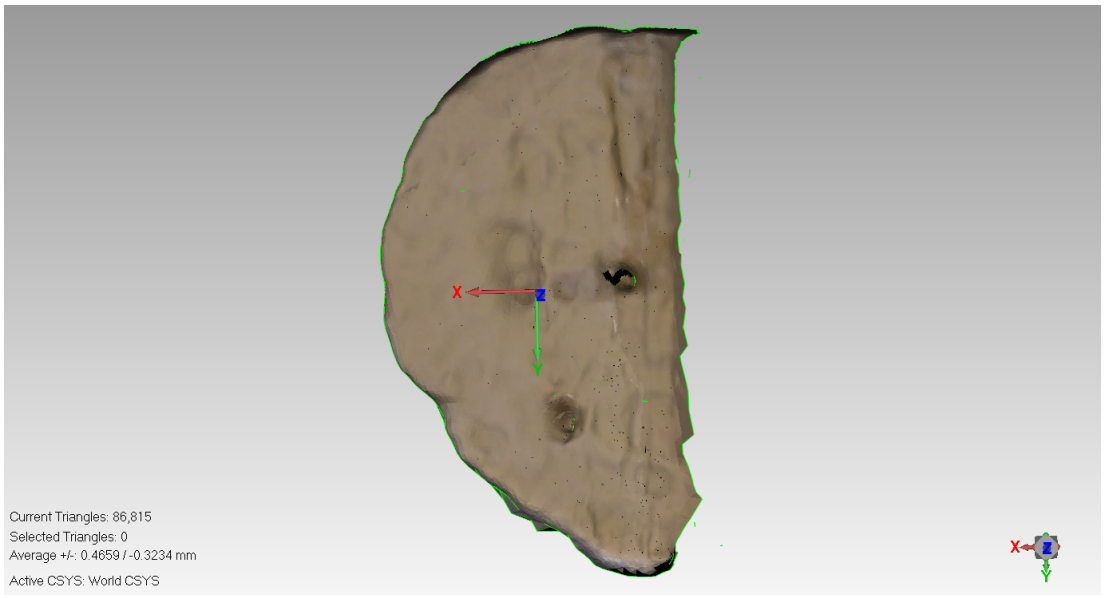


Figure D.31: Blue Belt Tibia Bone 9 - Cut Only View

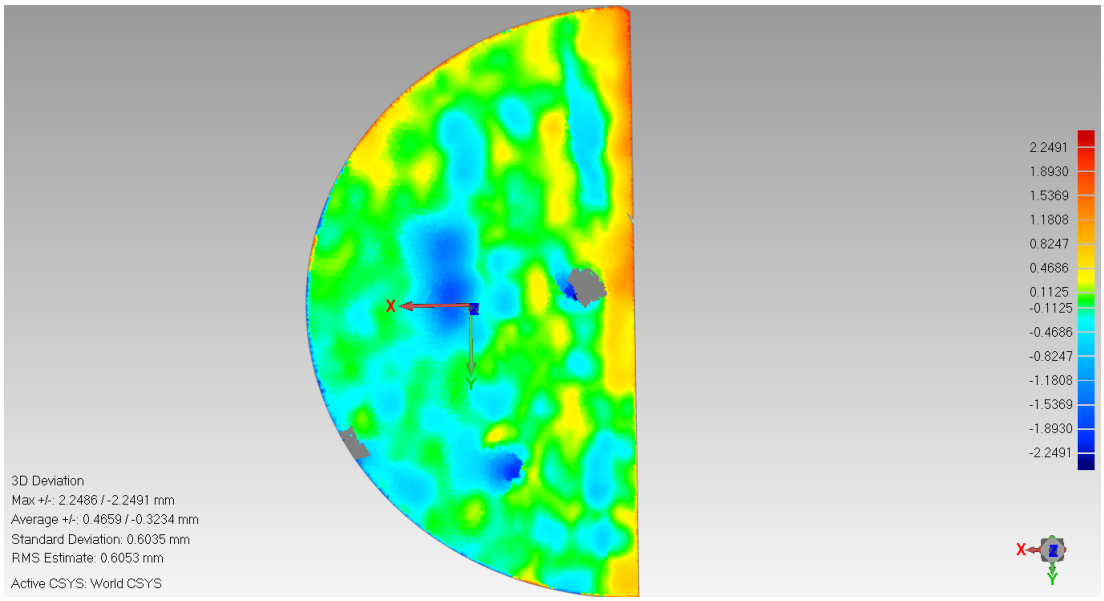


Figure D.32: Blue Belt Tibia Bone 9 - 3D Comparison View

Figure D.32 shows blue, powder blue, light blue, green, yellow and orange colours with errors of -1.89 *mm*, -0.46 *mm*, 0 for green, 0.46 *mm* and 0.62 *mm* respectively.