UNIVERSITE LIBRE DE BRUXELLES
Faculte de Polytechnique

August 29, 2016

Computational medical imaging for total knee arthroplasty using Visualization Toolkit

Professor Bernardo Innocenti

Final Master Thesis

Javier Romero Sánchez

Depending on the validity of the assumptions made in reducing the physical problem to a numerical algorithm, the computer output may provide a detailed picture of the true physical behavior or it may not even remotely resemble it.

Ray Clough, 1980

Contents

List of Figures

Computational medical imaging for total knee arthroplasty using Visualization Image Toolkit

Jose Javier Romero Sanchez

August 29, 2016

Chapter 1

Aim of the project

1.1 Origin

This project is presented as a Master Thesis in the field of Civil Engineering, Biomedical specialization. As the project of an Erasmus exchange student, this thesis has been under supervision both the Universite Livre de Bruxelles and the Universitat Politecnica de Catalunya. The purpose of this thesis to put in practice all the knowledges acquired during this Master in Industrial Engineering in UPC and to be a support for medical staff in total knee arthoplasty procedures.

Prof. Emmanuel Thienpont has been working for years as orthopaedic surgeon at the Hospital Sant Luc, Brussels. His years of work and research have been mainly focused on Total Knee Arthroplasty or TKA. During one of the most important steps of this procedure, the orthopaedic surgeon has to cut the head of the femur following two perpendicular cutting planes. Nevertheless, the orientation of these planes are directly dependant of the femur constitution.
This Master Thesis has been conceived in order to offer the surgeon a tool to determine the proper direction planes in a previous step before the surgical procedure. This project pretends to give the surgeon an open-free computational platform to access to patient geometrical and physiological information before involving the subject in any invasive procedure.

1.2 Objectives

The main objective of this thesis is to provide the medical staff an intuitive and open-source software for process and analyze medical data from contrast tomography scans of the patient leg (right or left). In the present project, several aspects have been taken into account in order to make the software fulfill the following specifications:

- To design and to implement computational algorithms to pre-process DICOM raw files in order to optimize memory consumption and processing speed.
- To design and to implement a visualization software for medical data, extracted from computer tomography scans.
- To set what kinds of geometric and property information are relevant to extract from the medical data.
- To design and to implement computational algorithms in order to extract geometric and property information from the medical data. This information should be able to be imported to other softwares for future analysis (for example, Excel files).

- To set the most ergonomic ways of interaction between user and machine for the design of the software interface.
- To design and to implement the software interface, following the ergonomics criteria set above.
- To make the software being validated by specialized Biomedical Engineering students, medical staff and experts in the field of biomechanics and orthopaedic surgery.

In order to fulfill all the objectives set above, the implementation of the software has been based on open-source free modules, such as VTK for the medical data visualization and PyQt4 for the interface implementation.

Chapter 2

The leg

The leg is the lower extremity of human body. Although it popularly comprises the extremity from the thigh to the foot, medical and anatomy dictionaries and guides refer the section from the lowest part of the knee to the ankle. The leg is one of the most important limbs in the locomotor system in humans and other biped animals. Legs are required for the majority of the most common movements in human life, such as walking, jumping, standing and so on.

2.1 Anatomy

In a biomechanical point of view, the leg can be divided in to systems: muscular leg system and bone leg system. Both systems play together the most important role in the locomotion of the leg. In a deeper definition for the bone system, the leg is composed by the femur (or thighbone), the longest bone in human body, which is attached to the pelvis by the femoral head (hip joint); the tibia and the patella (both attached with the femur, composing the knee joint); and the fibula. The edge of the tibia and fibula is known as ankle, and it composes the joint that links the foot with the rest of the leg. In order to understand future concepts and procedures, several important points for the bones of the leg need to be pointed. These points were previously stated in Victor et al [9] studies. The ones selected in Figure 2.2 are the points required to define the anatomical axes of the leg.

2.2 Anatomical and mechanical axes

For the bones of the leg, several axis can be stated from several couples of points presented in Figure 2.2. Theses axises may help the medical staff to diagnose possible anatomical deformities (axus/valgus) or knee misalignment. From the variety of axis that can be defined, the ones that may be of interest in this project are presented in Figure 2.2.

This axises were defined by Victor et al [9], Cherian et at [2] statements, and according to Decking et at [3] and Pickering et al [5]:

- Femoral Mechanical Axis: Line joining the femoral knee center (FKC) and the femoral hip center (FHC).
- Femoral Anatomical Axis: Line connecting the FKC and femoral piriformis fosa (midpoint of the intercondylar notch of the femur, FPF).

- Femoral Transcondylar Axis: Line connecting the ends of both medial and lateral femoral condyles (FLE, FME).
- Tibial Mechanical Axis: Line joining the tibial knee center (TKC) and the ankle. It is supposed to be coincident with the tibial anatomical axis.
- Transtibial Axis: Line connecting the center of the tibial medial and lateral condyle centers (TMCC, TLCC).

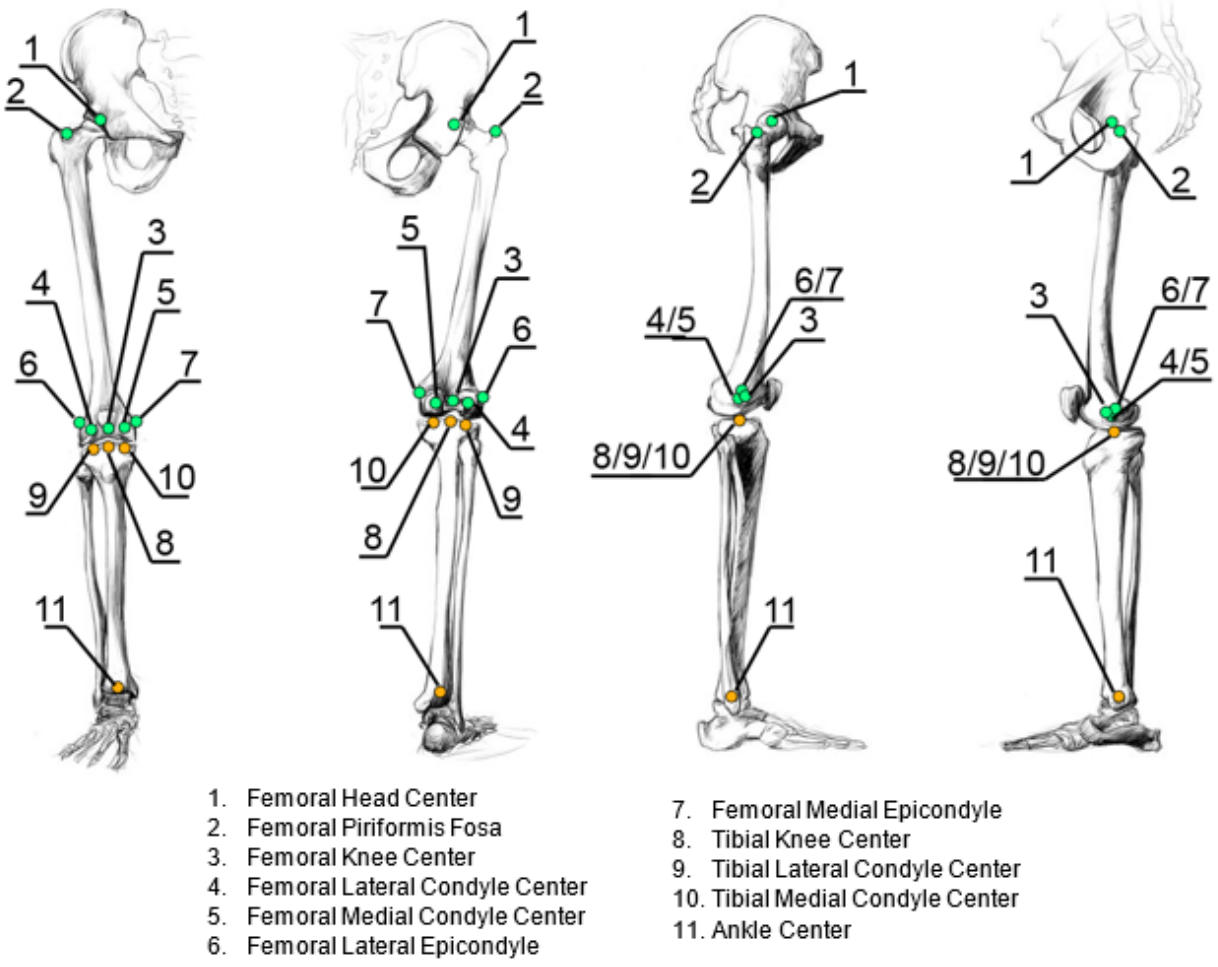

Figure 2.1: Relevant points for the leg in this project.

By defining the anatomical axises for the leg, a coordinate frame for the femur and the tibia can be defined. According to Victor et al [9], this coordinate frame can be defined as:

- Femoral Planes:
  - Frontal Plane: Plane that contains the Femoral Mechanical Axis (FMAx) and is parallel to the line joining the femoral condylar medial and lateral centres.
  - Horizontal Plane: Plane that contains the horizontal axis (perpendicular to FMAx and containing the FKC) and is perpendicular to the frontal plane.
- Tibial Planes:

– Frontal Plane: Plane that contains the Tibial Mechanical Axis (TMAx) and is parallel to the line joining the tibial condylar medial and lateral centres.
– Horizontal Plane: Plane that contains the horizontal axis (perpendicular to TMAx and containing the TKC) and is perpendicular to the frontal plane.

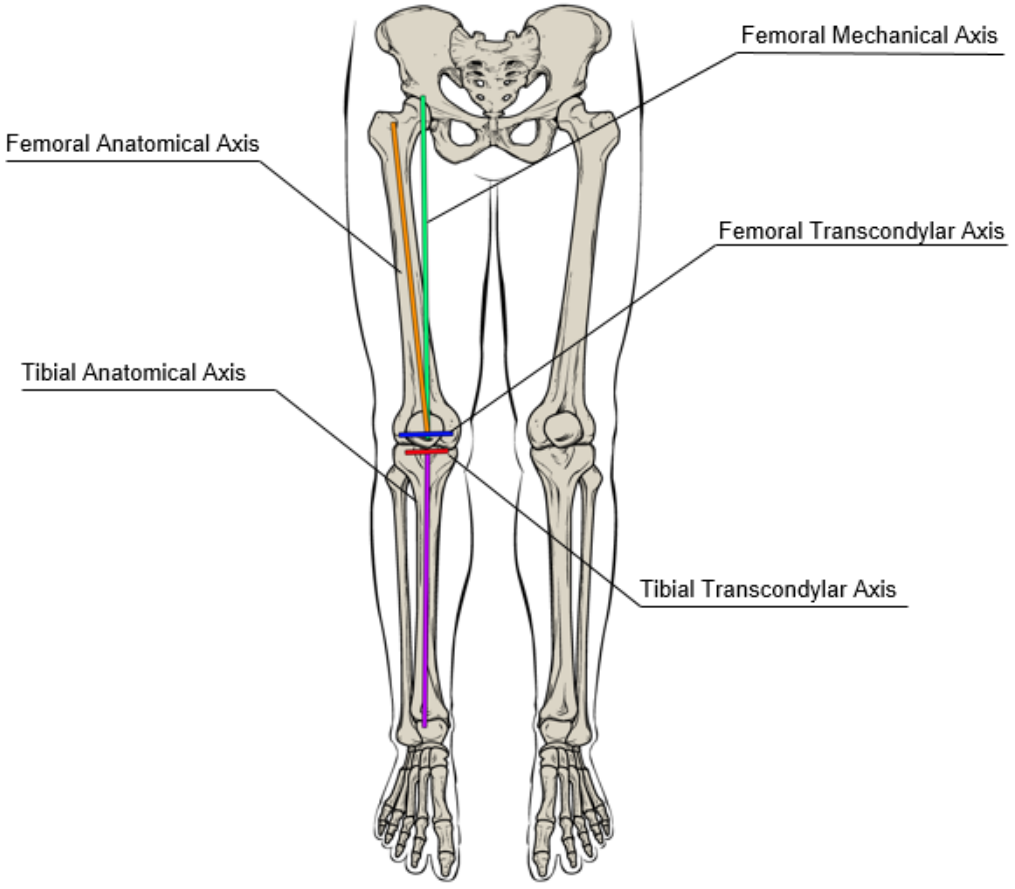

Figure 2.2: Anatomical and mechanical axis for the leg.

2.2.1 The knee

The knee is the largest joint in the body. It is composed by the lower end of the thighbone (femur), the upper end of the shinbone (tibia) and the kneecap (patella). The ends of these bones are covered with articular cartilage in order to prevent bone contact and to enable the bone to move easily. This joint is also composed by the menisci, located between the femur and tibia in order to act as a "shock absorber"; large ligaments (ACL, PCL and more) provide stability between the femur and the tibia; and synovial fluid, a thin lining that covers the bone surface in order to lubricate the cartilage and prevent bone contact.

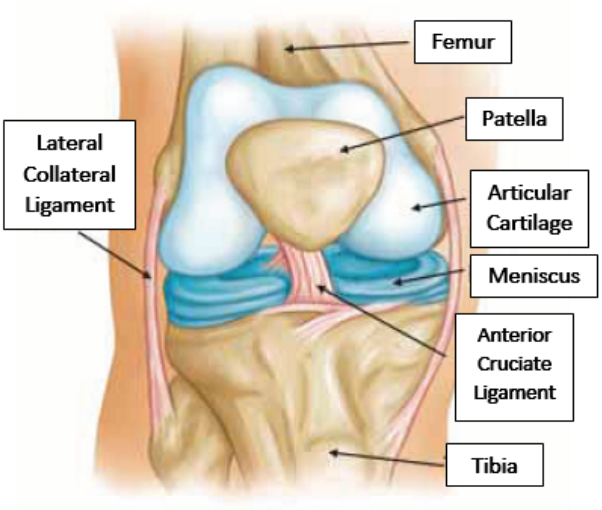

Figure 2.3: Anatomy of the knee

The knee injuries

Having a healthy knee is necessary in order to perform most everyday activities such as walking, jumping, climbing stairs or sitting down/standing up. Normally, all the components of the knee joint work in harmony However, some diseases or injuries can disrupt this harmony, making the subject to feel pain and reduced function of movement. Some of the most common chronic knee diseases are the following:

- Osteoarthiritis: it uses to appear in 50-year-old people or older, but may also occur in younger people. This disease is caused by cartilage wear, and it causes rub against one bone to another, producing knee pain and stiffness.
- Reumatoid arthritis: the most common form of inflammatory arthritis. It produces chronic inflammation of the synovial membrane, causing damage in the cartilage and eventually cartilage loss, pain and stiffness.
- Post-traumatic arthritis: it can appear due to a severe knee injury. Fractures of the bones surrounding the knee or tears of the knee ligaments may damage the articular cartilage over time, causing knee pain and limiting knee function.

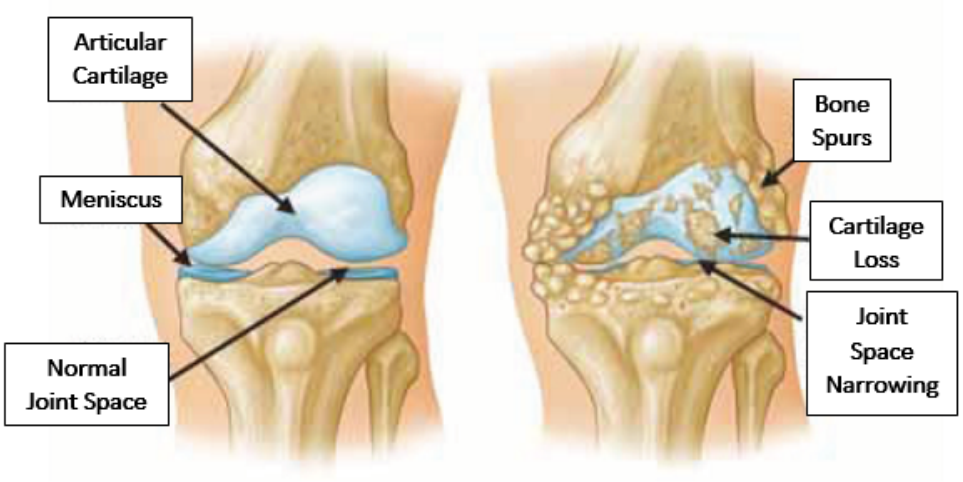

Figure 2.4: Comparison between healthy and arthritic knee

Total Knee Arthroplasty

Total Knee Arthroplasty, also known as Total Knee Replacement or TKA, is a surgical procedure performed in cases on severely damaged knees by arthritis or injuries. This procedure was performed for the first time in 1968 and it is one of the most common surgery nowadays, with more than 600,000 knee replacements every years only in the United States. A TKA consists on a resurfacing on the shape of the knee joint, because only the surface of the lower end of the thighbone (femur) and the upper end of the shinbone (tibia) are actually replaced.

2.2.2 Surgical procedure

In order to perform a TKA, basic steps use to be followed. Although surgical procedures differ on the patient's needs and the surgeon's approach, Bellemans et al [1] describe the general steps as the following:

1. The patient is monitorized in order to check vital signs such as blood pressure, heart rate, body temperature and oxygenation levels. A mark is made on the knee undergoing surgery.
2. Anesthesia is administered. Depending on the patient's health situation and surgeon's approach, either local or general anesthesia may be received.
3. An incision down the center of the knee about 20 to 25 centimeters long is made by the surgeon, and then cuts through deeper tissue, including the quadriceps tendon.
4. In order to access the joint easily, the knee is bended 90 degrees.
5. Once the bone is exposed, the surgeon proceeds to remove the damaged cartilage and bone surface area. The surgeon uses a bone saw to remove the arthritically damaged areas at bottom of the femur and the top of the tibia. Each bone is reshaped to exactly fit its new prosthesis. Because these cuts must be precise, the surgeon uses either a metal jig or computer assistance to line up the cuts.
6. The surgeon may resurface the back of the kneecap, or patella, and attach an implant. A polyethylene component may be attached to facilitate the patella's gliding against the new joint. Research has not shown a significant difference in outcomes for patients who received patella resurfacing and those who did not.
7. Components are attached to the femur and tibia and patella, if applicable. How the components are attached to the bone will depend on what type of component is used. Most knee replacement surgeries use cemented components that are affixed using bone cement. Cemented and cementless components offer different advantages.
8. A flexible cushion made of polyethylene is attached on top of the new tibia surfaces. This spacer acts as a shock absorber between the two new prosthetic surfaces.
9. The leg is flexed and extended to test the fit of the components and the new knee's range of motion.
10. The surgeon straightens the knee to allow the components, cement, and bone to bond together. Because the bone cement is fast acting, this takes only about 10 minutes.
11. The surgeon will repair any deep tissue that was cut during surgery and then stitch the skin at the incision.

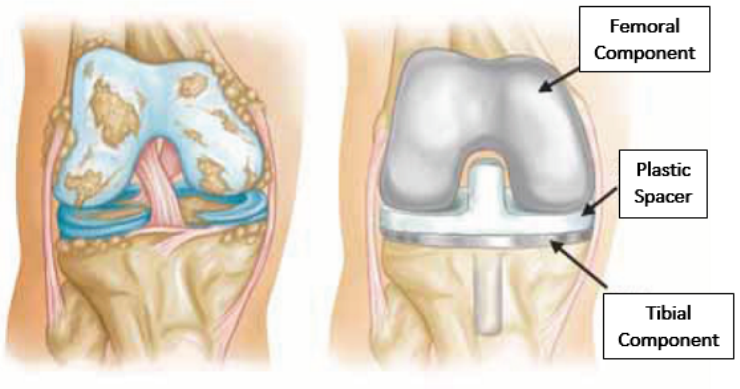

Figure 2.5: Total Knee Arthroplasty

2.2.3 Medical Imaging Review

During the last years, digital imaging field has suffered a rapid evolution. This field has opened a wide new way of processing medical images in order to get faster and more accurate diagnosis procedures and have helped in the implementation of new medical and surgical protocols. Traditional two-dimensional image viewers and image display programs have currently been almost substituted by 3D volume and surface renderers which suppose a faster and more efficient way to analyze large sets of data acquired by tomographic imaging techniques such as MRI and CT. 3D animators such as OpenGL graphic libraries offer a powerful processing tools and hardware acceleration, and open-source visualization libraries such as ITK and VTK provide a fast, flexible and accurate image rendering for volume and surface processing.

That is why these libraries have been used in several 3D Image Rendering projects such as OsiriX, a project of a open-source medical imaging software carried out by Rosset et al [8]. This software provides an intuitive way to explore and analyze medical data from DICOM files acquired by MRI/CT scanning in either two or three-dimensional environment. This software has been designed to work on Windows and MacOS X operative systems.

Another software based on VTK 3D imaging and rendering modules commonly used is 3D Slicer. Fedorov et al [4] describes it as an open-source application for medical imaging that provides versatile visualization and excellent tools in terms of image segmentation.

Even OsiriX or 3D Slicer suppose an efficient tool to display large sets of medical data, they were not designed as softwares for geometrical and property analysis. For that reason this project pretends to be an alternative these applications, being an open-source software developed in Linux OS and based in OpenGL environment and VTK libraries that allows the user to extract measures and geometrical information from DICOM data sets.

Chapter 3

Functional approach

3.1 Requirements model

3.1.1 General requirements

The software has been designed to work in Windows OS. It can be launched by a .exe file which can be supported for many Windows platforms. It is advisable to launch the program with at least 1 GB of RAM memory available. Devices provided with CPUs with 4 cores and above 2.5 GHz of frequency like Intel Core i5 fit better for the activity requirements.

3.1.2 Functional requirements

This software has been designed to be launchable for a single user. It has been supposed that the user reaches a certain level of knowledge in the medical and biomechanics field, and it has been considered that probably the final user of the application is a orthopaedic surgeon. The figure presented below shows a diagram containing all the requirements needed the application to fulfill under the action of the user.

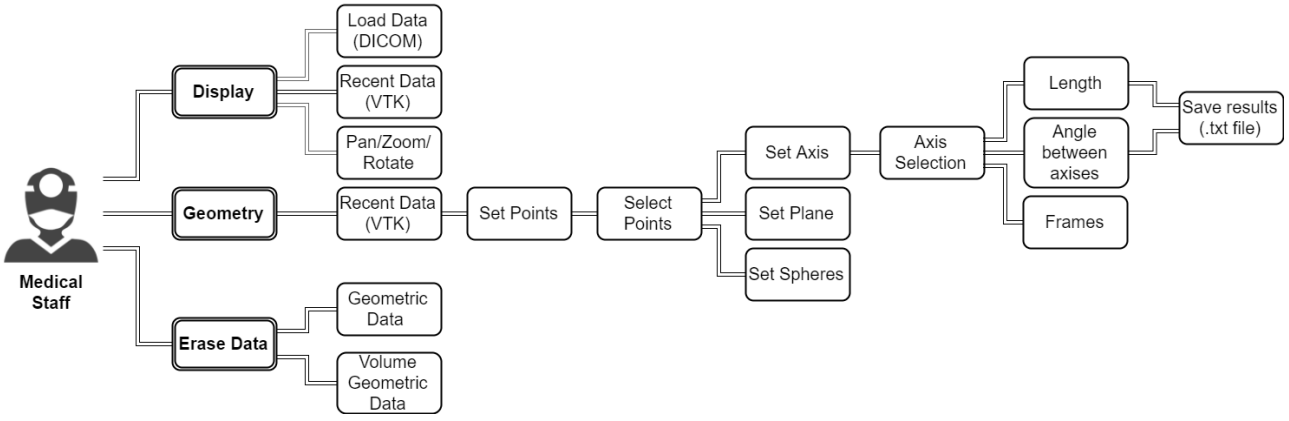

Figure 3.1: Requirements diagram based on the user interaction

3.1.3 Work-flow

This software has been designed in order to allow the user to follow intuitive steps during the volume visualization and geometry extraction. The work flow expected for the user to follow is shown in the following diagram.

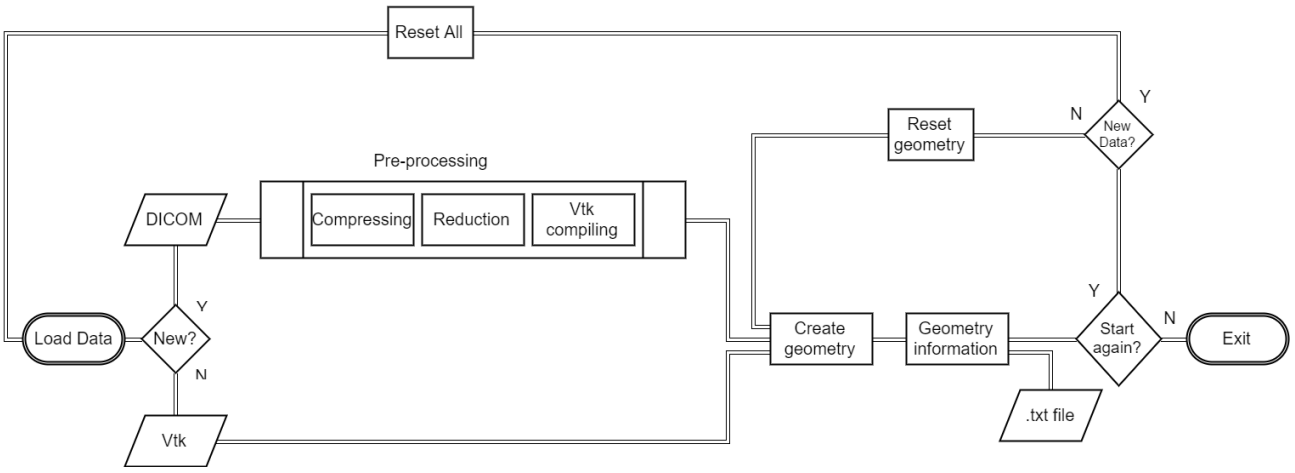

Figure 3.2: Software work-flow diagram

3.2 Design

In order to optimize the launching speed, an object-oriented class computational implementation design has been set. Depending on the relationships in the instances between classes and their invocations, it has been decided to present the class diagram for actors and interface independently.

3.2.1 Actors classes diagram

Actor classes diagram are the ones invoked in order to initialize and set the properties of the volumes, surfaces and actors and render them throughout the interface. They also contain the geometric information needed to extract results of axises lengths and angles.
To build the classes needed to launch the application, Python 2.7 programming language has been used. It has been necessary to inherit the VTK library to create the actors and the mappers connections needed to visualize geometrical data such as points, lines and planes.

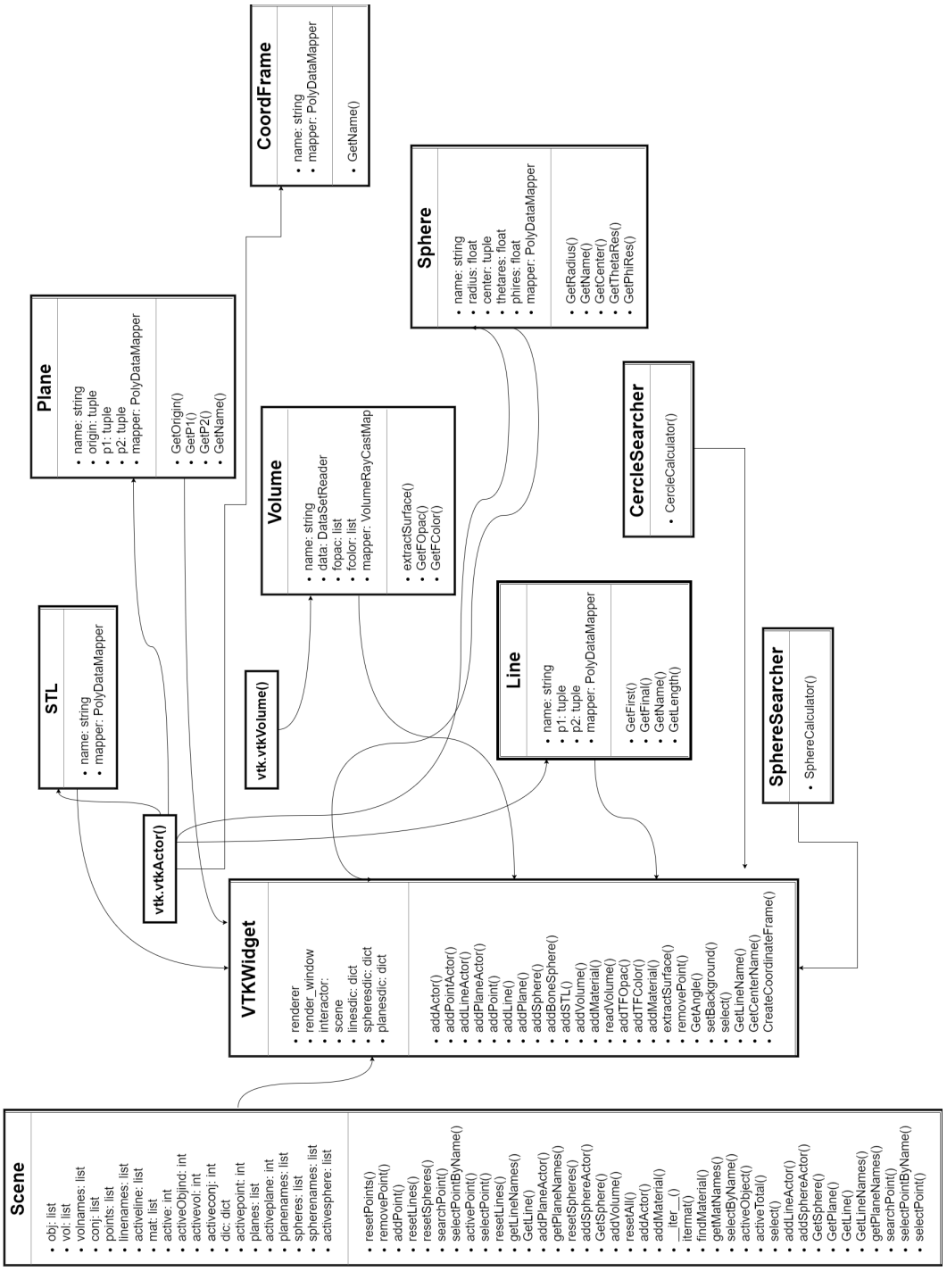

Figure 3.3: Actors classes diagrams. All classes are inherited from VTK library

3.2.2 Interface classes diagram

In this section, the classes diagram for the interface GUIs are presented. The interface has been designed in order to suppose an intuitive connection between the user and the software.

To build the classes needed to launch the application, Python 2.7 programming language has been used. For the interface classes, PyQt4 environment has been used in order to initialize the GUI objects needed to build the external interface such as buttons, layouts and checkboxes.

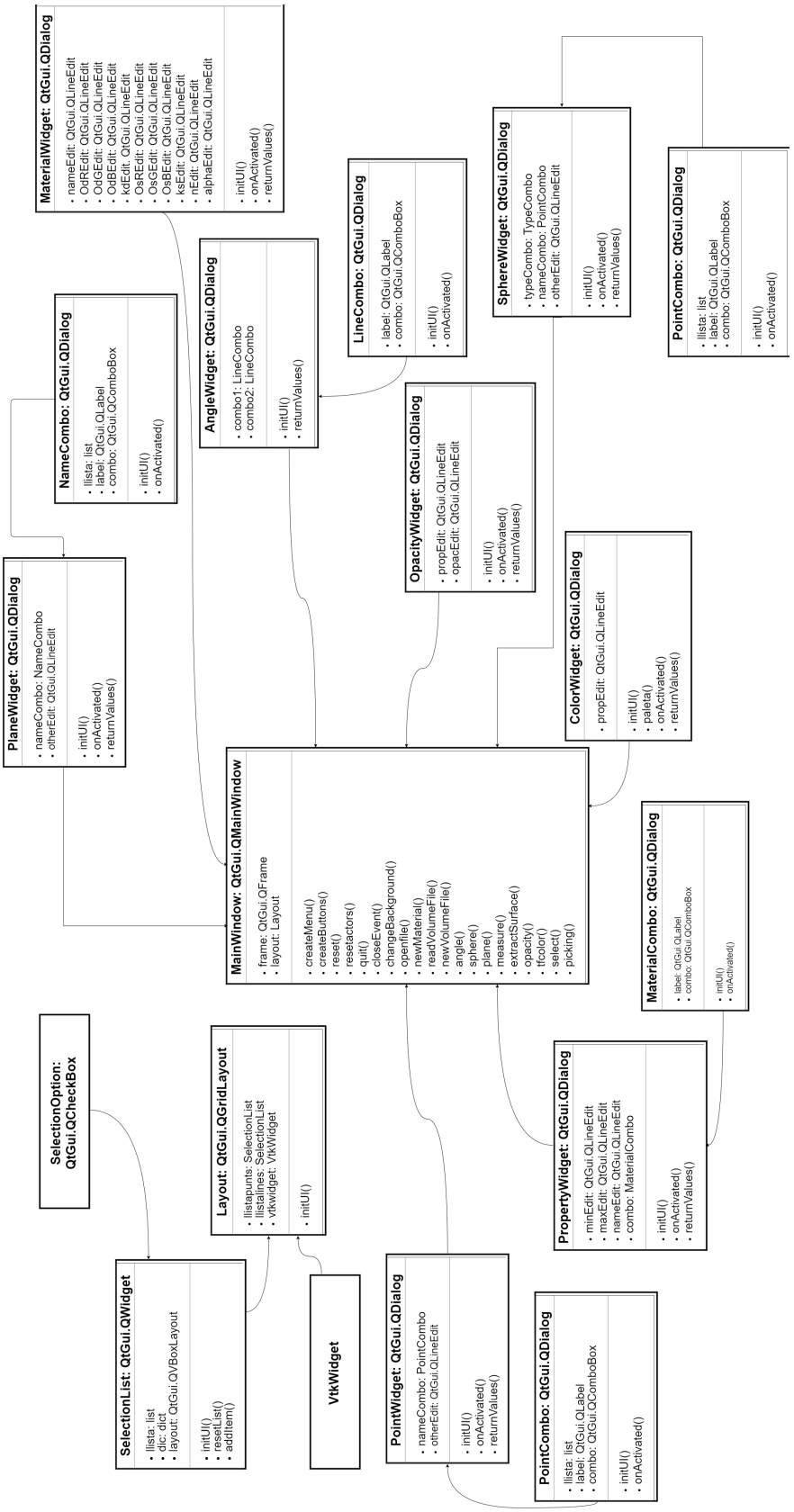

Figure 3.4: Interface class diagram. All classes are inherited from PyQt.QtGui library

3.3 Interface proposal

As explained in previous section, one of the most important objective is to let the user interact with the application in the most ergonomic and suitable way. To do so, it is required for the interface to be intuitive and easy to understand. This is why it has been decided to make a simple interface where the user can access to all the main functions by identifying the function by their name and a simple icon and pressing a set of buttons. To understand the performance of the interface, the following figure shows the design for the screen once the user launches the application.

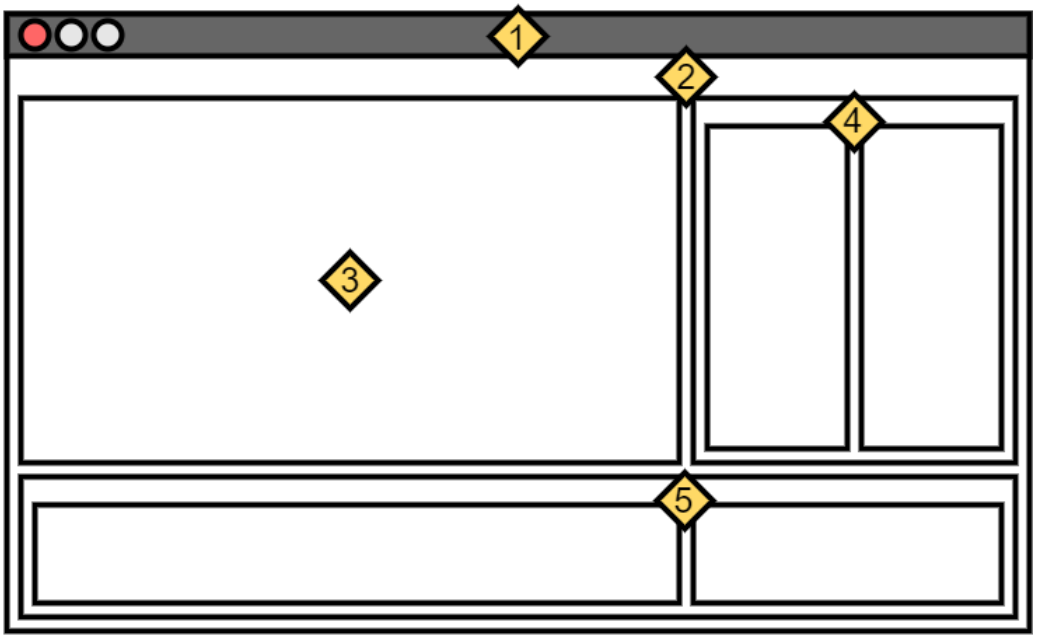

Figure 3.5: Proposed design for the application.

In Figure 3.3 different widgets can be identified. Each of these widgets realizes their proper functions and they are all needed to make the user understand the performance of the application and the identification of the elements.

1. Main Window: it contains the layout and sets the upper menu and the exit function.
2. Layout: it contains the rest of the elements of the application.
3. Frame: it contains the rendering information. It inherits the elements needed to render the data from VTK libraries from its attribute vtkwidget, which contains the renderer, the render window and the rendering interactor. Once the data is loaded, the frame renders the CT scan information as volumetric data, and lets the user edit the camera position by zooming/panning/rotating in order to get the best visual approach of the volume.
4. Selection Lists Workspace: it contains information about the active actors in the frame, such as selected points from the volume and created lines, planes and spheres. It can be divided in two classes:
    - Points Selection List: it contains the name of the current active selected points from the volume. Points can be checked in order to selected which points will be included in the creation of actors.
    - Actors Selection Lists: it contains the name of the current active actors in the frame. The term actor refers to any 1D/2D/3D entities based on geometrical properties (vertexes and their connections) and rendered in the frame following their material properties.

5. Buttons workspace: it contains the PushButtons that call for the main functions of the application. Like the Selection List Workspace, it can be divided in two classes, depending on the buttons' function:

    - File Buttons Workspace: it contains the set of buttons whose functions are related with the CT data extraction and rendering, either if it is a new file (New Data will realize the data pre-processing in order to build a .vtk renderable file), a previous set of CTs (Load Data will render an existing .vtk file) or resetting the existing volume if needed.
    - Actor Buttons Workspace: it contains the set of buttons whose functions are related with the extraction of geometrical information, such as the selection of points in the volume, the creation of lines, planes and spheres, and the erasing process of all of them.

Once the application is launched, the user can easily identify the main functions that have been set to be needed to access in previous sections. The user firstly loads the patient's data, either by a new data set by clicking in New Data or to an existing VTK file of a previous CT data set. Once the physiological data is loaded and the bones are rendered, the user sets the points needed to extract geometrical information by Select Points function. By adding points, the user creates lines, planes and spheres. All geometrical actors (points, lines, planes, spheres) are added in their respective Selection Lists in order to keep the preferred actors selected.
The following figure shows the interface appearance once physiological and geometric data have been loaded and created.

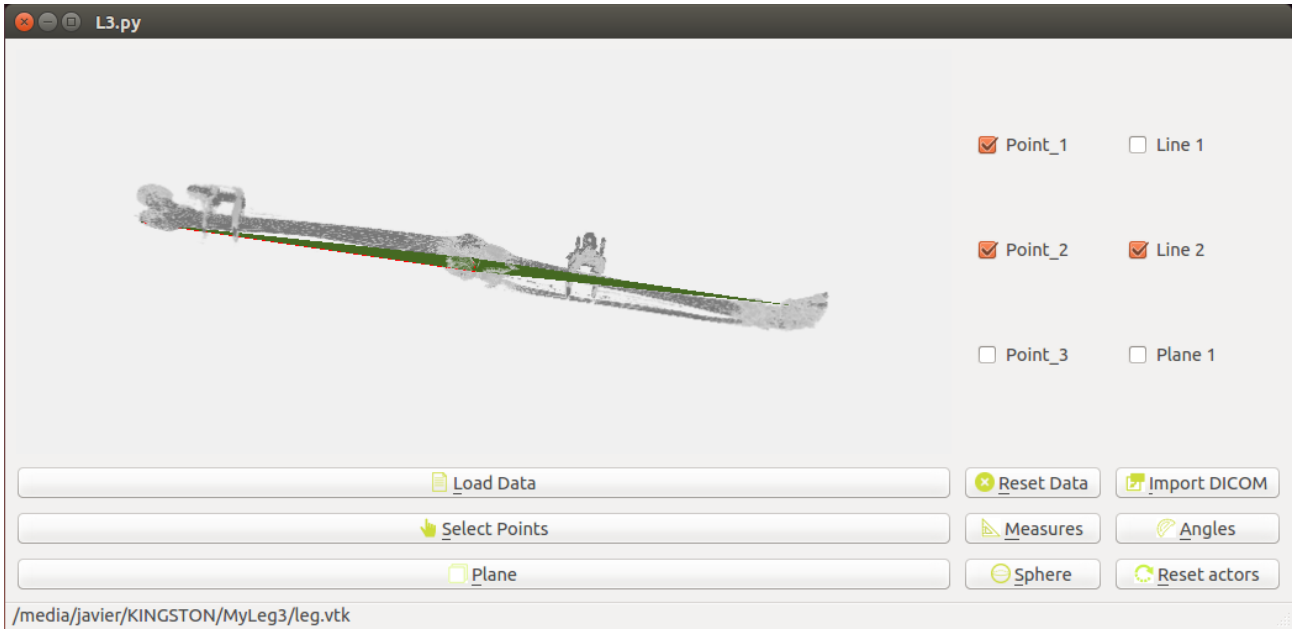

Figure 3.6: Application interface once CT scans have been loaded and actors have been set.

Chapter 4

Implementation

4.1 Libraries and programming tools

In this section it has been detailed which libraries and extensions were necessary in order to set the programming routines, the inheritations and the external modules imported.

1. GNU/Linux as Computer Operative System. This OS was originally developed as a free operating system for personal computers. It was first based on the Intel x86 architecture, and since its creation it has been ported to more computer hardware platforms than any other operative system. Its intuitive programming environment and its collaborative programming policy has let Linux to be the main operative system used in a wide range of smartphones, tablets, network routers, facility automation controls, televisions, video game consoles and smartwatches.

2. Python 2.7 as Programming Language. Python is one of the most widely used dynamic programming language. Its best programming quality is its design philosophy, which emphasizes code readability. Its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. It has been oriented to support object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Although it was initially designed to work in Linux OS environment, Python interpreters are available for many operating systems. Third-party tools, such as Py2exe or Pyinstaller can package Python code into stand-alone executable programs for operative system such as Windows or MacOS X, so Python-based software can be distributed to, and used on, those environments with no need to install a Python interpreter.

3. GNU Emacs 24 as programming text editor. GNU Emacs is the most popular and most ported Emacs text editor. It is directly included in GNU/Linux Operative Systems and can display files in multiple character sets, and has been able to simultaneously display most human languages.

4. PyQt4 as GUI toolkit. It is a cross-platform for GUI programming, and as well as Python it supposes a open-source software to build hundreds of classes, GUI widgets included.

5. VTK as 3D visualization toolkit. The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing, and visualization. It consists of a C++ class library and several interpreted interface layers including Tcl/Tk, Java and Python. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods,

as well as advanced modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation.

4.2 Pre-processing

4.2.1 Characteristics of the datasets

As explained in previous sections, this application has been first conceived as a visualization platform for medical data. Following the requirements model that has been designed, Omega has been built in order to extract volumetric information from DICOM data set from CT leg scans. The following table resumes the caractheristics expected from DICOM datasets given.

Source	ULB Bio-, Electro- And Mechanical Systems Department
Modality	CT
Object	Right/Left leg of a cadaver
Intensity	2 bytes
Resolution	512x512xLength

Table 4.1: Characteristics of the DICOM dataset. The length of the data set is dependant on the number of DICOM files, in function of the subject's leg length.

4.2.2 Pre-processing steps

In case the user calls the loading of a new DICOM data set, it has been decided to realize a pre-processing procedure in order to convert the multiple DICOM data files into a single VTK file. The reason is that, once the data is loaded and renderer, VTK files allows a faster and more accurate visualization, and provides memory saving instead of managing the rendering of thousands of DICOM slices. To do so, a three-steps pre-processing procedure has been designed. This procedure is carried out when PreProcessing.py is called for the main program. The pre-processing stages are the following:

1. Identification step. Once the foldername has been asked by the interface to the user, PreProcessing.py accesses to the data allocated in the folder by Glob library. Then it gets the name of all the files inside the folder. Then the code identifies only the CT files by the file names (it has been specified that the files contain the term 'CT' in their names), and sorts them into the correct order for the data set. Dataset pixel spacing and total resolution are also stored.

2. Extraction and reduction step. Once the DICOM files are identified, the code initializes a numpy dataset array. In order to reduce the final VTK size, it has been decided to erase the environmental information from the DICOM and access directly in those pixels where bone data is storaged. This numpy array is then set to be 200x200xLength sized. Then the DICOM pixel data is extracted and added to the numpy array, with a previous reduction of the pixel intensity from 2 bytes to 1 byte. It has been decided to realize this reduction step because the final file size was still too big to allow a rapid data loading and visualization.

3. VTK Compiling. Once the numpy array allocates the proper pixel data from DICOM data sets, it was then necessary to build the .vtk file. To do so, the previous study of the DICOM data set resolution was needed in order to set the correct header for the VTK file. Once the Pixel Spacing, the DICOM data set dimension and the number of points were calculated, the header is completed and added to the numpy array and the .vtk file is created. This file is automatically saved in the current folder for future possible analysis and then the VTK is rendered.

The following figure shows the work flow for the pre-processing algorithm with the main necessary variables.

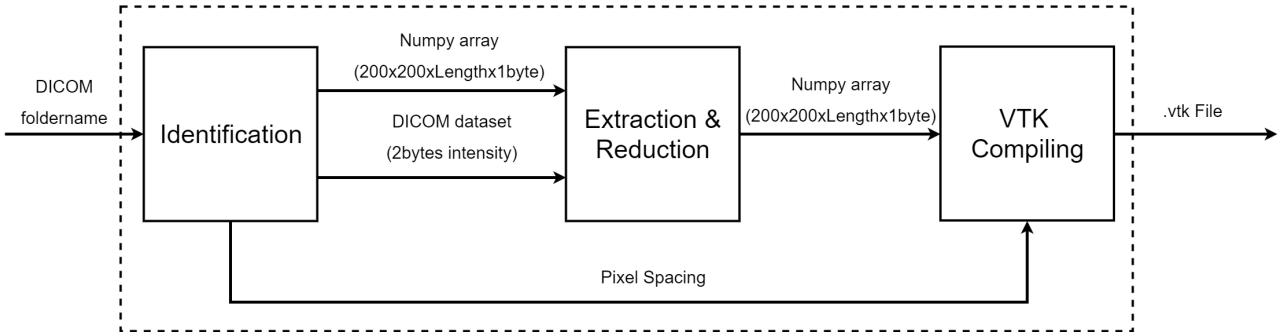

Figure 4.1: Pre-processing flow diagram.

4.3 Selection Tool

In this section, point selection from the 3D volume has been detailed. In order to give the user a fast and easy procedure to pick the desired pixel from the volume, some modifications have been needed to apply to the application interactor.

As interactor allows the user to rotate/pan/zoom the volume under the actions of the left and central mouse buttons, the action of the right one has been modified. Once the Select Points button in the interface is pressed, right button has been set to initialize the picking event.

During the picking event, vtkCellPicker class is inherited. As its own definition, vtkCellPicker shoots a ray into a 3D scene and return information about the first object that the ray hits. For vtkVolume objects, it shoots a ray into the volume and returns the point where the ray intersects an isosurface of a chosen opacity (see Figure 4.3). In this case, the chosen has been set as the bone surface opacity value, so the picker gets the information about the superficial pixel of the bone where the ray hits. As it has been specified for this application to allocate points on the bone surface, this method gets valid results.

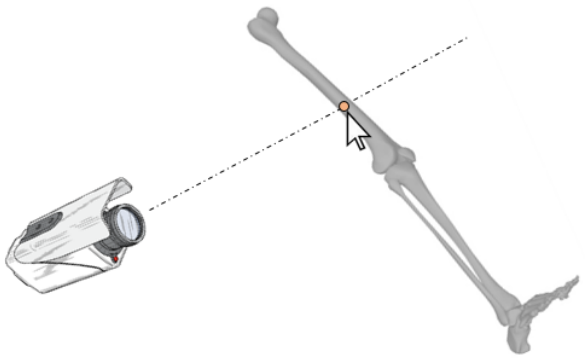

Figure 4.2: Cell picking procedure as the intersection between the bone volume and the raycast from the render camera.

The picking instance in this application has been used to extract the global position of the selected point. This information has been sent to VtkWidget class in order to render a point, added in the layout as a small sphere, on the bone surface. The knowledge of the position of the point selected was also used as inputs for upcoming geometrical data processors such as rendering planes, spheres, etcetera.

4.4 Geometrical Analysis

4.4.1 Planar Analysis

Plane Analysis

In order to visualize mechanical and anatomical planes given the geometrical information from points of the leg surface, planar analysis can be used. With this action, a subroutine for the calculation of the plane that contain three selected points P_i from the renderer is called. It automatically calculates the geometrical information for the plane and renders it, as it follows:

$$P_i(x_i, y_i, z_i), \qquad i = 1, ..., 3$$

$$\pi : Ax + By + Cz = D; \qquad \left\{ \begin{array}{c} A = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \\ B = (x_3 - x_1)(z_2 - z_1) - (x_2 - x_1)(z_3 - z_1) \\ C = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \\ D = Ax_i + By_i + Cz_i \end{array} \right\} \quad (4.1)$$

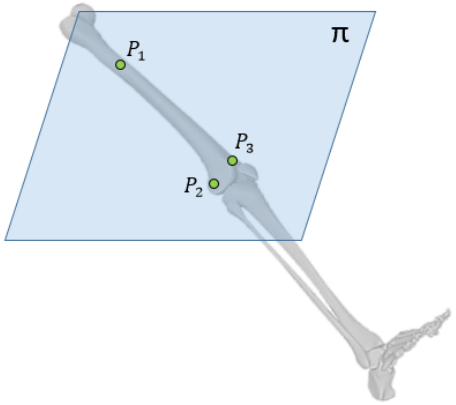

Figure 4.3: Plane calculation given 3 points.

Frame Analysis

Frame rendering is an automatic tool based on planar analysis. The user is able to get the main frames for the femur and tibia by calling the subroutine Frame. This subroutine needs for the render to have some points in its Point List to work. These points are the ones that define the frames for the femur and tibia, following the definitions given by Victor et al [9].

4.4.2 Analysis by interpolation

Circle Analysis

In order to access to the geometrical information of internal points of the leg, such as the Ankle Center, circular analysis can be used. With this action, based on the geometrical information from three points on the surface of the tibia, a subroutine for the calculation of the center of the circle that contains these three points is called. Having selected this three points P_i from the tibial surface, the cercle center is calculated as it follows:

$$P_i(x_i, y_i, z_i), \qquad i = 1, ..., 3$$

$$\pi : Ax + By + Cz = D; \qquad \left\{ \begin{array}{c} A = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \\ B = (x_3 - x_1)(z_2 - z_1) - (x_2 - x_1)(z_3 - z_1) \\ C = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \\ D = Ax_i + By_i + Cz_i \end{array} \right\}$$

$$\pi_{ij} : A_{ij}x + B_{ij}y + C_{ij}z = D; \qquad \left\{ \begin{array}{c} A_{ij} = x_i - x_j \\ B_{ij} = y_i - y_j \\ C_{ij} = z_i - z_j \\ D_{ij} = A_{ij}x_{ij} + B_{ij}y_{ij} + C_{ij}z_{ij} \end{array} \right\}$$

$$\text{C: } \left\{(x_0, y_0, z_0) \in R^3 | Ax + By + Cz = D, A_{ij}x + B_{ij}y + C_{ij}z = D_{ij}, i \neq j, ij \neq ji\right\}$$

$$\left\{ \begin{array}{c} A_{12}x_0 + B_{12}y_0 + C_{12}z_0 = D_{12} \\ A_{23}x_0 + B_{23}y_0 + C_{23}z_0 = D_{23} \\ Ax_0 + By_0 + Cz_0 = D \end{array} \right\} \Rightarrow MC = D \Leftrightarrow C = \left\{ \begin{array}{c} x_0 \\ y_0 \\ z_0 \end{array} \right\} = M^{-1}D$$

$$\text{M}= \begin{bmatrix} A_{12} & B_{12} & C_{12} \\ A_{23} & B_{23} & C_{23} \\ A & B & C \end{bmatrix}; \qquad \text{D}= \begin{bmatrix} D_{12} \\ D_{23} \\ D \end{bmatrix} \quad (4.2)$$

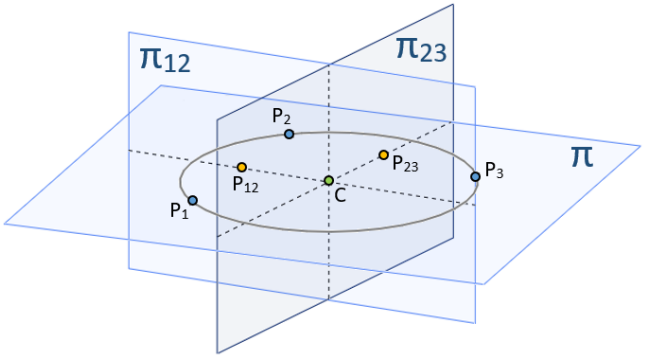

Figure 4.4: Sphere center calculation given 3 points.

Sphere Analysis

In order to access to the geometrical information of internal points of the leg, such as the Femoral Transcondylar Centers, spherical analysis can be used. With this action, based on the geometrical information from four points on the surface of the femoral condyles, a subroutine for the calculation of the center of the sphere that contains these four points is called. Having selected this four points P_i from the condylar surface, the sphere center is calculated as it follows:

$$P_i(x_i, y_i, z_i), \qquad i = 1, ..., 4$$

$$\text{P}_{ij}(x_{ij}, y_{ij}, z_{ij}), \qquad k_{ij} = \frac{k_i + k_j}{2}, \qquad k = x, y, z$$

$$\pi_{ij} : A_{ij}x + B_{ij}y + C_{ij}z = D; \qquad \left\{ \begin{array}{c} A_{ij} = x_i - x_j \\ B_{ij} = y_i - y_j \\ C_{ij} = z_i - z_j \\ D_{ij} = A_{ij}x_{ij} + B_{ij}y_{ij} + C_{ij}z_{ij} \end{array} \right\}$$

$$\text{C: } \left\{(x_0, y_0, z_0) \in R^3 | A_{ij}x + B_{ij}y + C_{ij}z = D_{ij}, i \neq j, ij \neq ji\right\}$$

$$\left\{ \begin{array}{l} A_{12}x_0 + B_{12}y_0 + C_{12}z_0 = D_{12} \\ A_{23}x_0 + B_{23}y_0 + C_{23}z_0 = D_{23} \\ A_{34}x_0 + B_{34}y_0 + C_{34}z_0 = D_{34} \end{array} \right\} \Rightarrow MC = D \Leftrightarrow C = \left\{ \begin{array}{l} x_0 \\ y_0 \\ z_0 \end{array} \right\} = M^{-1}D$$

$$\text{M}= \begin{bmatrix} A_{12} & B_{12} & C_{12} \\ A_{23} & B_{23} & C_{23} \\ A_{34} & B_{34} & C_{34} \end{bmatrix}; \qquad \text{D}= \begin{bmatrix} D_{12} \\ D_{23} \\ D_{34} \end{bmatrix} \quad (4.3)$$

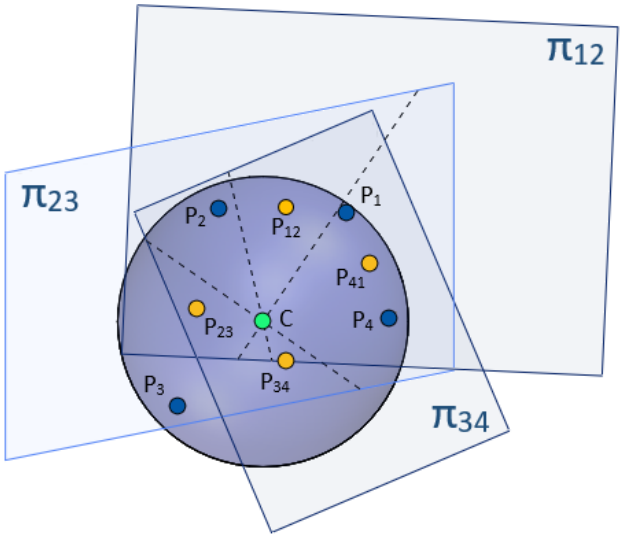

Figure 4.5: Sphere center calculation given 4 points.

In order to get feasible results for the Transcondylar Centers, it is necessary to select the external points accurately. This procedure should be realised by the surgeon or any other medical staff with anatomical knowledge.

Chapter 5

Results

5.1 User Validation

This software was conceived as an easy way for the user to understand and work with medical data. After designing and implementing the code and routines, work-flow validations for the user was necessary in order to identify possible errors and improvements.
After several compilations, debugging and improvements, a visual and intuitive interface was design. Following some simple rules, the user can manipulate the application from the beginning (loading medical data) to the end (extracting geometrical information). For the loading process, some pre-processing procedures were needed to implement. Starting from DICOM datasets, a subroutine to generate a .vtk format file is called, as explained in previuos sections. This pre-procedure takes some time, depending on the size of the DICOM datasets.
After optimizing this subroutine, it was achieved to complete this procedure after a mean value of 4 minutes. The rest of the possible operations in the application (to load .vtk files, pan/zoom/rotate, select points, create geometrical actors, etc.) are almost instantaneous.
This software implies a wide improvement comparing with other ways of analysis. The analysis of the geometrical information from patients in situ would require to open up the patient in surgical operation and take measurements directly from the bone surface. This would imply much more time in the analysis, a not so good approach of visualization of the leg from orientation and a high risk for the patient's health.

5.1.1 Geometrical Validation

One of the main objectives of this project was to design an application that could give the surgeon a reliable geometrical approach of the anatomy of the leg.
In order to validate the information given by the software, four different legs from cadavers were analyzed with this application using DICOM datasets from CT scans.
The pattern was the same for all the samples:

1. Superficial points were selected by the Selection Tool, such as FKC, TKC and FPF.
2. Internal bone points were calculated by using Polar Tools (Circle/Sphere Analysis), such as FLCC, FMCC, FHC or AC.
3. Femoral and Tibial main axis (FMA, FAA, TAA, FTA and TTA) were calculated using the points above and the Line Tool.
4. From the position of every point, axis lengths and angles were calculated using Length and Angle Tools.

5. Results were automatically written in a .txt file in order to be imported into an Excel file.

The main mechanical and anatomical axis for these legs were calculated, and the main leg angles were compared with the mean values given by medical bibliography. Figure 5.1.1 shows the angles included in the analysis.

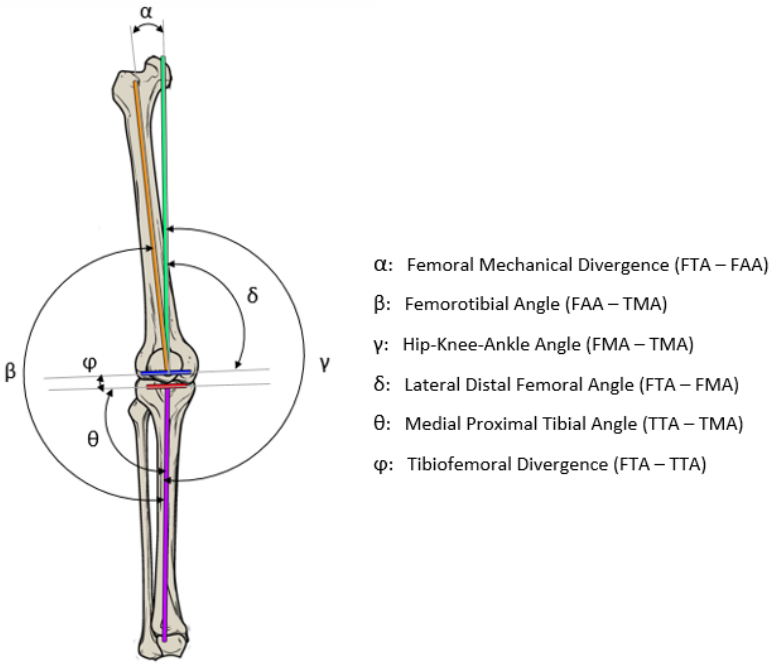

Figure 5.1: Main leg angles defined by femoral and tibial axis.

For the third and fourth leg, FTA was calculated taking the femoral epicondyles. This approximation was done because it can be assumed that the line connecting the two epicondyles and FTA are almost parallel, so the angle with the other axis would be closely similar, making the calculations faster and cancelling the possible calibration errors from the Sphere Calculation.

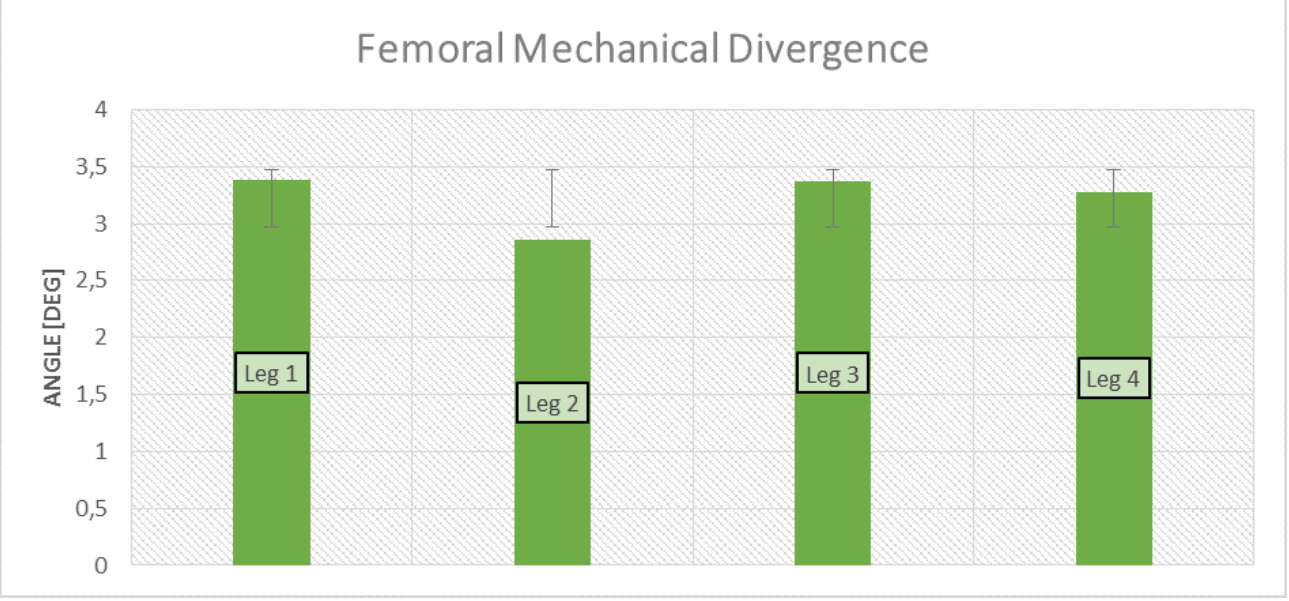

Figure 5.2: Results for the calculation of the Femoral Mechanical Divergence.

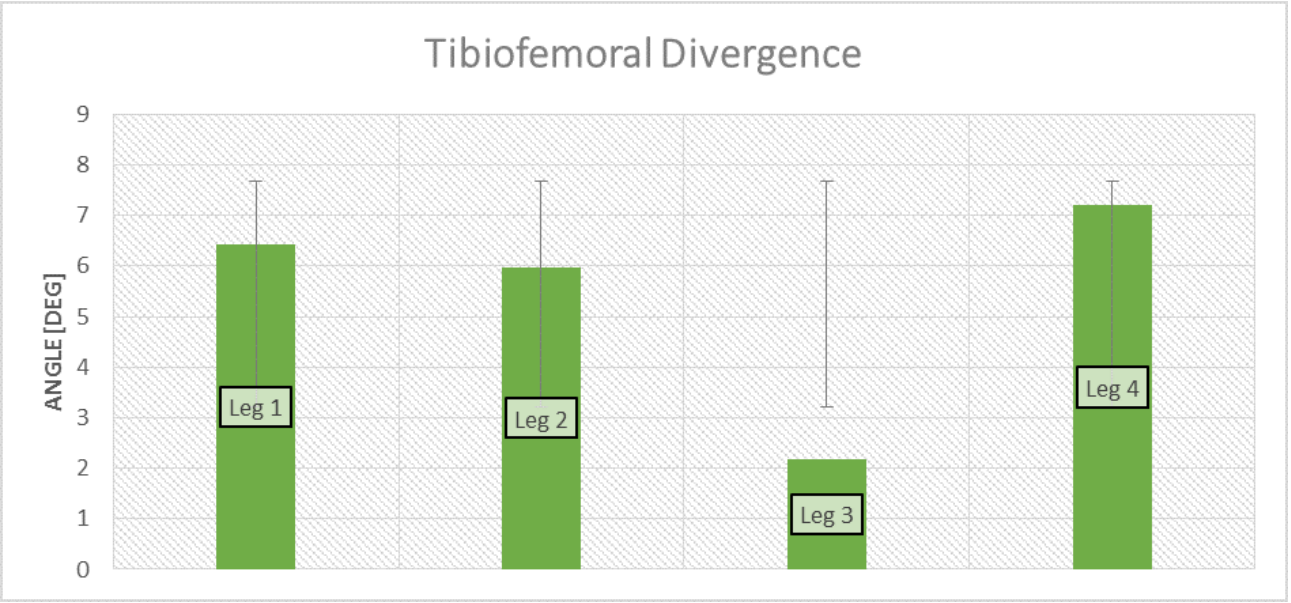

Figure 5.3: Results for the calculation of the Tibiofemoral Divergence.

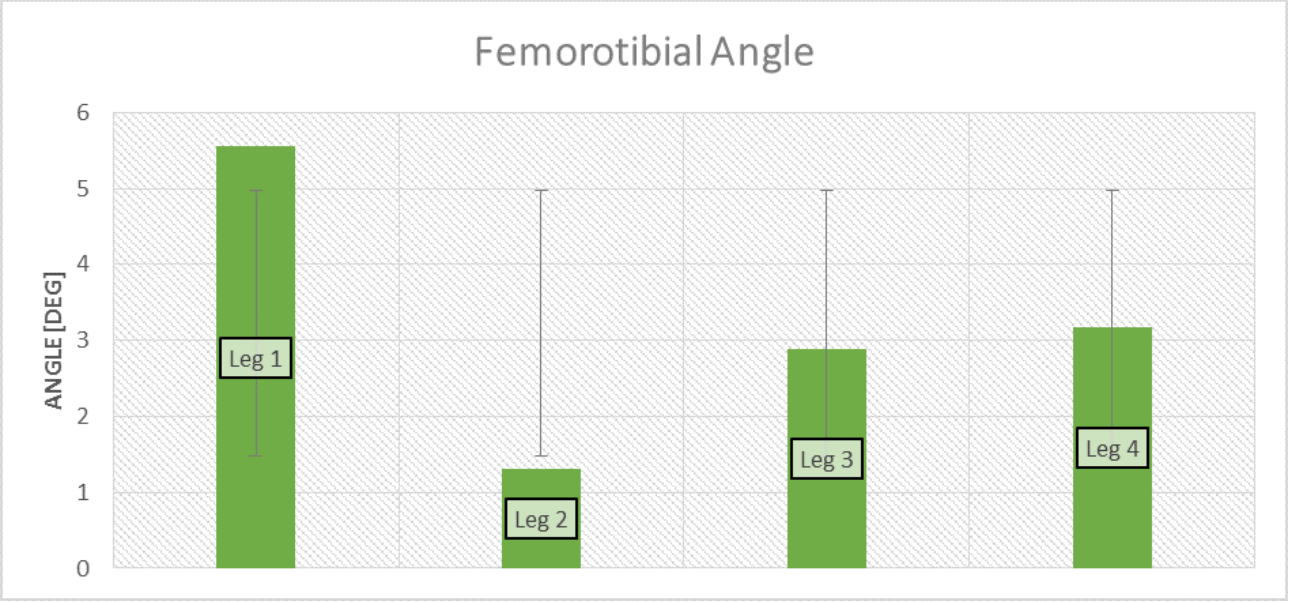

Figure 5.4: Results for the calculation of the Femorotibial Angle.

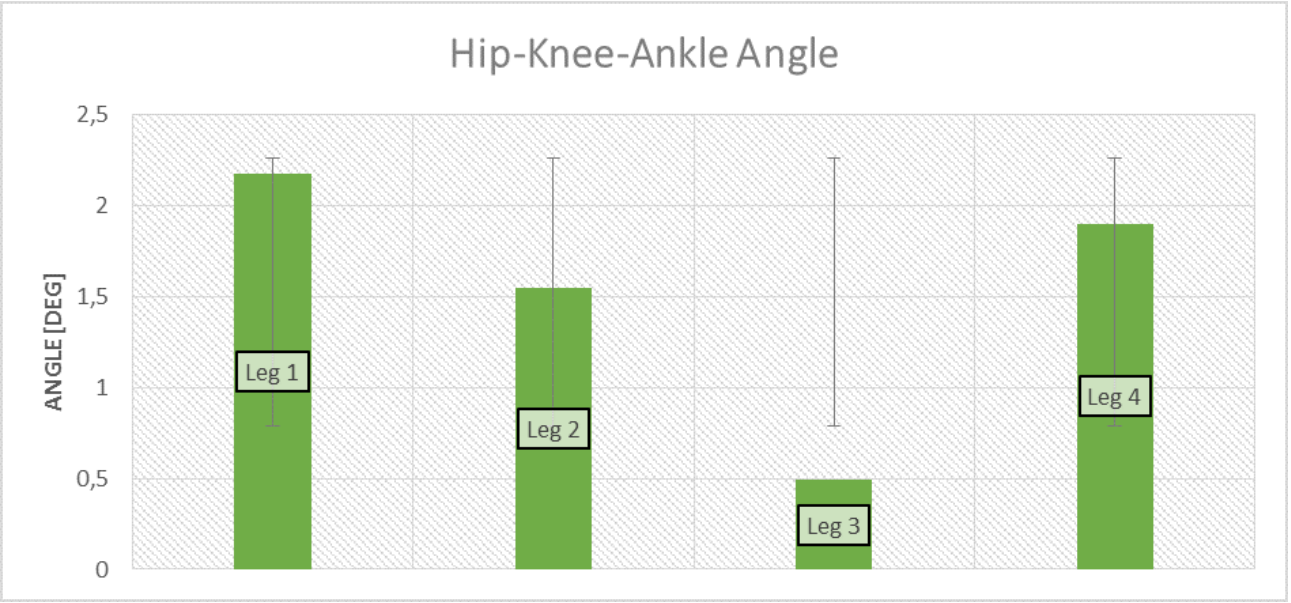

Figure 5.5: Results for the calculation of the Hip-Knee-Ankle Angle.

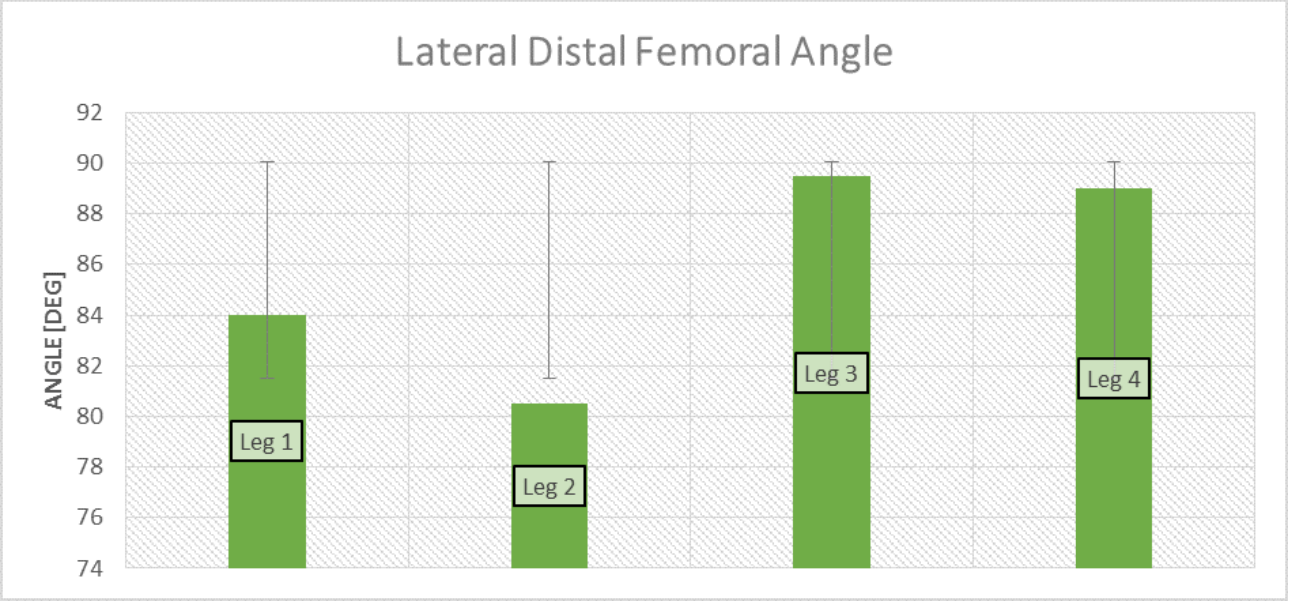

Figure 5.6: Results for the calculation of the Lateral Distal Femoral Angle.

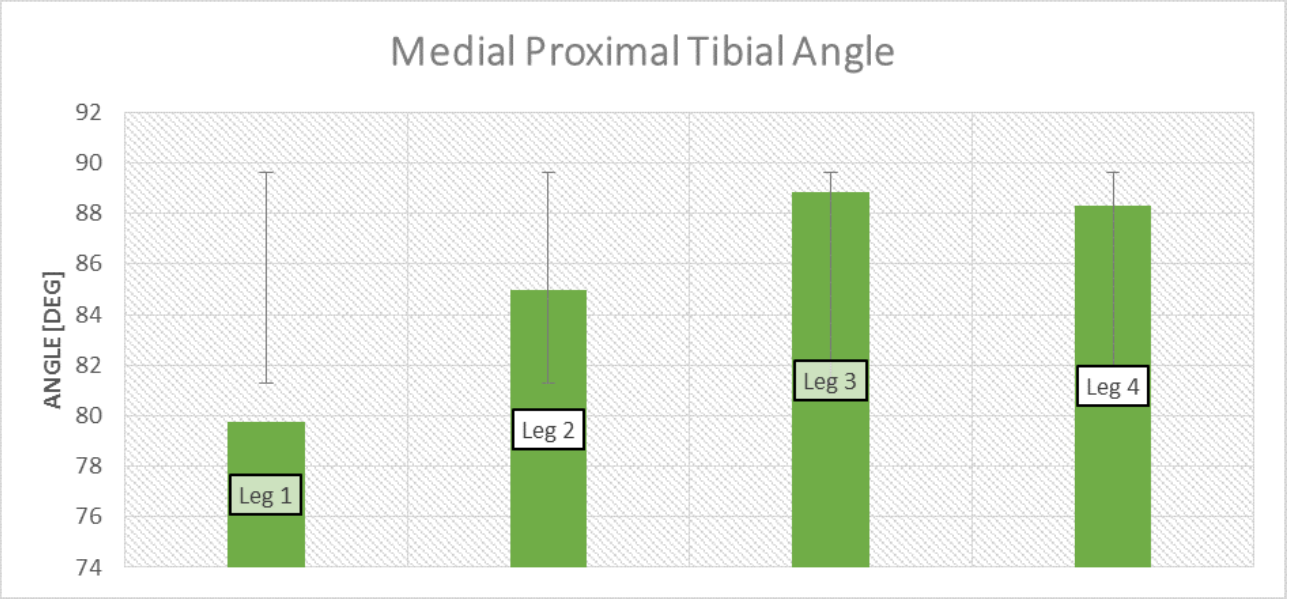

Figure 5.7: Results for the calculation of the Medial Proximal Tibial Angle.

To sum up all the conclusions that can be extracted for the graphics above, there were some aspects to highlight. First of all, graphics showed the same tendency in values for the different classes of angles calculated. This indicated that the geometrical procedures carried out by the software to determine the angles (length, vector norm, scalar products) worked independently of the axis in study. Also, it implied that the implementation of the Selection Tool, as well as the Interpolation Tools such as Circles and Spheres Creators, worked as expected. However, the results showed that not all of them are statistically significant depending on the sample and angle type. This could be explain as a consequence of the bad calibration of the Interpolation Tools. If the external points were selected improperly, the interpolated centers were not positioned realistically, and the axis implied were not well oriented. This results, then, could be improved by increasing the number of samples studied and also by improving the calibration of the external points selected.

Chapter 6

Conclusions and future work

This project was conceived as a Master Thesis in the field on the Civil Engineering, Biomedical specialization under supervision both the Universite Livre de Bruxelles and the Universitat Politecnica de Catalunya during a Master Erasmus exchange.
This project was born under the need of a orthopaedic surgeon to possess an open-source free software that could visualize and take measurements of medical images and decide his surgical procedure criteria.
Finally, taking into consideration the objectives described in previous sections, a GUI application based on visualization of medical data was designed and implemented.
Although this software was conceived as a tool for the analysis of the leg in Total Knee Arthroplasties, its construction allows quick and easy modifications in its structure in order to study other bones of the body such as the arms or head.
This application was designed in Python programming language to be launched in Linux environments. To implement all the visualization modules, actors and the renderer, it was needed to design the software using VTK and PyQt4 open-source modules and libraries.
In order to optimize the time of the loading the DICOM datasets, a pre-processing module was needed to design. With this procedure the reading time decreased from 17 minutes to 4 (mean values), although it implied a loss of pixel resolution by the reduction of the size of the dataset (from 2 bytes to 1 single byte per pixel of data) and creation of a .vtk format file.
In order to render the bone surface, some additions and modifications on the Color and Opacity Transfer Functions of the volume rendering were necessary. To access to the pixel information, it was required to modify the user-interface interactor by adding to the Right Mouse Button the property of selecting points using vtkCellPicker.
After programming, debbugging and testing the different libraries and actions implied, some trials for the calculation of angles between femoral and tibial axis. Results showed that all the calculations followed the same tendency, although a better calibration in the calculation of internal points (such as femoral epicondyles) should be more precise in some cases. This could be solved by selecting superficial points that adjust the interpolation spheres in a more realistic way.
Although the resulting software fulfills all the preliminar objectives, there are some aspects that can be improved in the future. First of all, the software should be more accessible being adapted to work under other platforms such as Windows or MacOS. A better visualization of the knee in terms of rendering could be tried. Automatic calculations for the femoral and tibial frames could be implemented as well. In terms of reliability, trials should be improved by the analysis of larger groups of samples in order to minimize the effect of the imprecision in calibration. Finally, the software should be validated and evaluated, by other specialist of medical and biomedical disciplines, so they could add other features to improve.

Bibliography

[1] J. Bellemans, M. Ries, and J. Victor. *Total Knee Arthroplasty: A Guide to Get Better Performance*. Springer, 2 edition, 2005.

[2] J. Cherian, B. Kapadia, S. Banerjee, J. Jauregui, K. Issa, and M. Mont. Mechanical, anatomical, and kinematic axis in TKA: Concepts and practical applications. *Current Reviews in Musculoskeletal Medicine*, 7:89–95, 2014.

[3] R. Decking, Y. Markmann, J. Fuchs, W. Puhl, and H. Scharf. Leg axis after computer-navigated Total Knee Arthroplasty. *The Journal of Arthroplasty*, 20:282–288, 2005.

[4] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Fillion-Robin, S. Pujol, C. Bauer, F. Jennings, D.and Fennessy, M. Sonka, J. Buatti, S. Aylward, J. Von Miller, S. Pieper, and R. Kikings. 3D Slicer as an image computing platform for the Quantitative Imaging Network. *Magnetic Resonance Imaging*, 30:1323–1341, 2012.

[5] S. Pickering and D. Armstrong. Focus on alignment in Total Knee Replacement. *The Journal of Bone and Joint Surgery*, 2012.

[6] PyQt4 Reference Guide. `http://pyqt.sourceforge.net/Docs/PyQt4/`.

[7] Cory Quammen. Scientific data analysis and visualization with Python, VTK, and Paraview. *Proceedings of the 14th Python in Science Conference (SciPy 2015)*, pages 32–39, 2015.

[8] A. Rosset, L. Spadola, and O. Ratib. OsiriX: An Open-Source Software for Navigating in Multidimensional DICOM Images. *Journal of Digital Imaging*, 17:205–216, 2004.

[9] J. Victor, D. Van Doninck, L. Labey, B. Innocenti, P.M. Parizel, and J. Bellemans. How precise can bony landmarks be determined on a CT scan of the knee. *The Knee*, 16:358–365, 2009.

[10] VTK, The Visualization Toolkit. `http://www.vtk.org/`.

[11] VTK/Examples/Python. `http://www.vtk.org/Wiki/VTK/Examples/Python`.