UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

A 3D Web Interface for Plastic Surgery Simulation

José Luís Pina Rolo

Orientação: Pedro Salgueiro e Salvador Abreu

Mestrado em Engenharia Informática

Dissertação

Évora, 2015

UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

A 3D Web Interface for Plastic Surgery Simulation

José Luís Pina Rolo

Orientação: Pedro Salgueiro e Salvador Abreu

Mestrado em Engenharia Informática

Dissertação

Évora, 2015

Abstract

In the Plastic Surgery domain, one of the most important things to know before the surgery takes place is the impact of the incisions made in the aesthetic aspect and in the patient comfort. Breast reduction surgery is not an exception, since it is a very sensitive procedure that may cause functional and aesthetic problems to the patient after the surgery.

The work described in this thesis presents a web interface for breast reduction surgeries simulation based on a research project that developed a mathematical approach to model the women breast [21, 19]. This research also developed a set of computer tools, which are the basis for the work presented in this thesis.

In this thesis we present an interactive tool to simulate breast reduction surgeries, providing a simple interface that allows an intuitive planning of real surgeries. It is designed for surgeons to simulate all aspects related to breast reduction before performing the real surgery.

Interface Web 3D para Simulação de Cirurgias Plásticas

Sumário

No domínio da cirurgia plástica, um dos aspetos mais importantes a saber antes da cirurgia, é o impacto das incisões feitas no aspecto e no conforto do paciente. A cirurgia de redução mamária não é uma exceção, dado que é um processo muito delicado que poderá causar problemas estéticos e funcionais ao paciente depois da cirurgia.

O trabalho descrito nesta tese apresenta uma interface web para simulação de cirurgias de redução mamária, baseado num projeto de investigação ainda em desenvolvimento, que desenvolveu uma abordagem matemática para modelar a mama da mulher [21, 19]. Esta investigação também desenvolveu um conjunto de ferramentas informáticas, que são a base do trabalho apresentado nesta tese.

Esta tese apresenta uma ferramenta interativa para a simulação de cirurgias de redução mamária, contendo uma interface simples que permite aos cirurgiões fazer um planeamento das cirurgias reais de uma forma intuitiva. Foi desenhada de modo a simular todos os aspectos relacionados com a cirurgia de redução mamária antes da cirurgia real.

To my Parents...

Acknowledgements

I would like to thank my parents for the support and stability needed during the development of my master thesis. Without their assistance this work would not have been possible.

I also thank to my supervisors, Prof. Pedro Salgueiro and Prof. Salvador Abreu, the new ideas and for always providing alternative solutions to address the problems occurred during this work, encouraging me to overcome the problems. Without their support, motivation and availability, it would be impossible to achieve the work described in this thesis.

I am also grateful to the members of the VAPS project, for providing the knowledge and the tools needed to understand the operation of the simulation software and the procedures of a real surgery. I specially thank to Dr. Igor Vasilevskiy the medical feedback during the development and his precious contribution for this thesis.

Finally I want to acknowledge Fundação para a Ciência e a Tecnologia (FCT) the research grant (EXPL/MAT-NAN/0606/2013) inside the VAPS project that conduce to the work described in this thesis.

Acronyms

API Application Programming Interface

CAD Computer-Aided Design

DOM Document Object Model

FVLib Finite Volumes Library

GIS Geographic Information System

Gmsh G mesh

HTML5 Hypertext Markup Language, version 5

OpenGLES2.0 Open Graphics Library for Embedded Systems, version 2.0

RGB Red, Green and Blue

SVG Scalable Vector Graphics

VAPS Variational Approach to Plastic Surgery

VRML Virtual Reality Modeling Language

WebGL Web Graphics Library

X3D Extensible 3D Graphics

X3DOM Extensible 3D Graphics based on a Document Object Model

XML Extensible Markup Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents an Introduction to the work presented in this thesis, introducing the breast reduction surgeries topic as well as the motivation for doing this work.

All surgical procedures have their own associated risks for the patient health, so all surgeries must be planned in detail, in order to minimize the risks. In the last years the software dedicated to health care increased, introducing high realism features that helps health professionals to prepare their procedures [19]. There are some kinds of software dedicated to the health professionals, such as augmented reality surgery software that helps the surgeon to interact with the human body through specific surgical tools [32, 34]. Among the health related software, there is also a category dedicated to the mathematical simulation of human organs like the heart [35, 31], liver [42] or skeleton [30].

With regard to surgical procedures, there are also some studies on numerical simulations that predict the final result of specific kinds of surgeries before the real surgery takes place, reducing the patient risks, such as the works presented in [19, 22, 23]. More specifically, the breast reduction simulation is an important tool to prevent common surgery mistakes, related with the breast aesthetic aspect after the surgery and the patient comfort.

The main advantage of using a surgery simulation to plan a real surgery is that surgeons can immediately view the consequences of a specific parameter and try another parametrization as many times as necessary, in order to achieve a good result. Without the simulation, the surgeon has only one chance to do the right parametrization. This type of simulation is particularly important on breast reduction surgeries, since the post-surgery breast shape is a very important aspect for the patient, and a simple error on the incisions can have a huge impact in the final result. With a simulation tool, doctor and

patient can plan the surgery together, and if all goes well there will be no problems to solve in the post-surgery [21, 19].

Breast modeling is an active field of research, with works on aesthetic surgery [19] and medical imaging analysis [17]. Although the active research in this field, the modeling of an adequate breast model has not been reached yet [21].

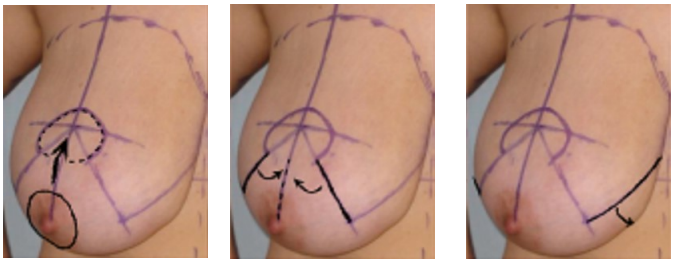
The work described in this thesis is based on the research project Variational Approach to Plastic Surgery (VAPS) (EXPL/MAT-NAN/0606/2013), a joint project of Universidade do Minho and Universidade de Évora. The work presented in this thesis was made in the context of a research grant funded by this project, and the development process was closely monitored by all project members.

The researchers of Universidade do Minho developed a software that simulates the breast reduction surgeries, based on mathematical models of the women breast. The work related with this simulation software are published in [21] and [19].

The main goal of the work described along this thesis was the development of a 3D web interface based on the existing numerical surgery simulation software for breast reduction surgeries, so that surgeons can use it to prepare their surgeries more accurately and with less risk for the patient. Also important, the interface should be intuitive so that surgeons can use it without any special computer skills. With this approach, surgeons can plan their surgeries virtually in any device with a modern browser, allowing the interaction with a 3D breast model in real-time.

1.1 Breast Reduction Surgery

One of the methods used by surgeons to perform breast reduction surgeries consists of several steps. In first place, the nipple is placed in a new position and the breast is incised by two plans orthogonal to the chest and by an oblique plan(Fig. 1.1a). Next, the tissues incised by the two orthogonal plans are sutured to each other(Fig. 1.1b). After that, the tissues incised by the oblique plan are sutured to the chest(Fig. 1.1c)[21]. The work described in this thesis is based on a numerical simulation software that simulates this surgery procedure.



(a) (b) (c)

Figure 1.1: Breast reduction surgery.

1.2 Breast Modeling Simulation

Currently, there are several mathematical approaches to model soft tissues of the breast [24, 25, 17, 22, 21], but the development of an adequate breast model is still an unsolved mathematical problem. The work described along this thesis is based on a numerical approach for modeling the breast[21]. Although the model is of low precision, it was verified by the authors that it can be used to identify the most common errors on this type of surgery, allowing surgeons to avoid them.[21]

This model considers the breast tissue as a hyperelastic material. Although most soft tissues are incompressible, it was assumed that the breast is a compressible Neo-Hookean material.[21]

The breast is formed by several types of tissues but its elastic properties cannot be deduced from the elastic properties of the tissues that forms it. Another important aspect is the skin that presents different properties from the other tissues of the breast, and therefore must be modeled differently. The breast tissue is modeled using three dimensional elements and the skin is modeled using only two dimensional elements. The breast skin has the function of containing the jelly-like breast tissue, so it is a very important aspect to understand and forecast the results of breast reduction surgery. The breast also has the so called Chassaignac's space, that is responsible for the connection between the breast and chest, playing a special role in the breast mobility. The Chassaignac's space is modeled as mass-spring system [21].

1.3 Motivation

There are many risks associated with breast reduction surgery, including the patient comfort and the aesthetic aspects of the breast after surgery. The simulation software for breast reduction surgery used in this work reduces the risks through a numerical simulation of the three main steps of the real surgery, namely: the pre-surgery process when the surgeon defines the location of the incisions; the surgery process including the incision and suturing steps; and the post-surgery that simulates the breast after the surgery under a gravity field to be as faithful as possible to reality. This enables surgeons to evaluate the breast shape before the real surgery takes place, preventing complex problems that might occur and avoid corrective surgeries to fix the breast shape. However, to use these numerical simulation software, surgeons must have programming skills, which is hard to find in health care professionals.

At the time, there is no tool that solves this problem in a simple and interactive way that surgeons can use in a daily basis. To address this problem, we decided to build an interface with the best of the both worlds. This interface must be easy to use by surgeons, sufficiently clean and organized to retain the user focus and the most important aspect,

it must have a faithful viewing mechanism that allows surgeons to have a good feedback about the consequences of a set of parameters.

Throughout this thesis, the bridge between plastic surgeons and complex mathematical simulations on plastic surgery will be established with an interface that presents the evolution of a 3D breast model, during the entire surgery simulation process.

Chapter 2

State of the Art

This chapter introduces the State of the Art related to the work described in this thesis. First we present a survey on surgery simulation systems and then we present the actual paradigms related to web application and it's interactive content in the browser.

2.1 Surgery Simulation

In the domain of surgery simulations, there are some systems that allow the simulation of a variety of medical interventions, such as laparoscopic procedures, cardiovascular surgeries, plastic surgeries and others. However, many of them are training platforms that surgeons use to practice the complex procedures of surgeries, like *Epona Medical* [4] and *SimSurgery* tools [11].

Surgery simulation is an active field of research, studying different approaches to the simulation of human organs like the heart [35, 31, 33], the liver [42, 26] or the skeleton [30, 20]. Some of the mentioned works are based on a physics system that simulates the human body, while others are based only on computer software based on mathematical approaches.

In the case of plastic surgeries, training platforms are an important tool for surgeons, but that is not enough, due the fact that they don't take into account the aesthetic aspects and the patient comfort after the surgery.

There are tools that simulate the physical aspect after the surgery, namely *Vectra XT 3D* [13], that is an imaging solution with a surgery simulator software and *Crisalix* [3], a

tool to be used mainly by patients in order to understand the impact of a given surgery. Usually these tools are only qualitative, i.e.: they don't add relevant information for the surgery. The *Vectra XT 3D* is the only tool that supports breast lift and there is no tool that supports breast reduction, possibly due the fact of breast augmentation and lift surgeries having more commercial potential than breast reduction surgeries, which are mainly used to solve patient problems, despite of the improvement of the women aesthetic aspect. The main difference between breast lift and breast reduction surgery is the amount of tissue removed, while in breast lift only the skin is removed, in breast reduction both skin and tissue are removed[2].

2.2 Native Application vs Web Application

A local application is very different from a web application due to the environment in which they operate, since a local application is made for a specific device or group of devices and has to be installed on the device while a web application executes directly inside a web browser without the need of additional software.

Nowadays we can find many examples of web applications over the Internet, such as *Google Docs* [5], which is a productivity suite based on browser or *Photopea* [8] which is a web based photo editor. The main differences that distinguish a normal website from a web application are not consensual, however a web application is always compared to a normal software that as the particularity of running in a browser.

Regardless to the Internet connection, although a web application can be used offline, this kind of usage is very limited because there is no connection between client and server, so, in the most cases an Internet connection is a required feature. A local application can also use an Internet connection, but it's not a requirement to perform their main tasks. Other important aspect is the updating process, while a local application must download their updates, which involves some waiting time by the user, in a web application the user just has to refresh the browser page and the application is updated.

One of the problems of web applications is the performance, since most of them don't execute any tasks in the client device and there is always a delay due the communication between client and server. However, with HTML5, web developers can take advantage of client hardware without any kind of browser plugins, which it was not possible before HTML5. This is a great feature, as it solves most performance problems related to the web applications. In fact, HTML5 is the major driver of web applications, mostly due the limitations of the previous HTML standard. 3D applications can also benefit from the use of HTML5, which introduces Web Graphics Library (WebGL), a 3D rendering API for the web, based on OpenGL®ES 2.0. [7] This feature opens the doors of the web to the high quality 3D rendering, like games and immersive 3D simulations.

2.3 Interactive Web Applications

Due the changes introduced by HTML5, the Web is becoming more capable to do things which until now were exclusive to native applications. These features allow web developers to build applications that are closer to the user, making them more interactive and intuitive to use.

2.3.1 Web based Interactive 3D Graphics

The presentation of 3D graphics in web browsers is not new, being introduced in 1994 by David Raggett as a new standard of platform-independent exchange of 3D worlds over the web, called Virtual Reality Modeling Language (VRML) [37] Although the innovation brought by this standard, it only supported static scenes. It was only in 1997 with the review of the initial standard that it was added the possibility of dynamic scenes, allowing the user to interact with the 3D scene. However the visualization of 3D content was only accessible through supported browsers and relying on a specific plugin that users should install.

After a few years, in 2001, the Extensible 3D Graphics (X3D) was introduced as an XML encoding of VRML. The X3D proposal added to VRML new features like shaders and geo-location along with custom support for different areas such as Medicine, Computer-Aided Design (CAD) and Geographic Information Systems (GISs). Currently, X3D is a well-known format in the 3D community, supported by professional software such as `Blender` [1], becoming a famous 3D format outside the web [15].

More recently, with the appearance of HTML5, there are new ways to present 3D content on the web. The low-level option is WebGL which is integrated in the 3D `canvas` context of HTML5 and derived from Open Graphics Library for Embedded Systems, version 2.0 (OpenGLES2.0) [7].

Alongside WebGL, there are some Javascript libraries that add some syntactic sugar to the WebGL API, like Scene.js [10], Three.js [12] or X3DOM[1]. Scene.js and Three.js are similar as both define the 3D scene through Javascript in a imperative way. However, X3DOM is different, since it tries to integrate the existing X3D format and HTML5. The main goal of X3DOM is to become a 3D standard for HTML5 like Scalable Vector Graphics (SVG) for 2D content. While the X3DOM isn't a standard it can be used through a Javascript library that implements some X3D elements on HTML5 Document Object Model (DOM) tree [16]. Figure 2.1 shows the relationship between various technologies relative to HTML5.

[1] pronounced X-freedom

Figure 2.1: Graphic technologies in HTML5. Retrieved from [16].

2.3.2 Dynamic Geometric Diagrams

A geometric diagram is a specific type of 2D figure, with the particularity that it only uses geometric primitives, e.g. lines, line segments, circles, among others. There are some ways to make this shapes interactive. One option is to use the canvas element present in HTML5 to draw the figures and make them interactive through Javascript. However to build an interactivity based on 2D primitives from scratch, with Javascript is a complex task.

Another option is to use an existent Javascript library like *JSXGraph*, *Three.js* or *Raphael.js* [6, 12, 9]. These libraries implement an abstract layer on top of Javascript that helps the programmer to build complex 2D drawings. This layer may contain features like path animations, intersections between shapes and other functions that are not native to the Javascript canvas element. However, to build an interactive geometric diagram, *JSXGraph* is the best option.

This library was designed specifically for dynamic geometric constructions on the web, where the user can manipulate almost anything and see the consequences in real-time. This approach is specially useful for demonstrations of geometric theorems, like the Euler line theorem (See Figure 2.2). In the example presented in Figure 2.2, the user is free to manipulate the points `A`, `B` and `C`, changing the geometry of the triangle, while the points `H`, `S` and `U` remain collinear, demonstrating the Euler line theorem. This capability of *JSXGraph* is very good to teach and explore complex geometric structures in an interactive way.

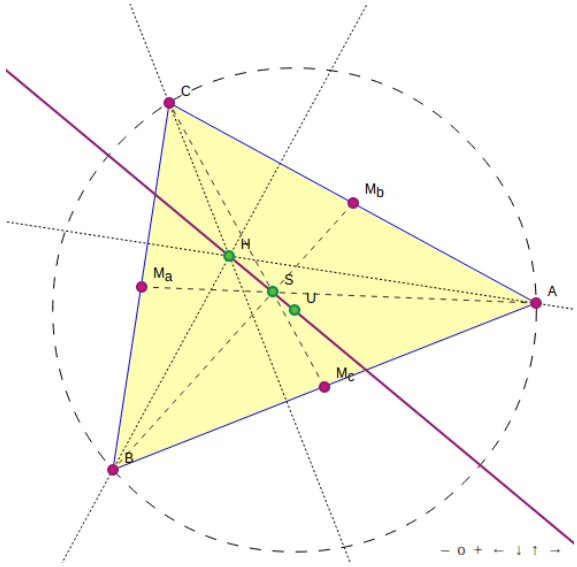


Figure 2.2: Euler line theorem represented with *JSXGraph*.

2.4 Conclusions

In this chapter we introduced the necessary background to understand the technologies used in this work, presenting the state of the art regarding the technologies used to implement the web application described in this thesis.

We started by presenting some health solutions designed to improve surgical procedures using simulators, followed by a comparison between native and web application, focusing on the strengths and weakness of both. We also introduced some aspects related to the interactive use of the web and associated technologies, namely 3D and 2D interaction. The graphics technologies described on section 2.3 are an important aspect to this thesis, since the main goal is to develop a web application containing 2D and 3D models.

Chapter 3

Breast Reduction Surgery Simulation

This chapter describes the simulation software used in this work. The simulation tools described in this chapter were designed to simulate breast reduction surgeries which is a specific type of plastic surgery. Throughout this chapter, all the components of the simulation software, as the connections between the simulation stages, are described in detail. This simulation software is also the basis of the web application described in Chapter 4.

3.1 Introduction

As mentioned in Chapter 1, the work described in this thesis is based on software that simulates breast reduction surgeries. This software was developed by some members of the VAPS project, namely the authors of the work presented in [21] and [19], which was achieved by the use of mathematical techniques related with breast modeling and simulation.

The software is made of two tools: the surgery simulator and the displacement simulator. The first one is used to simulate the surgery itself, while the second one is used to apply a gravity field to the post-surgery model.

This chapter explains how these tools work together and how they can be used to simulate breast reduction surgeries. We will also explain the relations with some auxiliary tools, such as *Finite Volumes Library (FVLib)* and the *G mesh (Gmsh)*.

3.2 Numerical Simulation Software

The simulation of breast reduction surgery is divided in three main steps: 1) the first step is dedicated to the definition of the breast geometry and the breast incision angles; 2) the second step is the surgery simulation itself; and 3) the last step is the application of the gravity field to the previous result.

All of these steps use software simulation tools that have their own operation and are still in the development stage. These tools are the *Surgery Simulator* and the *Displacement Simulator*. The *Surgery Simulator* receives a previously defined breast mesh that represents the patient breast and the incision parameters that will be used to make the breast reduction surgery. This initial mesh is obtained with an existent template file filled with the new parameters (Geometry file Fig. 3.1). After that the software performs the surgery simulation, suturing the incisions made on the breast mesh. The *Displacement Simulator* that receives the mesh produced by the *Surgery Simulator* and places it in a customizable gravitational field to understand the behavior of breast under gravity.

The first step is used only for the definition of the breast geometry and planning the incisions, so there is no need of a simulation tool in this step. The workflow of the simulation process is shown in Fig. 3.1.

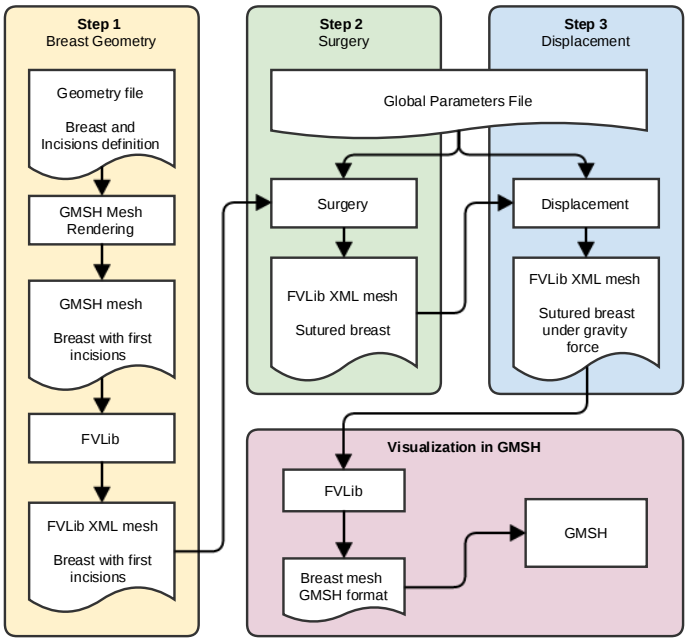


Figure 3.1: Overall structure of Simulation Software.

The simulation tool use a library called FVLib that provides a layer for complex mathematical calculations and a XML specification for 3D models. This custom 3D representation (FVLib XML mesh) is used between all the simulation programs, and the 3D breast model must be in this format. This library also provides some tools that converts `.msh` (Gmsh

mesh format) into FVLib XML format and vice-versa. These auxiliary tools are described in Section 3.6.

In order to make all the set of parameters accessible for all the simulation tools, there is a global parameters file made in XML (See Figure 3.1), where all the required parameters are defined. The structure of the global parameters file is defined in Section 3.2.1.

3.2.1 Global Parameters File

The global parameters file is shared between the simulation tools and is used to store the parameters of all steps, including some parameters related with the simulation algorithm, such as the maximum number of iterations. The structure of a normal parameters file is presented in Listing 3.1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<FVLIB>
   <PARAMETER>
      <!-- Filenames definition -->
      <parameter CirurgiaMeshFile="breast.xml"
      CirurgiaPositionFile="position.xml"
      CirurgiaDisplacementFile="displacement.xml" />
      <parameter SuturedMeshFile="breast_sutured.xml"
      SuturedPositionFile="position.xml"
      SuturedDisplacementFile="displacement.xml" />

      <!-- Surgery and Displacement simulation related parametes -->
      <parameter Lambda="1000" Mu="150" Density="1000"
      CoefChassignac="60000" />
      <parameter LambdaP="8000" MuP="1600" SkinThickness="0.002" />
      <parameter XGravity="0.0" YGravity="0.0" ZGravity="-9.8" />
      <parameter EpsilonDerivative="1.e-8" DerivativePrecision="1.e-10"
      MaxIter="1000" />
      <parameter NormEpsilon="1.e-6" LogParameter="1.e-8" />

      <!-- Parameters related with the breast modeling -->
      <parameter TransformLeft="15" TransformRight="15" TransformBotL="0"
      TransformBotR="0" Plano_Rot="0" />
      <parameter H="0.14" R="0.10" S="0.06" />

      <!-- Simulation specific parameters -->
      <parameter ZMinFixation="-1.0" ZMaxFixation="-0.03"
      RotationFixation="0.0" />
      <parameter CodeFixation="1" CodeChassignac="2" CodeSkin="3" />
      <parameter CodeLeftSuturation="4" CodeRightSuturation="5"
      CodeBottomSuturation="6" />
      <parameter CodeDoubleSuturationLeftSide="7"
      CodeDoubleSuturationRightSide="8" CodeTableMatchLeft="9"
```

```
34      CodeTableMatchRight="10" CodeJoin="11" />
35      <parameter CodeDispLeft="104" CodeDispRight="105" />
36    </PARAMETER>
37  </FVLIB>
```

Listing 3.1: Global Parameters File

The parameters file starts with a typical XML header where the version and encoding are defined (Line 1). After that, there is a XML block delimited by the tags `<FVLIB>` and `<PARAMETER>` (Lines 2, 3, 36 and 37), which is a container for all the defined parameters.

The first set of parameters are related with the filenames that will be used during all the simulation process (Lines 5 to 10):

- The `CirurgiaMeshFile` is the input file that contains the breast model with the breast geometry and incisions definition. The `CirurgiaPositionFile` and the `CirurgiaDisplacementFile` are auxiliary files that must be generated during the simulation and do not contain anything that can be visualized.

- The `SuturedMeshFile` is the output model of the surgery simulation, that after the displacement process takes place, will generate the `SuturedPositionFile`, which is an auxiliary file and the `SuturedDisplacementFile`, which is the displacement that must be applied to surgery result (`SuturedMeshFile`) in order to simulate the given gravity on the breast model.

The next set of parameters are related with the simulation itself (Lines 13 to 19). The `Lambda`, `Mu`, `Density`, `CoefChassignac`, `LambdaP`, `MuP` and `SkinThickness` are related with the *Surgery* process, while `XGravity`, `YGravity` and `ZGravity` are the parameters for the *Displacement* process. The `EpsilonDerivative`, `DerivativePrecision`, `MaxIter`, `NormEpsilon` and `LogParameter` are used to finely tune the simulation algorithm.

The third set of parameters are related with breast modeling and the incisions definition (Lines 21 to 24). The `TransformLeft`, `TransformRight`, `TransformBotL`, `TransformBotR` and `Plano_Rot` are related to the incisions angles and to the suture plane, while `H`, `R` and `S` are related with the breast geometry. The last parameters must be included in the global parameters file because they are needed across all the simulation steps.

The last set of parameters are related with the creation of the 3D model (Lines 26 to 35), namely the codes assigned to the faces in order to distinguish the different parts of the breast.

3.3 Breast Parametrization

In the first step of the surgery simulation process, the surgeon will input all pre-surgery aspects related to the patient physiognomy. This step is divided in two parts: 1) the definition of the breast geometry and 2) the incisions parameters.

The definition of the breast geometry is made by a sectioned sphere and the surgeon only needs to input two parameters: the depth of the breast (H), which represents the place where the sphere is sectioned by a vertical plane, and the radius of the circle (R) made by the intersection between the vertical plane and the sphere that represents the breast radius close to the chest.

Figure 3.2 represents the initial breast geometry with all the measurements needed by surgeon to define the geometry of the breast and the incisions that will be made.

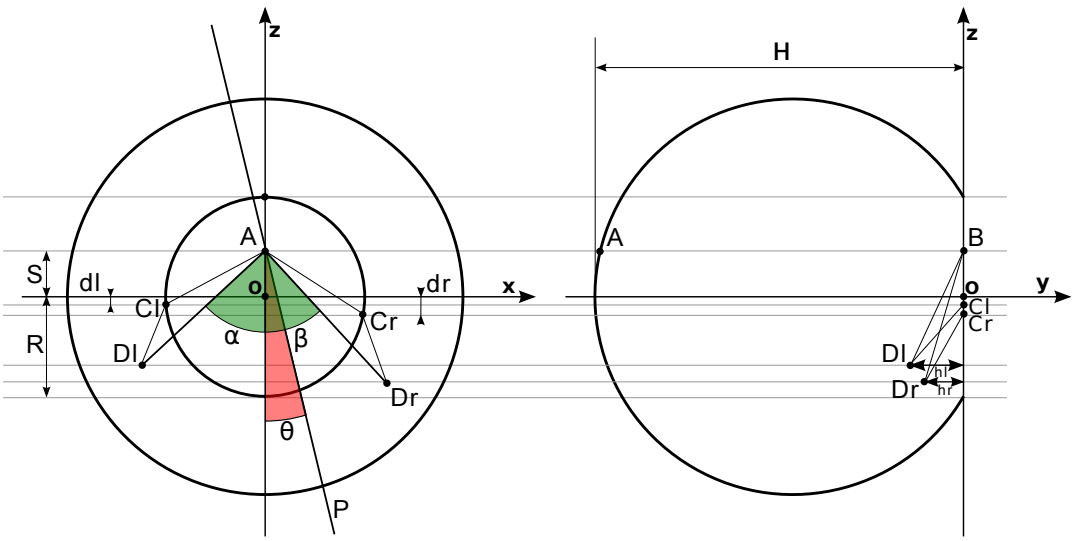


Figure 3.2: Breast Geometry

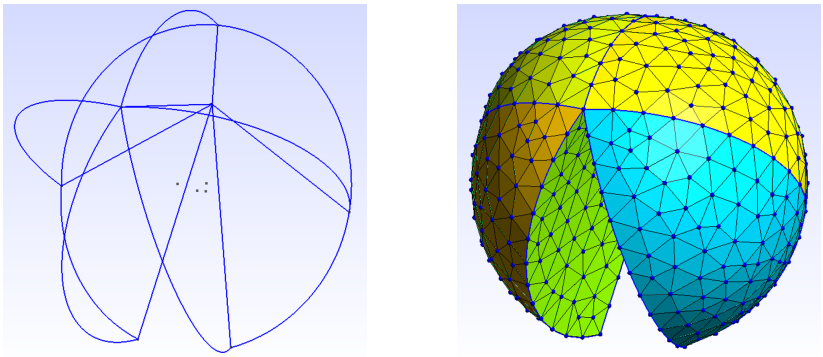
To model the surgery, the surgeon must define four incisions: two in the front of the breast (left and right side) and two in the rear of the breast, close to the chest. The frontal incisions are made with two planes orthogonal to the chest through point A, which is the new position of the nipple. The left and right incisions form an angle of α and β degrees, respectively, with the z negative axis. The new position of the nipple can be modified by changing the value of S. The rear incisions are made with two obliques planes to the chest. In the right side of the breast the plane goes through B, Cr and Dr with a distance hr between Dr and z axis. The left plane passes through B, Cl and Dl with a distance hl between Dl and the z axis. In order to finely adjust this planes, the surgeon can change the values of dl and dr, which will affect Cl and Cr. He can also modify s, which will change the position of points A and B. The surgeon is also free to manipulate the plane P, which defines the angle of the suture. If the angle is zero, the suture will be

symmetric, but if is greater or lesser than zero the suture will be made in the right or left side, respectively.

This is the most important step in the entire surgery simulation since this is the step where all the patient customization is made. An error in this initial parametrization will lead to an unrealistic surgery simulation that will not correspond to the results of a real surgery. The simulation software cannot check if the entered parameters make sense from a surgical point of view.

3.3.1 Components

The initial parametrization with the parameters presented in the last section is made with an external tool called Gmsh that is a "3D finite element mesh generator with built-in pre- and post-processing facilities" [27]. Gmsh has it's own scripting language to define 3D meshes, which is used to parametrize the initial breast model. The result of the parametrization script is shown in Fig. 3.3a (only with geometry rendering) and in Fig. 3.3b (with 3D mesh rendering). Figure 3.3a is the viewable result before the step "Gmsh mesh rendering" step presented in Fig. 3.1, while Fig. 3.3b is the result after this first step.
The 3D mesh presented in Fig. 3.3b considers all parameterization done by the surgeons. If the surgeon is satisfied, he can continue to the next phase, the surgery simulation. If there is something to adjust, the Gmsh script has to be modified in order to reflect these new adjustments and the 3D mesh needs to be rendered again.



(a) `Gmsh` geometry rendering (b) `Gmsh` 3D mesh rendering

Figure 3.3: `Gmsh` rendering

3.4 Surgery Simulation

After the definition of the parameters related to the breast geometry and incisions, we are ready to advance to the next step, the surgery simulation. In this step, the incisions defined in the previous step will be sutured according to the parameters introduced by the surgeon.

The breast surgery is a very complex process, so even small adjustments in the parameters can have a large impact in the final shape of the breast, which is very important for the patient comfort after the surgery. The surgery parameters are related with the dynamics of the breast, such as the *Breast Density*, *Skin Thickness*, *Elasticity* and *Compressibility* of the skin and the interior tissue, and the *Chassignac Coefficient* which affects the interaction of the breast with chest [21].

To perform the surgery simulation, the mesh produced in the parametrization step must be converted to the FVLib XML format, making it compatible with the surgery simulation tool. This conversion is made with *fvcm*, a tool included in the FVLib library. After that the global parameters file (See Section 3.2.1) must be created, including all the parameters related to the surgery simulation process, namely the parameters defined in the first step that are related to the breast geometry and to the incisions, the parameters introduced in this step related to the breast dynamics, and the parameters that will be defined in the next step, which are not available in this step. This parameters file also contain the location of all files needed for the simulation, including the input and output filename of the surgery simulation and the displacement simulation together with some other auxiliary files that are needed. The parameters file is not created before because the simulator is only executed from the surgery simulation. After the parameters file is correctly configured and the input mesh calculated, the surgery simulator is ready to be executed.
Fig. 3.4 presents the result of a surgery simulation in a gravity free environment. The result of the simulation is in FVLib XML format, which has to be converted into Gmsh, so that the surgeon can see the result (See Fig. 3.1).

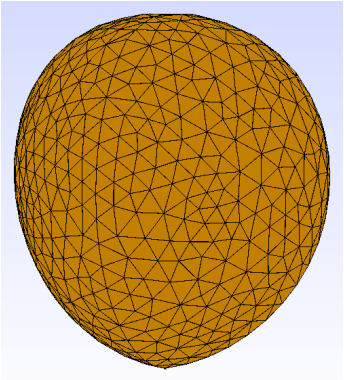


Figure 3.4: `Gmsh` rendering of the breast after the surgery

3.5 Displacement Process

After the surgery simulation the surgeon has a global idea about the shape of the breast and the implications of a specific parametrization. However the breast shape obtained after the surgery simulation is not consistent with the reality due the fact that the model is in a zero gravity field. The zero gravity field is very useful when the surgeon is modeling

and parameterizing the breast, but for a realistic final evaluation, the breast model must be inserted in a gravity field.

The *Displacement Process* is the final stage of the simulation. In this step, the surgeon can analyze the behavior of the breast model under a defined gravity field, which means that the surgeon is free to simulate the breast behavior in all positions, even experience with gravity forces different from the ones found in Earth.

In terms of the organization of the software, this step receives the breast model created in the second stage (*Surgery Simulation*) alongside with a gravity vector which indicates the gravity force for each axis. The gravity vector is defined in the global parameters file like in the previous step. After that, the *Displacement Process* can be executed. This process generates a set of displacement vectors that represents the displacement of the breast under the specified gravity. Then, the displacement vectors are applied to the breast model calculated in the second step using *fvcd*, which is a FVLib tool to add a certain displacement to a gravity free model. The *fvcd* tool returns a new 3D breast model under the specified gravity in the FVLib XML format. This file needs to be converted into the Gmsh format so that the surgeon can visualize the result (See Fig. 3.1). An example result of the *Displacement Process* is shown in Fig. 3.5.

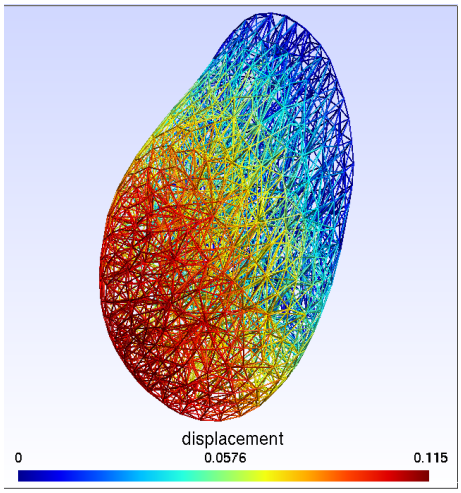


Figure 3.5: `Gmsh` rendering of the breast after the *Displacement Process*

3.6 Auxiliary Tools

As mentioned before, there are some auxiliary tools that are used in the simulation process. These tools are: 1) the FVLib library, which is used by the simulation programs to represent 3D meshes and perform some calculations, and 2) Gmsh, which is a mesh modeling tool used to build the initial breast model at the first step of the simulation. Each one of these tools has it's own format for representing 3D meshes, but during the simulation work-flow there is the need to make a conversion between them. This conver-

sion is made with tools provided by the FVLib library. Sections 3.6.1 and 3.6.2 explains the 3D representation used by Gmsh and FVLib, respectively.

3.6.1 Gmsh File Format

Gmsh has two different types of files: the geometry script file which is based on a scripting language, used to define geometry shapes and perform calculations with these shapes, and the Gmsh mesh format, which is the output format after rendering a geometry script.

Geometry File Format

The geometry file is the source-code of a 3D model before the 3D rendering. The whole structure of the 3D model must be defined in the geometry file, i.e important points, vertices, edges and polygons.

The Gmsh script language has a simple syntax, based on an incremental construction of the model. As it can be seen in Listing 3.2, which represents a cube, the script begins with the definition of all points (Lines 2 to 9) followed by the definition of the edges (Lines 12 to 2) and then the definition of the faces (Lines 26 to 31) and surfaces (Lines 34 to 39). Figure 3.6 shows a cube generated by the source-code presented on Listing 3.2 after the Gmsh rendering. Building a breast geometry is much more complex than building a simple cube, as it can be seen in the geometry template present on Annex A.

```
1  // Vertices Definition
2  Point(1) = {0, 0, 0};
3  Point(2) = {.1, 0, 0};
4  Point(3) = {0, .1, 0};
5  Point(4) = {.1, .1, 0};
6  Point(5) = {0, 0, .1};
7  Point(6) = {.1, 0, .1};
8  Point(7) = {0, .1, .1};
9  Point(8) = {.1, .1, .1};
10
11 // Edges Definition
12 Line(1) = {1,2};
13 Line(2) = {2,4};
14 Line(3) = {4,3};
15 Line(4) = {3,1};
16 Line(5) = {5,6};
17 Line(6) = {6,8};
18 Line(7) = {8,7};
19 Line(8) = {7,5};
20 Line(9) = {5,1};
21 Line(10) = {6,2};
22 Line(11) = {8,4};
```

```
23 Line(12) = {7,3};
24
25 // Faces Definition
26 Line Loop(13) = {1,2,3,4};
27 Line Loop(14) = {5,6,7,8};
28 Line Loop(15) = {-1,-9,5,10};
29 Line Loop(16) = {-2,-10,6,11};
30 Line Loop(17) = {-3,-11,7,12};
31 Line Loop(18) = {-4,-12,8,9};
32
33 // Surfaces Definition
34 Plane Surface(19) = {13};
35 Plane Surface(20) = {14};
36 Plane Surface(21) = {15};
37 Plane Surface(22) = {16};
38 Plane Surface(23) = {17};
39 Plane Surface(24) = {18};
```

Listing 3.2: Gmsh geometry file representing a Cube.

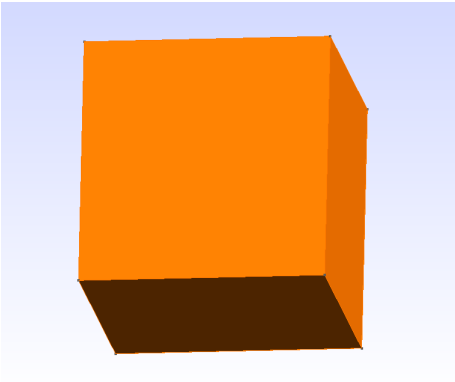


Figure 3.6: Simple Cube made with Gmsh

The first code block (Lines 2 to 9) of the Gmsh geometry file (Listing 3.2) is the definition of the cube vertices i.e. the declaration of the coordinates that form the cube. Each point is made of two components: a number that identifies the point(L) and a set of coordinates (X, Y and Z), according to Listing 3.3.

```
1 Point(L) = {X, Y, Z};
```

Listing 3.3: Point syntax.

The second code block (Lines 12 to 2) is the definition of the edges that make the cube. Each edge ($Line$) is made of three components: a number that identifies the edge (L), a number that references the first vertice of the edge (V_1) and a number that references the last vertice of the edge (V_2), as shown in Listing 3.4.

```
1 Line(L) = {V1, V2};
```

Listing 3.4: Line syntax.

The third code block (Lines 26 to 31) is the definition of the cube faces. Each cube face (Line Loop) is made of two components: a number that identifies the face (L) and a list of edges that compose the face (E_1, E_2, E_3 and E_4), which in the case of the cube are four edges. The minus signal behind the edge number means that edge are interpreted reversely i.e: if a edge was previous defined from point A to point B, with the minus signal it is interpreted from point B to point A. Listing 3.5 presents the syntax of the *Line Loop* element.

```
1 Line Loop(L) = {E1, E2, E3, E4};
```

Listing 3.5: Line Loop syntax.

The last code block (Lines 34 to 39) is the definition of a surface. The main differences from the *Line Loop* element are: the *Plane Surface* element may reference more than one *Line Loop*, while the *Line Loop* can only build a face (with more than three edges). Regarding the 3D rendering, only the *Plane Surface* will generate the opacity of the cube faces, otherwise it will be generated a wireframe cube. The main advantage of the *Plane Surface* is the possibility to create a new surface composed by n previously defined holes (*Line Loop*). In the case of the cube, each node of a *Plane Surface* is made of two components: a number that identifies the surface (L) and a number that references the face of the cube (*Line Loop*) (F_1), as shown in Listing 3.6.

```
1 Plane Surface(L) = {F1};
```

Listing 3.6: Plane Surface syntax.

The syntax described in the last paragraphs is based on the *Gmsh Reference Manual* [28].

Mesh File Format

The *Gmsh* mesh format is very extensive [28], in this section we only describe the elements used by the simulation process and supported by FVLib converters. The example source-code presented in Listing 3.7 represents a simple cube.

The mesh format is composed by three main blocks (Listing 3.7): 1) the indication of the mesh format version, which is delimited with `$MeshFormat` (Line 1) and `$EndMeshFormat` tags (Line 3); 2) the definition of all the points (nodes) that form the mesh, which are defined between the `$Nodes` (Line 4) and `$EndNodes` tags (Line 14); and 3) the definition of the geometric primitives built with the nodes defined before delimited with the `$Elements` (Line 15) and `$EndElements` tags (Line 23).

Each line of the nodes definition block (Lines 4 to 14) is made by the node number, which is the label of the node, and the three coordinates of the point. The first line of the block is an exception, reserved to define the number of nodes that form the mesh.

In the elements definition block (Lines 15 to 23), each line is composed by: the element number, which is the label of the element; the element type which is the type of the geometric primitive that we want (1 for lines, 2 for triangles, 3 for quadrangles, 4 for tetrahedrons and so on); the numbers of the element tags that comes next; the element tags (not used by the simulation process) and finally the node list that compose the geometric element, defined with the node numbers declared in the previous block (Lines 4 to 14). As in the previous block, the first line indicates the number of elements defined in the current block.

```
1  $MeshFormat
2  2.2 0 8
3  $EndMeshFormat
4  $Nodes
5  8
6  1  0.0000  0.0000  0.0000
7  2  0.0000  1.0000  0.0000
8  3  0.0000  1.0000  1.0000
9  4  0.0000  0.0000  1.0000
10 5  1.0000  0.0000  0.0000
11 6  1.0000  1.0000  0.0000
12 7  1.0000  1.0000  1.0000
13 8  1.0000  0.0000  1.0000
14 $EndNodes
15 $Elements
16 6
17 1 3 0 1 2 3 4
18 2 3 0 5 6 7 8
19 3 3 0 6 2 3 7
20 4 3 0 5 1 4 8
21 5 3 0 8 7 3 4
22 6 3 0 6 2 1 5
23 $EndElements
```

Listing 3.7: Gmsh file format

3.6.2 FVLib XML File Format

The FVLib file format uses XML to define the 3D model and is divided in four parts: 1) the list of all points and respective coordinates of the 3D model; 2) the list of the edges of the 3D model, i.e. the definition of connections between two or more collinear points; 3) the list of the faces made by three or more edges; 4) the list of the cells composed by

one or more faces. These four different kinds of definitions are modeled as four different XML blocks, as seen in Listing 3.8.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <FVLIB>
3     <MESH dim="3" name="3Dmesh">
4        <VERTEX nbvertex="4">
5        <!-- label code coordinates -->
6           1  1  0.0000  0.0000  0.0000
7           2  1  0.0000  1.0000  0.0000
8           3  1  0.0000  1.0000  1.0000
9           4  1  0.0000  0.0000  1.0000
10       </VERTEX>
11       <EDGE nbedge="4">
12       <!-- label code numberOfVertices listOfVertices -->
13          1  0  2  1  2
14          2  0  2  2  3
15          3  0  2  3  4
16          4  0  2  4  1
17       </EDGE>
18       <FACE nbface="1">
19       <!-- label code numberOfEdge  listOfEdges -->
20          1  0  4  1  2  3  4
21       </FACE>
22       <CELL nbcell="1">
23       <!-- label code numberOfFaces listOfFaces -->
24          1  0  1  1
25       </CELL>
26    </MESH>
27 </FVLIB>
```

Listing 3.8: FVLIB XML file format

The FVLib file format begins with the typical tag found on XML documents, where the XML version and the encoding are defined. After that, there is a block called *FVLib* (Lines 2 to 27) which is a container for all the model data. Inside de *FVLib* block there is another block called *MESH* (Lines 3 to 26) which indicates that we are defining a Mesh in three dimensions (*dim* field), called "3Dmesh" (*name* field).

The next block is the *VERTEX* definition (Lines 4 to 10) which is a container for all the vertices that form the mesh. The field *nbvertex* indicates the number of vertices defined inside the block.
Each line of the *VERTEX* block (Lines 6 to 9) is made by three components: a label that identifies the vertex, a code that can be used to distinguish different kinds of vertices and a list of coordinates.

Then is defined the *EDGE* block (Lines 11 to 17) which is a container for the edges that form the mesh. The field *nbedge* indicates de number of edges defined.

Each line of the *EDGE* block (Lines 13 to 16) is composed by four parts: a label that identifies the edge, a code that can be used to distinguish different kinds of edges, the number of vertices that made the edge and a list of vertices labels referencing the vertices defined in the last block.

Follows the *FACE* block (Lines 18 to 21) which contains all the faces that compose the mesh. The field *nbface* indicates the number of faces defined.
Each line of the *FACE* block (Line 20) is made by four components: a label that identifies the face, a code that can be used to distinguish different kinds of faces, the number of edges that form the face and a list of edges labels referencing the edges defined in the *EDGE* block.

The last block is the *CELL* block (Lines 22 to 25), which contains the faces that form a cell. The field *nbcell* indicates the number of cells defined.
Each line of the *CELL* block (Line 24) is made by four parts: a label that identifies the face, a code that can be used to distinguish different kinds of faces, the number of faces that forms the cell and a list of faces labels referencing the faces defined in the last block.

3.7 Conclusions

In this chapter was described the simulation software that is the basis of the web application made in this work. The comprehension of the simulation software workflow and its auxiliary tools is fundamental to create the abstract layer described in Chapter 4 necessary to the proper functioning of the web application.

Throughout this chapter was described in detail all the simulation steps needed to complete a surgical simulation, from the geometry file until the displacement calculus, passing through the 3D representation using *Gmsh* and *FVLib* formats.

This chapter can be considered a low-level version of the work described in Chapter 4.

Chapter 4

Web Application for Breast Reduction Simulation

This chapter presents the Web Application for Breast Reduction Simulation implemented in the context of this work, including it's architecture and the necessary improvements to the Simulation Software in order to make it compatible with a web based application. This chapter also details each step of the simulation process using the web application.

4.1 Introduction

Breast reduction surgery is a complex process, where small changes can have a huge impact in the final breast shape, and when not done in the right way, it can lead to catastrophic results which are hard to solve. In order to achieve a realistic surgery simulation, surgeons must be able to manipulate and interact with the breast model in all steps of the simulation process: 1) Breast parametrization; 2) Surgery simulation and 3) Gravity simulation.

To allow an easy interaction with the simulation process, including an adequate breast modeling, it was decided to create a web application that reflects the simulation process in an easy way. This web application should use recent 3D capabilities found on modern web browsers to give the surgeon the freedom to inspect and interact with the 3D breast model before and after the simulation of the surgery. It was also decided that the interface should be as intuitive as possible and avoid the use of any type of client-side plugins.
This aspects are very important because they eliminate the majority of barriers between

surgeons and this simulation tool, since the web application is virtually accessible from any device with a modern browser and an internet connection.

To implement the web application it was decided to use X3DOM [16], which is an open-source JavaScript framework and runtime for 3D graphics on the web, based on WebGL [16]. Using X3DOM it is possible to create 3D scenes inside an HTML5 web page, allowing to visualize and interact with these 3D graphics on HTML5-enabled browsers without the need of any client-side plugins.
Although X3DOM is not part of the HTML5 specification, it's the best candidate to fulfill the current HTML5 specification for declarative 3D content [16].

4.1.1 Web Application Architecture

The web application is made of three layers:

- **User Interface**
  Responsible to present the content to the final user and catch the user actions.
- **Process the UI content**
  Process the user requests, delegates the hard work to the execution layer and wait for the results. This layer is implemented in PHP.
- **Execution**
  Responsible for the execution of user requests and sending the results back. This layer is implemented in Python.

The layers are interconnected as described in the diagram presented in Figure 4.1.

The normal work-flow of the web application starts with some parametrization made by the user on the interface. Then these parameters are sent to the Processing layer, which in turn will trigger the Execution with the user parameters. The Execution layer executes the simulation for the actual simulation step and saves the results inside a storage area. Note that each simulation step (Breast Parametrization, Surgery and Gravity Application) has it's own implementation of each described layer, each one specialized on a specific simulation step.

After the Execution layer has finished, the results are sent to the Processing layer, which in turn will send them to the User Interface.

The Execution layer only receives an input file on the second and third step because these steps need the previous result to continue, i.e. the surgery simulation step needs the parametrization made in the first step, and the Gravity Application step needs the Surgery simulation result to apply the gravity. The first step doesn't have dependencies, so there is no need to input any file into the Execution layer.

The Execution layer is explained with more detail in Section 4.2.

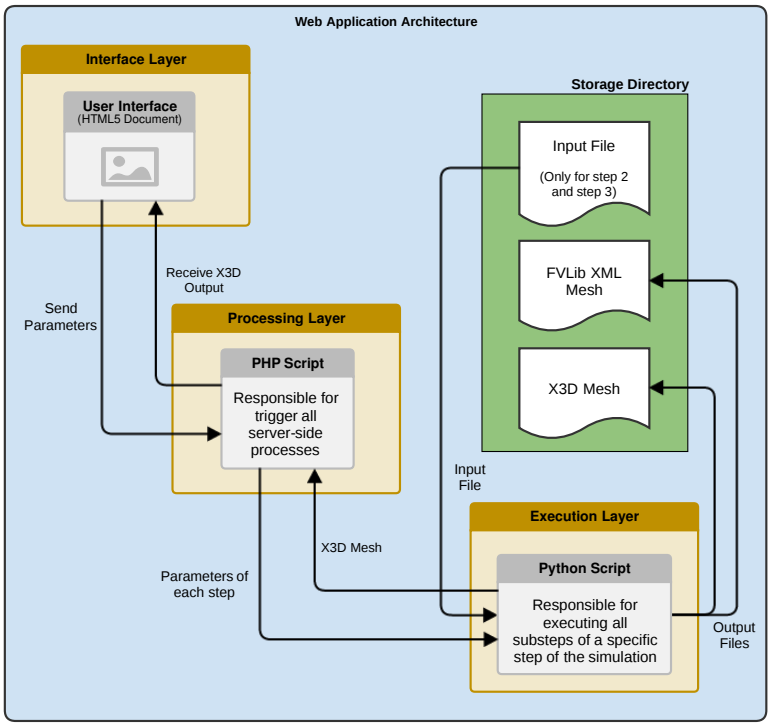


Figure 4.1: Web Application Architecture

Cache Feature

In the Processing layer it was added a feature that checks the user request/parametrization, and if there is an already calculated mesh with the same set of parameters, that file is directly presented to the user, without the need to recalculate it. This feature acts like a cache, allowing the user to switch between the simulation steps without loosing time calculating what it's already been calculated.
When the cache system is triggered, the Execution layer is not reached, because the result is directly obtained by the Processing layer.

The cache system is only activated if the previous parameters were equal to the actual parameters. If this is true, it means that the previous calculated result is the same as the correct result. Due to this, the recalculation time is saved, and the last result is immediately presented. This can vary between seconds to minutes depends on the selected mesh quality.

4.1.2 Web Application Development

The development of the web application was made in several stages. In a first step it was made a Gmsh to X3D format converter. This is necessary to present a 3D model of the breast in the web browser, and Gmsh is not suitable for that task.

Then, it was made the first prototype, consisting of a web page that was made with HTML forms with all possible parameters of the simulation, an upload area where the user should upload the initial parametrization of the breast using the Gmsh geometry file format and a frame in a corner of the page that shows the result of the simulation.

After receiving some feedback of the team that was developing the simulation software, we concluded that this design/architecture was good enough for a first approach, but not enough to be used by surgeons.
The initial parametrization was still made through the scripting language provided by Gmsh and during the simulation process, users could not make any changes or visualize the intermediate steps until the simulation end. The solution was to split the simulation process in three main steps: the initial parametrization and modeling of the breast; the surgery itself; and the displacement step. Each one of this steps have it's own parameters and 3D model of the breast. To achieve this design we automated all the steps using a Python script. Also, in the first step, the application generates the Gmsh script that was necessary to upload in the previous version, based on a few parameters that describe the breast geometry and the incisions that will be made during the surgery. Note that all calculations are performed on server-side. At this point the web application was divided into three pages, representing the three steps of the simulation. According to all project members, this version was much better than the previous one and theoretically, it could be used by anyone that have medical knowledge.

The development process was made in parallel with the development of the simulation software, introducing incremental updates which should be included in the web application. With these updates, it was suggested to introduce an alternative method of inputting the breast parameters in the first step, namely a visual method where the surgeon can understand the relation between all parameters and the impact of changing any of them.

After this suggestion, we started to build the interactive diagram that is present on the actual version of the web application (Fig. 4.8). To do so, we used a JavaScript library, called *JSXGraph* that creates a very intuitive layer to build interactive geometric shapes in web pages. The development of this 2D diagram was a complex task, since the geometry related with the 2D representation of the breast must be accurate, otherwise the parameters will be inconsistent.To do so, we introduced the mathematical formulas that define the breast model into the logic of the 2D diagram.

After we made a stable version of the web application with the diagram we received some good reviews about the application, including reviews from a doctor and a physician, part of the project team, which appreciated the work made on the web application.

4.2 Automating the Simulation Process

The surgery simulation process described in the Chapter 3 has a complex work-flow, so we needed to simplify the overall structure of the simulation in order to make it compatible with a web application.

The first thing to do is to group the activities related with each step of the simulation, into one single tool that receives as input all the parameters related with each step of the simulation. Figure 3.1(page 12) shows the grouping of the various steps of the simulation. To address this problem, it was decided to build a Python [41] application that will hide all the subtasks needed by each step. Each simulation step will be associated with a Python application that abstracts all the subtasks needed by the step. The choice of Python is related to the fact that this is a very powerful tool in several domains, from scientific computation to simple scripting tasks. Due the simplicity of the Python syntax, the automation of the several steps becomes more simple to implement

First Step - Breast Parametrization

The first step is about the definition of the initial model of the breast and it's parametrization. As represented in Fig. 3.1, the generation process of the breast mesh from the initial geometry file, and it's conversion to the FVLib XML format must be automated. The parameters file must be defined with the parameters in this step. To do so, was created a Python application that receives as input the following parameters (as described in Fig. 3.2):

- Radius of the breast (R distance)
- Depth of the breast (H distance)
- Angles of frontal incisions (α and β angles)
- Angle of suture (θ angle)
- Distance between chest and the bottom incisions (hl and hr distances)
- The new position of the nipple (S distance)
- Adjustments on the incision planes (points dl and dr)
- Mesh Quality

The first task to be done is the creation of the Gmsh geometry file with these parameters. This is made with a template file that is modified by the Python application (See Annex A for detailed information about this template). In this template the following variables are updated:

- R, which is the Breast Radius in meters
- H, which is the Breast Depth in meters
- a_left, which is the Left Incision Angle in radians
- a_right, which is the Right Incision Angle in radians
- s, which correspond to the s distance in Figure 3.2(page 15)
- dl, which is the dl distance in Figure 3.2
- dr, which is the dr distance in Figure 3.2
- $h2l$, which is the distance of the Bottom Left Incision with a shift
- $h2r$, which is the distance of the Bottom Right Incision with a shift
- lc, which is the Mesh Quality Coefficient (0.15 - Thin; 0.2 - Normal; 0.3 - Coarse). Scaled with the Breast Radius. Directly influences the distance between vertices.

After that, the Python application will generate the geometry file using Gmsh, and convert the resulting mesh into the FVLib XML format and X3D, both a representation of the defined breast model. The FVLib XML file must be generated because the next step of the simulation will need this file to proceed, but in order to show the result of this step in the browser it must also be converted to X3D. The X3D conversion process is explained with detail in Section 4.2.1.
This first process also updates some parameters in the global parameters file used in the surgery simulation. The work-flow of the first step can be visualized in Figure 4.2.

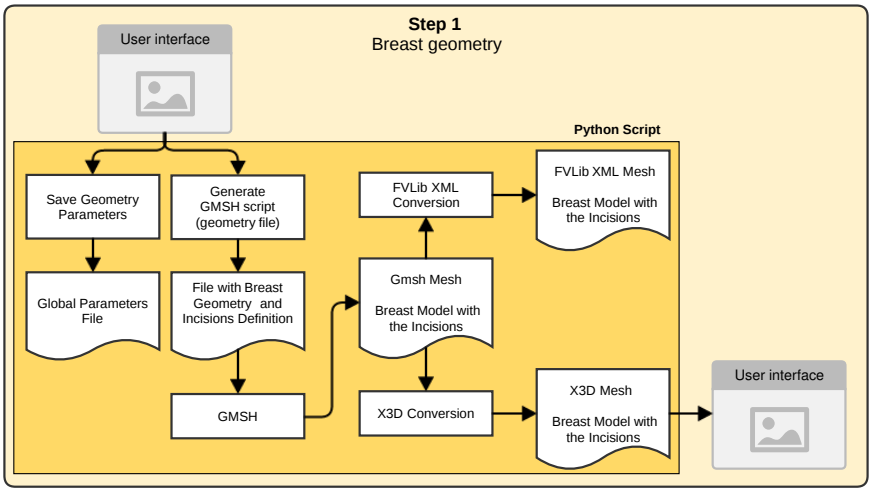


Figure 4.2: Architecture of the first step.

Second Step - Surgery Simulation

The second step is about the surgery simulation itself and the parametrization of some aspects related with the breast dynamics. So, the Python application for this step will receive as input the file generated in the last step alongside the following parameters:

- *Lambda* and *Mu*, which are two coefficients that define the breast as an elastic material.
- *Breast Density*, which is the density of the breast, i.e. weight divided by volume.
- *Chassignac Coefficient*, which is the coefficient that defines the connection between the breast and chest, as a spring.
- *LambdaP* and *MuP*, which defines the elasticity of the breast skin.
- Skin Thickness, which is the thickness of the breast skin.

With these inputs, the Python program will add the new parameters to the global parameters file and execute the surgery simulation program.
The surgery simulation process returns a FVLib XML file with the breast model after the suture. After that the FVLib XML file is converted into X3D in order to be visualized in a web browser. The surgery process can be seen in Figure 4.3. The X3D conversion process is explained with detail in Section 4.2.1.

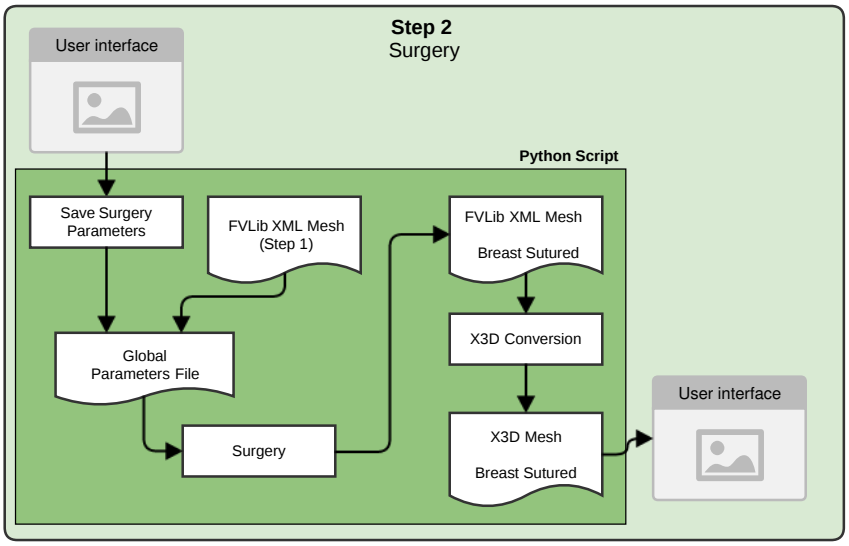


Figure 4.3: Architecture of the second step.

Third Step - Application of Gravity

The last step is the application of a gravity field to the breast model, also called the displacement process due to the displacement that the breast will suffer, in order to simulate the gravity. This step is very similar to the previous one, also having a Python script that processes all the parameters.
The application receives as input the result of the previous step, which is the breast model after the suture, along with the parameters of this step, which in this case is a vector that represents the gravitational field that will be applied. The application will add the new parameters to the global parameters file and execute the displacement simulation software. This software will calculate the amount of displacement for each breast cell according to the given gravity field, returning a special FVLib file that represents the final displacement. After that, the displacement must be applied to the previous step. This merge is made with the help of FVLib, which has a tool specifically made for this task. The result of the converter is a Gmsh mesh that must be converted to X3D in order to be viewable in a web browser. The X3D conversion process is explained with detail in Subsection 4.2.1. All the stated workflow is shown in Fig. 4.4.

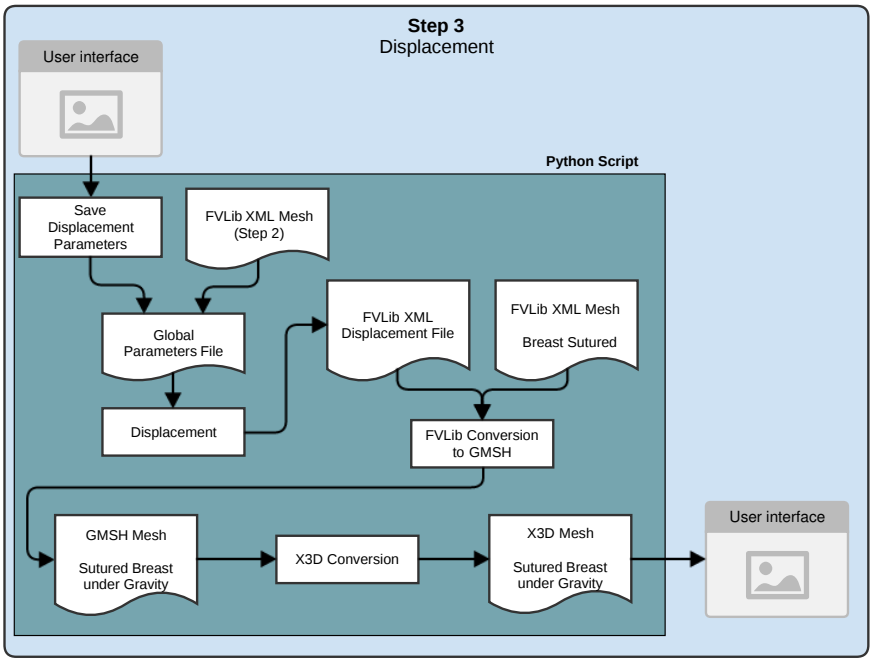


Figure 4.4: Architecture of the third step.

4.2.1 X3D Converters

As mentioned in Section 3.6, the simulation software relies on two file formats to represent the 3D models. These formats are the Gmsh and the FVLib XML file formats, detailed

in Sections 3.6.1 and 3.6.2.
Without the web interface requirements, these two types of 3D representations are enough, but to present the 3D models in the web browser, we must convert them to the X3D file format. To do so, we need two converters, one to convert the Gmsh format to X3D and another one, to convert the FVLib format to X3D. The first one (Gmsh to X3D) will be used in the first and last step of the simulation process, since in this steps, the 3D model is in Gmsh format. In the second step we only need to use the FVLib-X3D converter, since the simulation process only has the 3D model in the FVLib format, which is used between the simulation steps.

The conversion process from Gmsh format or FVLib format, is an iterative process. The input file is read and loaded into a data structure stored in memory, then, in a second stage we iterate over the data structure and write the X3D output file. This task must be done in two steps because the conversion process needs to know the entire structure of the input file before doing the conversion. The language used to write the converters was Python, since it's a powerful language that allows an easy approach to a scripting problem like these converters.

X3D Format

As referenced in Section 2.3, X3D is not new and is an improvement to the VRML format. The X3DOM is like a bridge, used to present X3D scenes in a web browser. X3DOM is also a candidate to the HTML5 declarative 3D standard.

The X3D format is based on XML tags and declarative by nature. Each X3D file defines a scene graph, containing the 3D world that will be created, where each node in the graph is an instance of the available node types, i.e.: a `Shape` node defines a new shape and a `Transform` node transform the child nodes (position, rotate and scale) [18].

To build a complex 3D shape, such as a breast model, the best node type to choose is the `IndexedFaceSet`, since it allows the definition of a 3D shape based on vertex coordinates that together form a face. Akin to the `IndexedFaceSet` node, there is the `IndexedLineSet`, but while the first one builds a set of faces that compose the shape (surface), the later is used to build the edges of the shape (wireframe).

The `IndexedFaceSet` and the `IndexedLineSet` are made by a list of the vertex coordinates and a list of vertex indexes, using the following syntax:

```
1 <Coordinate point = 'x1 y1 z1 x2 y2 z2 xn yn zn'/>
2 <IndexedFaceSet coordIndex = '0 1 2 3 -1 4 5 6 0'>
```

Listing 4.1: Example of X3D syntax

Line 1 of Listing 4.1 defines a list of coordinates only separated by space and Line 2 defines a construction of the faces with the indexes of coordinates, separated by -1. For example the index 0 reference the first tuple of coordinates defined and so on.

A simple example of a cube built with this kind of nodes is represented in Figure 4.5 and can be modeled as in Listing 4.2.

```
1  <X3D>
2      <Scene>
3        <Shape>
4           <Appearance>
5               <Material diffuseColor='0 0.5 1'/>
6           </Appearance>
7           <IndexedFaceSet coordIndex='0 1 2 3 -1 7 6 5 4 -1 0 4 5 1 -1 1
8           5 6 2 -1 2 6 7 3 -1 3 7 4 0'>
9              <Coordinate point='-1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 -1.0 -1.0
10             1.0 -1.0 -1.0 -1.0 1.0 1.0 -1.0 1.0 1.0 -1.0 -1.0 -1.0 -1.0
11             -1.0' />
12          </IndexedFaceSet>
13       </Shape>
14    </Scene>
15 </X3D>
```

Listing 4.2: X3D Source-Code for a Cube

In Listing 4.2, there is a root node (Lines 1 and 15), called `<X3D>`, meaning that his content complies with the X3D standard. After that, the Scene graph is defined through the nodes `<Scene>` (Line 2) and `</Scene>` (Line 14). Inside the Scene graph are defined all the desired 3D objects, each one in a separate `<Shape>` node (Lines 3 and 13). In the given example the `<Shape>` node is made by two children nodes: the `<Appearance>` node (Lines 4 and 6) and the `<IndexedFaceSet>` node (Lines 8 and 12).

The `<Appearance>` node is used to define all the aspects related with the aspect of the 3D object, such as color and texture. In this simple case it is only defined a simple diffuse color through the `<Material>` node. The desired color is defined with a Red, Green and Blue (RGB) tuple, representing the percentage of RGB colors needed to form the new color. In this case, to achieve the blue color showed in Fig. 4.5, it was used 0% of red, 50% of green and 100% of blue.

The `<IndexedFaceSet>` node defines the geometry of the 3D object. The `coordIndex` field defines the cube faces referencing the tuples defined in the children `<Coordinate>` node. For example, when the number 0 is written in the `coordIndex` the first coordinate inside the `<Coordinate>` node is referenced, in this case the $(-1.0, 1.0, 1.0)$ tuple. All the faces defined are separated by -1 which does not reference any coordinate.

The `<Coordinate>` (Line 11) node, which is a children of the `IndexedFaceSet` node contains all the coordinates used to define the desired 3D object, as shown in line 1 of Listing 4.1.

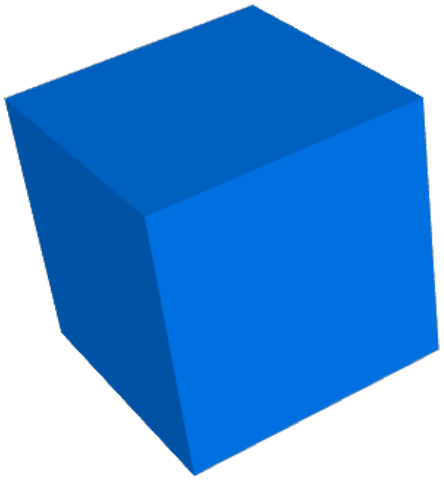


Figure 4.5: 3D representation of a cube

4.3 Web Interface

The chosen interface design is very simple, composed by two main areas: a lateral parameters input and a viewing area.
The user input for each step is grouped in the lateral area exclusively dedicated for the parameters needed by the simulation. The viewing area is mainly composed by the 3D breast model, with the traditional interactive controls (zoom, pan and rotate). The viewing area also includes a set of buttons that allows to rotate the 3D model to some predefined positions, according to the medical jargon (Fig. 4.11):

- The **Saggital View**, according to saggital plane, perpendicular to the ground, divides the human body into left and right. (Fig. 4.6)
- The **Coronal View**, according to coronal plane, also perpendicular to the ground, divides the human posterior and anterior portions. (Fig. 4.6)
- The **Axial View**, according to axial plane, parallel to the ground, divides the human body into superior and inferior parts. (Fig. 4.6)

In the first step the viewing area is slightly different from the other steps which is shared between the interactive diagram and the 3D model of the breast. The viewing area was splitted in this way due the user focus on what really matters, the modeling of the breast.

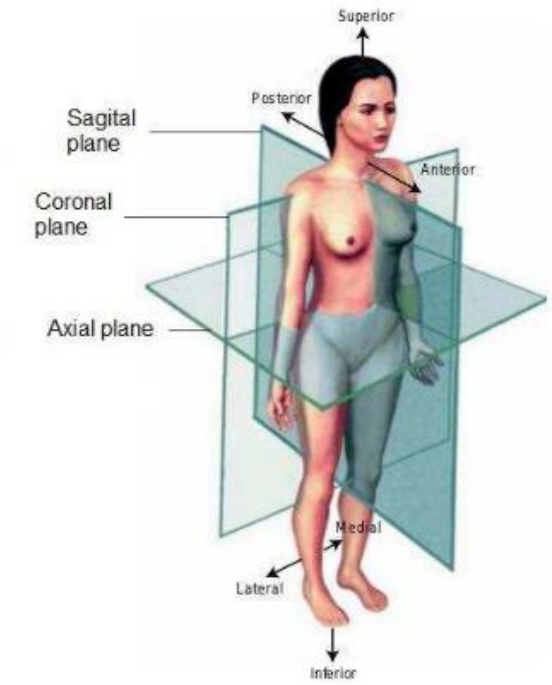


Figure 4.6: Anatomical Planes. Retrieved from [29, page: 34]

To make the user interface easier to use and more intuitive, we followed some of the heuristics created by Jakob Nielsen and Rolf Molich to Interface Design [36]. Some of them are: *Visibility of the system status*, which is implemented through some status boxes with the status of the system; *Consistency and Standards*, achieved with a global consistency with keywords and users commands; *Flexibility and Efficiency of use*, present in the parameters input form where advanced users can recalculate the breast model pressing the *Return* key; *Help and Documentation*, implemented in the X3D visualization where users can access a page that explains how to use the system; and the *User Control and Freedom*, which is implemented across the entire interface, including the required waiting time between each step where the user is free to abort the calculation process and return to the last state. This heuristics are used in all steps, maintaining the consistency between all steps.

4.3.1 First Step - Defining Breast Geometry

The first step of the breast reduction surgery is the modeling of the breast, which should be made with the highest fidelity as possible, otherwise the surgeon and the patient, will trust on a model that doesn't correspond to the reality, leading to wrong simulation results.

To avoid these kind of problems, which are very difficult to solve, the breast modeling can be made in two ways. The first one, less intuitive but more accurate is based on a set of parameters, and the second one more intuitive but less accurate is based on an interactive 2D diagram, where the user has the freedom to move key points of breast model in the

diagram. This diagram is presented in detail in the following paragraphs. According to the medical team these two methods are not exclusive and should be used together. The second one should be used to start the modeling process and the first one to finely tune the model already created.

All the adjustments made are shown in a 3D model in real-time, inside a frame that can be maximized in order to see the breast model with more detail. The surgeon can also fully interact with the 3D model, zoom, pan, rotate, change and reset the view.

These components can be visualized in Figure 4.7, that presents the layout of the interface for the first step.

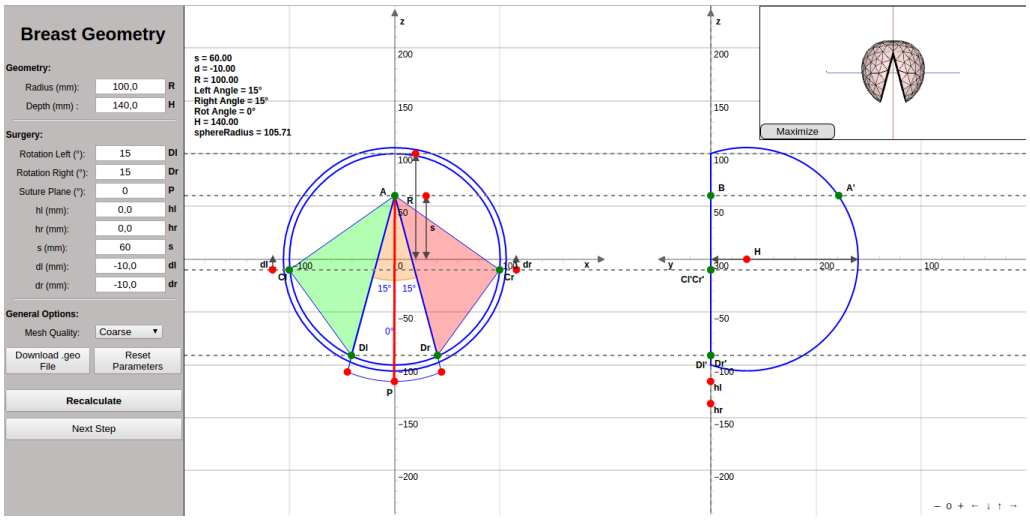


Figure 4.7: Screenshot of the first step of simulation.

Classic Parameter Input

As previously mentioned, there are two different ways to model the breast. One of the options is the use of a HTML form where the users write the desired values into the parameters field, pressing the Recalculate button when all parameters are set (See Fig. 4.7). This type of parameter input is useful when the user is familiarized with the Web Application and knows exactly what he wants or to finely tune the breast model, otherwise, users should start the parametrization of the breast model with the interactive 2D diagram (See Section 4.3.1), since it is easier to change the parameters and the consequences are shown in real-time.

Interactive Parameter Input

To allow an easy modeling of the breast, it was designed an alternative method to input the required parameters. This approach is based on the diagram presented in Figure 3.2 (page 15), which is the basis of the calculations made during the simulation process.

This interactive diagram was entirely made with JavaScript, with the help of a library called *JSXGraph* [6], made in the University of Bayreuth, Germany.

The drawing of the diagram itself was a simple task. Meanwhile we had problems when we wanted some elements to be dependent of each other: so when the user changes the hr distance, the diagram should reflect this change on points Dr and Dr', which are defined by a set of complex formulas that must be calculated. The same thing happens with the Radius (R) or the Depth (H) of the breast, when one of these measures changes, the entire diagram must be adapted to the new parameters. This problems were solved by using the formulas used to create the initial 3D breast model, together with many "invisible" auxiliary geometric constructions, used to help in the creation of the dependencies inside the diagram. An example of these auxiliary geometric constructions are the two "invisible" semi-circles used to restrict the path of each red point that defines the incisions angles. Without these "invisible" constructs the user was free to drag this two red points to wherever he wanted, causing a potential inconsistency in the diagram.

Another problem is the consistency between the measures introduced in the interactive diagram and the corresponding values found in the lateral input. The changes made inside the diagram are relatively easy to transform into the form values, but the diagram transformations according to the values inserted in the form are more difficult. The main problem is related with the angles, since with the existing dependencies, the library does not allows to set the angle value that two segments should have, so the solution is to move the only free point of an angle to the position corresponding to the value entered in the form. These calculations were made with the help of formulas that explain the diagram dependencies with the necessary improvements to achieve what we want, since the original formulas don't reflect these kind of problems, and were not made to solve these problems.

Along with the previously described problems, we also had problems with some missing features on *JSXGraph* that we need. An example is the case of the two collinear axes present in the diagram, for example. To achieve this, we had to modify the library in order to adapt it to our needs (See next paragraph for details).

Figure 4.8 shows the geometric diagram with more detail.

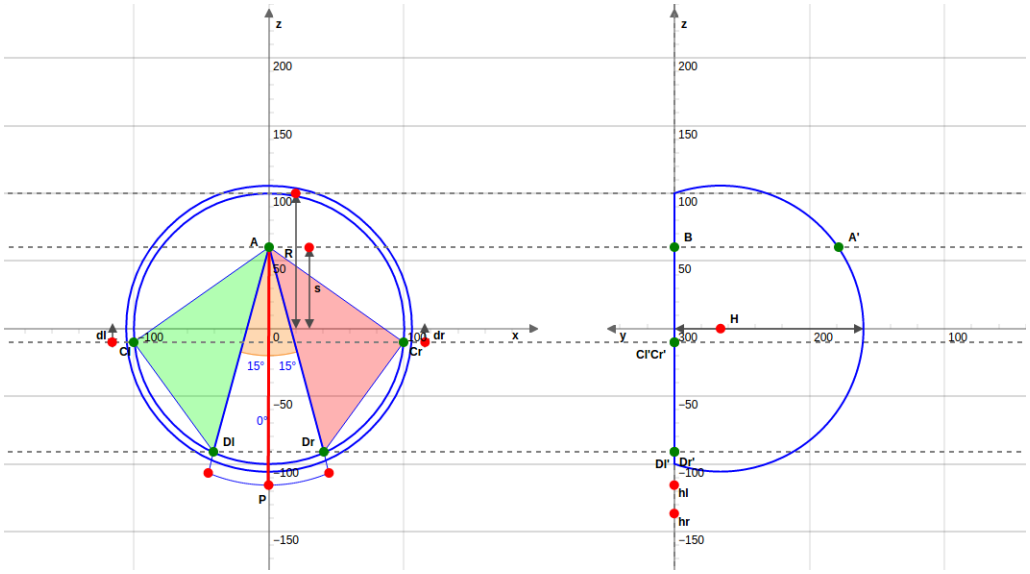


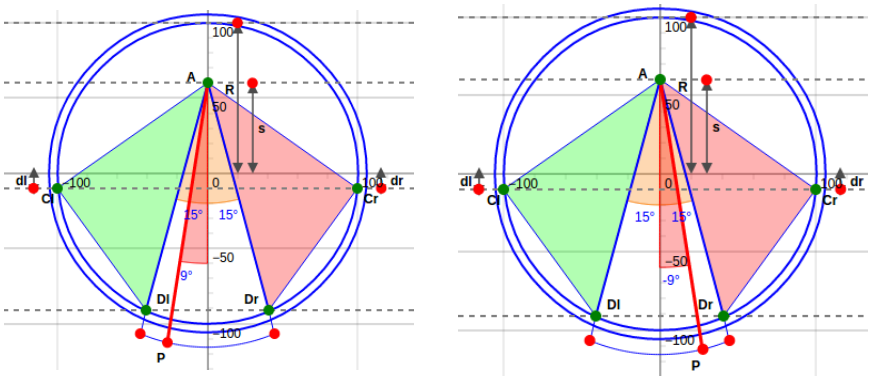
Figure 4.8: Screenshot of the geometric diagram.

JSXGraph Modifications

The modifications made to the library are related with two problems, the first one associated with the angle representation of the suture plane and the second related with the creation of two collinear axes.

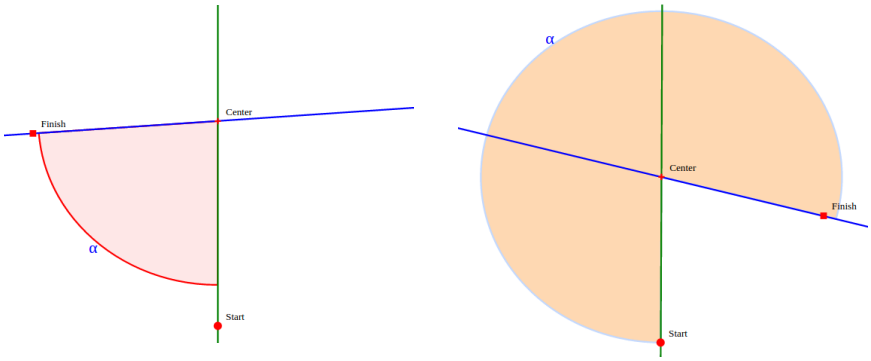
The suture plane angle is a specific angle measured between z axis and the suture plane, which is positive for the left side and negative in the right side. See Figure 4.9. The default representation of the library is defining an angle with three points: the center, the start point and the finish point. The last ones can be free to move. In this case we have two fixed points (center and start point) and the finished point is free (to allow user define the angle), so when the free point overtakes the start point, the angle is measured about 360 deg, because it is measured from the start point to the finish point. This problem can be better viewed in Figures 4.10a and 4.10b. To address this problem was created a new flag in the angle configuration, which when set to true the library will draw minimum angle possible whether from start to finish or from finish to start. If this new flag is set to false, the angle will have the normal behavior. This flag can be easily changed when the angle is built.

Since the main diagram is made of two views (frontal and lateral) we need to draw two collinear axes (x and y) (Fig. 4.8). However, collinear axes are not natively supported by *JSXGraph*. Along with the collinear axes problem, the two axes must have opposite direction in the diagram, which in case of the y axis seems to be a counter intuitive. The y axis(Fig. 4.8) has an opposite direction in relation to the original diagram(Fig. 3.2), because the user can change the breast depth (H) and with the original direction it is



(a) (b)

Figure 4.9: Suture Plane Angle (P) representation.



(a) (b)

Figure 4.10: Angle representation problem.

possible to overlap two views (lateral and frontal), when user increases the depth of the breast.

In order to draw an axis, *JSXGraph* uses others defined elements, namely *Line* and *Ticks*. The *Line* element is used to draw the axis line and *Ticks* element is used to add ticks to the created line. To solve the collinear axes problem, we added a new flag to the axis element that indicates if the axis is collinear with other axis, along with the start and end coordinates of the axes. If the collinear flag is set to true the axis will use a *Segment* element instead of a *Line*, limiting the space occupied by the axis and solving the collinearity problem of the axis. Regarding to the direction of the y axis, the *Ticks* element has an option that allows to specify the direction of the ticks labels, but this option is not accessible when we create a new axis. To solve this problem, if the collinear flag is active the axis options are passed to the *Ticks* element, allowing the modification of a specific *Ticks* option when we create an new collinear axis.

4.3.2 Second Step - Surgery Simulation

The second step is dedicated to the surgery simulation itself. In this step, the incisions made in the previous step will be sutured, resulting in a post-surgery breast model in a gravity free environment. The suturing process is determinant to the final shape of the breast, so it is very important for the surgeon to spend some time tuning the coefficients related to the breast dynamics, described in Section 4.2 (page 31), e.g.: *Skin Thickness* or the *Breast Density*, among others.

The interface of this step is simpler than the previous one, due to the absence of the geometric diagram. It is made of a lateral area where the surgeon inputs the parameters and a viewing area where the surgeon can inspect the result of the given parameters. Figure 4.11 shows the interface of the second step.

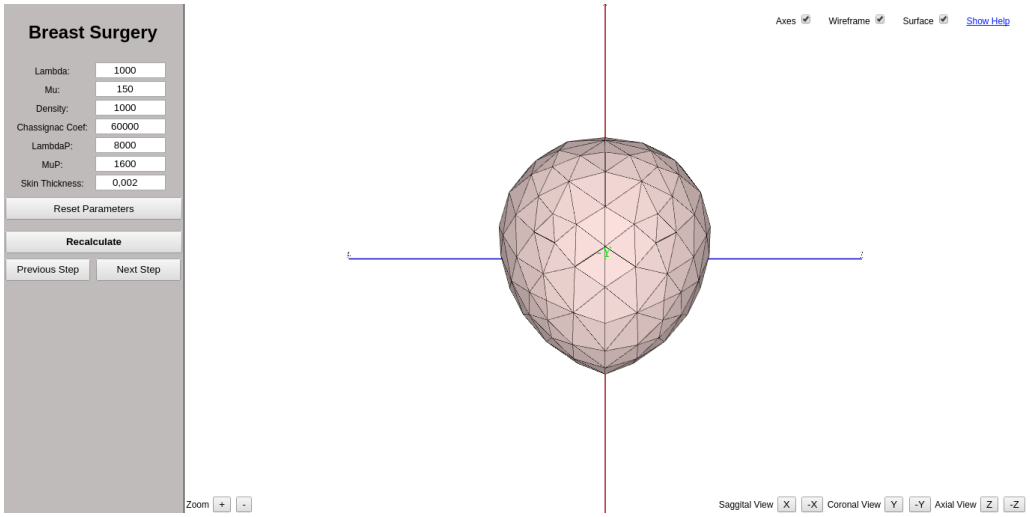


Figure 4.11: Screenshot of the second step of simulation.

4.3.3 Third Step - Gravity Simulation

The last step is the application of a gravitational field to the previous calculated breast model. The interface is consistent with the previous one, with a viewable area and a parameters input area. The gravitational field is adjustable by a vector that represents the gravity that will be applied on each axis of the breast model. The input parameters area also have a set of buttons with predefined "gravities" for *Patient Standing*, *Lying Down* and other relevant positions, although the user is still free to set the desired vector of gravity. Figure 4.12 show the interface for the third step.

The predefined gravities vectors associated to the buttons are defined in Table 4.1.

Patient Position	Gravity Vector
Standing	$(0, 0, -9.81)$
Lying Up	$(0, 9.81, 0)$
Lying Down	$(0, -9.81, 0)$
Standing Forward Bend	$(0, -\sin(45^o) \cdot 9.81, -\cos(45^o) \cdot 9.81)$

Table 4.1: Predefined gravity vectors.

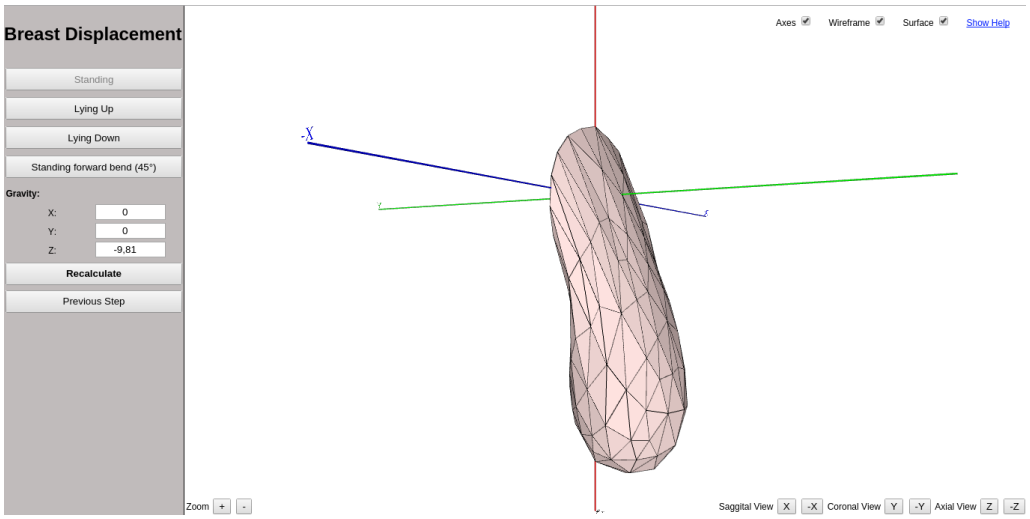


Figure 4.12: Screenshot of the third step of simulation.

4.4 Conclusions

In this chapter we described the improvements made to the original simulation software in order to implement the web application described in this work. The first part of the chapter (Section 4.2) describes the improvements made in order to create an abstraction layer between the web interface and the simulation software presented in Chapter 3. The second part of the chapter (Section 4.3) describes the interface of the application, its features and the connection between the simulation software and the web application.

Chapter 5

Overall Evaluation

In this chapter we present the overall evaluation of the work presented in this thesis. In particular we present a qualitative evaluation, a performance evaluation and finally a Medical feedback about the project and the work done.

5.1 Introduction

The evaluation of the web application is a very important task that cannot be neglected. Due to the evaluation being a subjective task we divided it into three areas: a "personal" evaluation, as a general critical review; the performance of the simulation process, measured in terms of execution time; and a general evaluation made by a medic.

The qualitative evaluation was made by us but from the user perspective, i.e. we tried to think as a normal user in order to detect problems which are not easily seen to the developer.

Regarding the simulation performance we measure and evaluate the simulation execution time in different machines with different types of mesh qualities. This perspective give us a quantitative feedback of the general performance, allowing us to compare the simulation performance with different mesh qualities and in different environments.

The medical evaluation is an evaluation made by Dr. *Igor Vasilevskiy*, member of the VAPS project.

Using these three different types of evaluations we plan to achieve a more complete overall assessment, using different perspectives.

5.2 Qualitative Evaluation

Regarding the simulation performance, the execution time is not the best, namely in the Suture Simulation (Step 2) and in the Gravity Application (Step 3). The calculation time is very high, specially when a high quality mesh is used, making it difficult to change parameters depending on the results.
The execution time depends directly on the performance of the simulation software. The simulation software used in Step 2 and Step 3 is based on an iterative algorithm that performs complex calculations for each node of the 3D breast model, so, as the number of nodes inside the mesh increase, the execution time also increases. This problem was target of a big improvement by the researchers involved in this task, since at the beginning of our work the total execution time was almost four hours, now it is reduced to about four minutes. The development of the simulation software is an ongoing task, so there is space for many improvements.

As for the normal use of the web application, it has the expected behavior on typical browsers as *Google Chrome* or *Mozilla Firefox*, however it doesn't work as expected in *Internet Explorer*. When using *Internet Explorer*, the breast model becomes static and it doesn't update according to the introduced parameters.
This problem was detected in an earlier stage of the web application development, but to solve it, we need to write specific code for the *Internet Explorer*, decreasing the focus on what is really important, an adequate operation of the web application, so, we focused on *Google Chrome* and *Mozilla Firefox*. Due to this reasons, the developed web application is not compatible with *Internet Explorer*, however the necessary improvements to make the web application compatible with *Internet Explorer* are included in the *Future Work* of this thesis (See Chapter 6).

There are still some problems with the simulation that have to be taken into account, but we must not forget that the simulation software is still in development and there is still no version without problems. Globally, the web application looks good and allows to do what is designed for, bridging health professionals and a mathematical approach that simulates a specific kind of surgery. However, as mentioned before, this is not a version ready to be used in the real world.

5.3 Simulation Performance

The evaluation of the simulation performance is an important task that cannot be neglected. As mentioned in Section 5.2, there is a problem related with long execution times of the simulation process, however, with a qualitative evaluation we cannot check how long the execution time is. The duration of the simulation is an important aspect to the final user, since it is the only performance related issue that affects the normal use of the simulation tool.

In order to measure the execution time of the simulation, it was written a simple *Python* script, that simulates the execution of the simulation, in the same way as the web application does, saving the performance results in a log file. In order to obtain reliable results, this tool executes the simulation ten times in a row for each mesh quality. All the simulations executed with this tool include the execution of the three steps of the simulation, generating exactly the same files that are generated by the web application. The simulation parameters used were always the default parameters of the web application, maintaining the consistency between several executions. At the end of the simulations, this tool calculates the average times as well as the maximum and the minimum time.

The simulation performance was measured in two different machines: 1) a computer used for developing and testing the web application (Table 5.1 - Machine 1); and 2) a machine used to host the web application and execute the simulation software, triggered by the web application (Table 5.1 - Machine 2). These machines are very different but the main difference between them is that Machine 2 is a virtual machine, which is natively supported by the host CPU using *VirtualBox*[14], versus the "real" system presented on Machine 1. The main hardware features of the used machines are described in Table 5.1.

	Machine 1	**Machine 2**
Operating System	Ubuntu 14.04 LTS	Ubuntu 14.04 LTS
RAM	4 GB	1 GB
CPU	Intel(R) Core(TM) i5 M430	Intel(R) Xeon(R) E5640
Architecture	64 bits	64 bits
CPU Frequency	2.27 GHz	2.67 GHz
Cores	2	1
Core Threads	2	1
Cache (L3)	3 MB	6 MB

Table 5.1: Description of used machines

The results obtained on the described machines, corresponding to ten simulation executions for each mesh quality, are shown in Table 5.2. The Machine 2 achieves better execution times than Machine 1, increasing the performance difference when the mesh becomes more thinner (higher quality).

This performance discrepancy can be explained by the type of operations made by the simulation software. The simulation software is basically based on a set of mathematical calculations iteratively done across the given breast model until it achieves a specific condition that indicates the end of the simulation. So, most of the operations are arithmetic calculations with almost no input/output operations. Assuming that the breast model is in memory at runtime, some of the most determinant factors that affect the execution time are the CPU frequency and the cache size, which also are the aspects where Machine 2 is better than Machine 1 (Table 5.1), explaining the performance increase on Machine 2.

	Thin Mesh		
	Average	**Minimum**	**Maximum**
Machine 1	254.92 s	230.87 s	281.26 s
Machine 2	210.29 s	204.47 s	214.36 s

	Normal Mesh		
	Average	**Minimum**	**Maximum**
Machine 1	100.98 s	100.35 s	101.58 s
Machine 2	67.12 s	65.57 s	70.92 s

	Coarse Mesh		
	Average	**Minimum**	**Maximum**
Machine 1	22.37 s	22.24 s	22.55 s
Machine 2	21.02 s	20.72 s	21.31 s

Table 5.2: Execution times using different meshes and machines.

Since the simulation software is not prepared to run in a multi-core environment, the performance increase observed on machine 2 comes from the hardware differences present on host machine which support the virtual environment.

5.4 Medical Evaluation

The medical evaluation has a special role in the evaluation of the work done so far, since the medical community, namely the plastic surgeons, are the target of the whole project, and without their feedback, all the work becomes closed on itself.

In order to receive some feedback about the work presented in this thesis, we prepared a set of questions to be answered by *Igor Vasilevskiy*, who is the representative of the medical community in the VAPS project. Despite not being a surgeon, he follows the project from the beginning, keeping in touch with plastic surgeons teams and establishing the bridge between the project and the real world.

Follows the list of question together with his answers:

1. **Regarding to the breast modeling process, is it easy to use? Is the initial model, presented in the first step of the simulation, realistic?**

   Actually the surgeon makes a drawing similar to the geometric diagram presented on the first step, so it is easy to use it because surgeons are familiar with it. The breast model without gravity in the first step can make it difficult to understand at first sight.

2. **Regarding the 3D model of breast generated across the simulation steps, is the interaction with the breast useful? Is it simple to use?**

   Yes, the 3D model is very useful and is easy to interact. One of the strongest points of the project, in my opinion.

3. **How easy can the simulation parameters be applied to an actual surgery?**

   The hard part is the measure of the breast in a neutral state (without gravity). Since it is not possible to measure the breast without gravity, it is necessary to measure it in several positions, which will always generate some uncertainty. So it will always be necessary to give some room for error in applying the parameters of the simulator to a real surgery.

4. **Are the simulation results close enough to the results of a real surgery?**

   The ideal similarity will never be achieved because in human beings, nothing is symmetrical. However, the breast shape and the postoperative positioning of the nipple-areola complex, are already quite realistic, representing an aid to the surgeon.

5. **Is it possible to the medical community to adopt the use of this project? In what form?**

   An experienced plastic surgeon knows the breast mechanics and has a good forecast of the final result according to the incision scheme applied. However, for surgeons who are not so accustomed to this type of intervention or the interns, this represent a good training tool. So, the use of this simulator goes through educational programs and support for plastic surgery interns, not ruling out the possibility of being used by experts, if it reaches a very high level of realism.

6. **If this project achieves enough maturity to be used as a planning tool for breast reduction surgeries, how do you see the adoption of this tool by health professionals?**

   It will depend on the doctor. Those who are interested in innovation will use willingly this tool for teaching purposes or for their own use, as for the more conservative surgeons they will hardly accept something like this. The simulator may also have some relevance to private clinics, especially when the reason for mammoplasty, is primarily aesthetic. It will be interesting to a surgeon, discuss with the patient in the office the final appearance of the breast through a graphic representation in real time.

7. **What are the differences between the use of simulation tool and a normal planning, in a real surgery preparation?**

   For now it is difficult to talk about difference, since the simulator has never been applied in a real surgery, I mean so far it has not been taken any surgical decision based on the simulation results. Ideally, the function of a simulator of this kind is to prevent surgical mistakes, without unexpected results in the postoperative phase.

There are several publications in medical databases describing errors in this type of interventions. The impact that a simulator of this kind must have is the maximum reduction of surgical errors. Plastic surgery is the only medical specialty that provides guarantees related to the postoperative result, so this match the plastic surgery needs.

8. **What are the strengths and weaknesses of the project?**

   As mentioned before one of the strongest points of the interface is the handling of the 3D model of the breast. I also highlight the interactivity of the surgical scheme and the possibility to input the parameters without writing values, but in an interactive way. Regarding to simulation itself, what is unprecedented in this project is the consideration of the Chassaignac space, which is very important for the breast biomechanics. About to the weakest points, there is the long waiting time between the steps of the simulation, the exclusion of the nipple-areola on the simulation calculus(although it is an absolutely necessary simplification), and the presentation of the breast model in a neutral state, which difficult the surgeon work.

9. **Regarding the usability of the interface, does it answers to the needs of the medical community?**

   In this case, the medical community is restricted to the plastic surgeons and their interns. The goal of medicine is generally to treat diseases and, if possible, create well-being. Plastic surgery adds the purpose of the creation of what is aesthetically good. Since this simulator is a tool that acts to reduce surgical errors and facilitate the work of health professional, is clearly a useful addition. This interface allows a surgeon to obtain a viable simulation with relative facility, after learning to work with the program, so we can say that the interface meets the medical community needs.

Through the questions answered by Dr *Igor Vasilevskiy*, we have the notion on what is appreciated by the plastic surgeons, and what is difficult the use by the surgeons. There is still a lot of work to be done, namely the consideration of the gravity across all the steps of the simulation and improving the simulation performance in order to have acceptable execution times compatible with a web interface. However, there are some aspects which are already appreciated by the health professionals, such as the interactive 3D model of the breast, the interactive surgical scheme to input the breast parameters and the consideration of the Chassaignac space in the simulation calculus.

5.5 Conclusions

This chapter evaluates the work presented in this thesis using three different perspectives: a qualitative perspective (Section 5.2), a technical point of view (Section 5.3) and finally a medical evaluation (Section 5.4). These three types of evaluation were useful because they pointed out the strengths and the weaknesses of the work made.

As a final evaluation, we conclude that the execution times are not the best for a web environment when a high quality mesh is used, and that the representation of the breast model in the initial steps could include a gravity field instead of presenting the breast in the neutral state. However, it is consensual that the web application presented in this work represents a considerable advance to the original simulation software, mostly due to the introduction of an interactive 3D model of the breast and an interactive method to model the breast shape.

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions about the work presented in this thesis, as well as further work that will improve some aspects in the future.

6.1 Assessment

Breast reduction surgeries are the most common surgical procedures in breast surgery and is becoming more important in women over 40 years of age [19]. However, mastering the surgical techniques takes time and becomes a challenge for less experienced surgeons to understand what should be done when facing a particular type of breast [19].

In the plastic surgery domain, but specially breast reduction surgeries, a simple mistake in the surgery planning can lead to an irreversible postoperative result [21, 19].

The work presented in this thesis establishes the bridge between plastic surgeons and an existent simulation software for breast reduction surgeries. We made a web application that contains the main steps of a real breast reduction surgery: the surgery planning, where the geometric definition of the breast is made, as well as the incision scheme that will be applied during the surgery; the surgical procedure itself; and finally, the postoperative result. Each step contains a 3D model of the breast that can be manipulated by the surgeon, in order to detect in advance errors that otherwise would only be detected during or after the surgery.

During the development of the web application, it was verified that the first step of the simulator was not very intuitive, mostly due to the manual introduction of several parameters that define the breast. To address this problem we adapted an existing 2D diagram of

the breast in order to make it interactive and to automatically adapt itself to any change made on the parameters that define the breast. This new method of specifying the parameters makes the simulator much easier and intuitive to use, decreasing the probability of errors in the planning phase. This new feature was well accepted by the members of VAPS project.

Throughout the work described in this thesis, we improved the way that surgeons interact with a set of simulation software for breast reduction surgeries. With this web application, in particular the less experienced surgeons, can train and prepare their surgeries in a simple and intuitive way.

The work presented in this thesis was published in a peer-reviewed international conference [43], and in a peer-reviewed national conference [40]. This work was also presented in the "Seminars"[39] class, part of the Masters program in Computer Science of Universidade de Évora.

6.2 Future Work

Although the web application is completely functional, it is not yet a finished work. There are still problems related with the web application and the simulation software behind the web application, which is also not finished.

As future work, we need to redesign the steps of the simulation to include the effect of gravity in the first step. The modeling of the breast using a free gravity environment wasn't a good choice, since surgeons have to *imagine* how the breast will be affected by gravity. This problem was also pointed out by Dr *Igor Vasilevskiy* in Section 5.4.

Another problem is the long time of the simulation process. This is a very complex problem, because the simulation is based on complex mathematical calculus that takes some time. We also plan to allow the remote execution of the simulation software, making it possible to run the simulation software in dedicated hardware, e.g.: a cluster, which would increase it's performance. Also, as future work, we plan to introduce a progress bar indicating the estimated time for each step to finish. This issue was also mentioned in Sections 5.2, 5.3 and 5.4.

About the web application itself, there is an issue related with the browser support. We tested the web application only on *Google Chrome*, *Mozilla Firefox* and *Internet Explorer*. At the moment, *Internet Explorer* is incompatible with the web application, needing special code to reload the 3D model on real-time. As future work, this incompatibility problem must be solved, since *Internet Explorer* is a common web-browser for *Windows* users. We also plan to test more browsers like *Opera* or *Safari* in order to extend the spectrum of supported browsers.

Regarding the 3D model of the breast and the interactive surgical diagram in the first step, it would be interesting to join them together, since they are appreciated by the members

of the project. The inclusion of the 2D diagram into the 3D breast will further improve the parameters input, since the 3D model will be directly modified through key points, and recalculated in real-time.

Bibliography

[1] Blender. http://www.blender.org/. Accessed: 2015-02-17.

[2] Breast Lift or Breast Reduction? http://www.oakleafmedical.com/hv/2012_sp/sp2012_breastsurgery.php. Accessed: 2015-02-11.

[3] Crisalix - Virtual Aesthetics. http://www.crisalix.com/pt. Accessed:2015-05-21.

[4] Epona Medical. http://www.epona.com/site/medical/. Accessed: 2015-05-21.

[5] Google Docs. https://www.google.com/docs/about/. Accessed: 2015-05-21.

[6] JSXGraph - Dynamic Mathematics with Javascript. http://jsxgraph.uni-bayreuth.de/wp/. Accessed: 2015-03-01.

[7] Khronos - WebGL. https://www.khronos.org/webgl/. Accessed: 2015-02-11.

[8] Photopea. http://www.photopea.com/. Accessed: 2015-05-21.

[9] Raphaël—JavaScript Library. http://raphaeljs.com/. Accessed: 2015-03-01.

[10] Scene.js. http://scenejs.org/. Accessed: 2015-05-21.

[11] SimSurgery. http://www.simsurgery.com/. Accessed:2015-05-21.

[12] Three.js. http://threejs.org/. Accessed: 2015-03-01.

[13] Vectra XT 3D. http://www.canfieldsci.com/imaging-systems/vectra-xt-3d-imaging-system/. Accessed:2015-05-21.

[14] VirtualBox. https://www.virtualbox.org/. Accessed: 2015-6-9.

[15] X3D and VRML The Most Widely Used 3D Formats. http://www.web3d.org/x3d-vrml-most-widely-used-3d-formats. Accessed: 2015-02-17.

[16] X3DOM - Instant 3D the HTML way. `http://www.x3dom.org/`. Accessed: 2015-02-17.

[17] Fred S Azar, Dimitris N Metaxas, and Mitchell D Schnall. Methods for modeling and predicting mechanical deformations of the breast under external perturbations. *Medical Image Analysis*, 6(1):1–27, 2002.

[18] Don Brutzman and Leonard Daly. *X3D: Extensible 3D Graphics for Web Authors*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[19] Augusta Cardoso, Gustavo Coelho, Horácio Zenha, Vera Sá, Georgi Smirnov, and Horácio Costa. Computer simulation of breast reduction surgery. *Aesthetic plastic surgery*, 37(1):68–76, 2013.

[20] J Cecil, P Ramanathan, Miguel Pirela-Cruz, and M Bharathi Raj Kumar. A virtual reality based simulation environment for orthopedic surgery. In *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*, pages 275–285. Springer, 2014.

[21] S. Clain, G. J. Machado, R. M. S. Pereira, and G. Smirnov. Soft tissue modelling for analysis of errors in breast reduction cirurgy. *11th World Congress on Computational Mechanics (WCCM XI)*, pages 1–10, 2014.

[22] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 5(1):62–73, 1999.

[23] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[24] A Pérez Del Palomar, B Calvo, J Herrero, J Lopez, and M Doblaré. A finite element model to accurately predict real deformations of the breast. *Medical Engineering & Physics*, 30(9):1089–1097, 2008.

[25] Herve Delingette. Toward realistic soft-tissue modeling in medical simulation. *Proceedings of the IEEE*, 86(3):512–523, 1998.

[26] Yan Hong Fang, Bin Wu, and Zheng Yi Yang. Study on virtual liver surgery simulation system with real-time haptic feedback. *Applied Mechanics and Materials*, 536:900–906, 2014.

[27] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[28] Christophe Geuzaine and Jean-François Remacle. Gmsh Reference Manual. `http://geuz.org/gmsh/doc/texinfo/gmsh.pdf`. Accessed: 2015-05-13.

[29] B.A. Gylys and R.M. Masters. *Medical Terminology Simplified: A Programmed Learning Approach by Body System*. F.A. Davis Company, 2009.

[30] Kazunori Hase and Nobutoshi Yamazaki. Computer simulation study of human locomotion with a three-dimensional entire-body neuro-musculo-skeletal model. i. acquisition of normal walking. *JSME International Journal Series C*, 45(4):1040–1050, 2002.

[31] V Hurmusiadis. Virtual heart: Simulation-based cardiac physiolgy for education. In *Computers in Cardiology, 2007*, pages 65–68. IEEE, 2007.

[32] Takakazu Kawamata, Hiroshi Iseki, Takao Shibasaki, and Tomokatsu Hori. Endoscopic augmented reality navigation system for endonasal transsphenoidal surgery to treat pituitary tumors: technical note. *Neurosurgery*, 50(6):1393–1397, 2002.

[33] Ines Langemeyer. Learning in a simulation-ot in heart surgery and the challenges of the scientification of work. *Journal of Education and Work*, 27(3):284–305, 2014.

[34] Jacques Marescaux, Francesco Rubino, Mara Arenas, Didier Mutter, and Luc Soler. Augmented-reality–assisted laparoscopic adrenalectomy. *Jama*, 292(18):2211–2215, 2004.

[35] David M McQueen and Charles S Peskin. Heart simulation by an immersed boundary method with formal second-order accuracy and reduced numerical viscosity. In *Mechanics for a New Mellennium*, pages 429–444. Springer, 2002.

[36] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.

[37] David Raggett. Extending WWW to support platform independent virtual reality. In *Proceedings of the 5th Joint European Networking Conference (JENC5)*, volume 464, page 2, 1994.

[38] Serge Rezzonico and Daniel Thalmann. Browsing 3D bookmarks in BED. In *WebNet*. Citeseer, 1996.

[39] José Rolo. 3D Web Interface for Plastic Surgery Simulation, 2014. Article presented in "Seminars" class of Master degree.

[40] José Rolo, Pedro Salgueiro, and Salvador Abreu. An hybrid 3d web interface for plastic surgery simulation. *Jornadas de Informática da Universidade de Évora*, 2015.

[41] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, 1995.

[42] Shinichi Saito, Junichi Yamanaka, Koui Miura, Norio Nakao, Tomohiro Nagao, Takaaki Sugimoto, Tadamichi Hirano, Nobukazu Kuroda, Yuji Iimuro, and Jiro Fujimoto. A novel 3d hepatectomy simulation based on liver circulation: application to liver resection and transplantation. *Hepatology*, 41(6):1297–1304, 2005.

[43] Pedro Salgueiro, Salvador Abreu, José Rolo, and Stéphane Clain. An Interactive Web-Based Tool for Breast Reduction Surgery Simulation. In *3D User Interfaces (3DUI), 2015 IEEE Symposium on*, March 2015.

Appendices

Appendix A

Gmsh Geometry File Template

The source-code presented in Listing A.1 represent the template used to generate the *Gmsh* geometry file.

```
// breast.geo

// Global Vars
MINUS_THREE_DIM = 2147483648;
MINUS_TWO_DIM  = 1073741824;
MINUS_ONE_DIM  =  536870912;

// parameters of the breast
R = [BREAST RADIUS]; // Breast Radius
H = [BREAST DEPTH]; // Breast Depth

a_left = [LEFT INCISION]; // Rotation Left Incision (RADs)
a_right = [RIGHT INCISION]; // Rotation Right Incision (RADs)
s = [S DISTANCE];// s distance
dl = [DL DISTANCE]; // dl distance
dr = [DR DISTANCE]; // dr distance

shift = r-H;
h2l = [BOTTOM LEFT DISTANCE] + shift;// bottom left distance
h2r = [BOTTOM RIGHT DISTANCE] + shift;// bottom right distance

lc = [MESH QUALITY] * R; // 0.15 - Thin; 0.2 - Normal; 0.3 - Coarse

// making the breast geometry
r=0.5*(R*R+H*H)/H;
```

```
a = Pi/15;
h=0.04;
epsi=0.0;
tga=Tan(a);
tga_left=Tan(a_left);
tga_right=Tan(a_right);
tgb_left=Tan(b_left);
tgb_right=Tan(b_right);



Point(1) = {0+epsi,0.0+shift,0.0,lc};//point O
Ax=0.0;Az=s;Ay=-Sqrt(r*r-s*s);
Point(2) = {Ax+epsi,Ay+shift,Az,lc};//point A
Ex=0.0;Ez=R;Ey=H-r;
Point(3) = {Ex+epsi,Ey+shift,Ez,lc};// point E


//------------//
Delta_bl=4*Az*Az*tgb_left*tgb_left*tgb_left*tgb_left-4*(1+tgb_left*tgb_left)*
(tgb_left*tgb_left*Az*Az+0.0*0.0-R*R);
delta_z=0.5*(2*Az*tgb_left*tgb_left-Sqrt(Delta_bl))/(1+tgb_left*tgb_left);
Dy=-tgb_left*(Az-delta_z);

Delta=4*Az*Az*tga_left*tga_left*tga_left*tga_left-4*(1+tga_left*tga_left)*
(tga_left*tga_left*Az*Az+Dy*Dy-R*R);
Dz=0.5*(2*Az*tga_left*tga_left-Sqrt(Delta))/(1+tga_left*tga_left);
Dx=tga_left*(Az-Dz);


Dy=-h2l+shift;
delta_M=Tan(-a_left+(Pi*0.5));
delta_A=1+(delta_M*delta_M);
delta_B=2*Az*delta_M;
delta_C=(Az*Az)+(h2l*h2l)-(r*r);
Dx=(-delta_B-Sqrt((delta_B*delta_B) - (4*delta_A*delta_C)))/(2*delta_A);
Dz=Az+delta_M*Dx;


Point(4) = {Dx,Dy,Dz,lc}; //point D

Printf("Ponto 4:: %g,%g,%g -- tan:%g -- r:%g -- delta:%g,%g,%g", Dx, Dy, Dz,
delta_M, r, delta_A, delta_B, delta_C);

Delta_br=4*Az*Az*tgb_right*tgb_right*tgb_right*tgb_right-4*
(1+tgb_right*tgb_right)*(tgb_right*tgb_right*Az*Az+0.0*0.0-R*R);
delta_z=0.5*(2*Az*tgb_right*tgb_right-Sqrt(Delta_br))/(1+tgb_right*tgb_right);
Dy_r=-tgb_right*(Az-delta_z);

```

```
75  Delta_r=4*Az*Az*tga_right*tga_right*tga_right*tga_right-4*
76  (1+tga_right*tga_right)*(tga_right*tga_right*Az*Az+Dy_r*Dy_r-R*R);
77  Dz_r=0.5*(2*Az*tga_right*tga_right-Sqrt(Delta_r))/(1+tga_right*tga_right);
78  Dx_r=tga_right*(Az-Dz_r);
79
80  Dy_r=-(h2r-shift);
81  delta_M=Tan(a_right-(Pi*0.5));
82  delta_A=1+(delta_M*delta_M);
83  delta_B=2*Az*delta_M;
84  delta_C=(Az*Az)+(h2r*h2r)-(r*r);
85  Dx_r=(-delta_B+Sqrt((delta_B*delta_B) - (4*delta_A*delta_C)))/(2*delta_A);
86  Dz_r=Az+delta_M*Dx_r;
87
88  Point(5) = {Dx_r,Dy_r,Dz_r,lc};
89
90  Printf("Ponto 5:: %g,%g,%g -- tan:%g -- r:%g -- delta:%g,%g,%g", Dx_r, Dy_r,
91  Dz_r, delta_M, r, delta_A, delta_B, delta_C);
92
93  //pontos laterais obtidos com um certo angulo
94  Cz=-dr;Cy=Ey;Cx=Sqrt(R*R-(dr*dr));
95  Point(6) = {Cx+epsi,Cy+shift,Cz,lc};// Point C
96  Cz_r=-dl;Cy_r=Ey;Cx_r=-Sqrt(R*R-(dl*dl));
97  Point(7) = {Cx_r+epsi,Cy_r+shift,Cz_r,lc};
98
99  Fx=0.0;Fz=0.;Fy=H-r;
100 Point(8) = {Fx,Fy+shift,Fz,lc};// point F centre of base circle
101 Bx=Ax;By=Ey;Bz=Az;
102 Point(9) = {Bx+epsi,By+shift,Bz,lc};// Point B
103 ux=(Cy-By)*(Dz-Bz)-(Cz-Bz)*(Dy-By);
104 uy=(Cz-Bz)*(Dx-Bx)-(Cx-Bx)*(Dz-Bz);
105 uz=(Cx-Bx)*(Dy-By)-(Cy-By)*(Dx-Bx);
106 lambda=(Bx*ux+By*uy+Bz*uz)/(ux*ux+uy*uy+uz*uz);
107 Gx=lambda*ux;Gy=lambda*uy;Gz=lambda*uz;
108
109 ux=(Cy_r-By)*(Dz_r-Bz)-(Cz_r-Bz)*(Dy_r-By);
110 uy=(Cz_r-Bz)*(-Dx_r-Bx)-(Cx_r-Bx)*(Dz_r-Bz);
111 uz=(Cx_r-Bx)*(Dy_r-By)-(Cy_r-By)*(-Dx_r-Bx);
112 lambda=(Bx*ux+By*uy+Bz*uz)/(ux*ux+uy*uy+uz*uz);
113 Gx=lambda*ux;Gy=lambda*uy;Gz=lambda*uz;
114
115
116
117 //center of arc
118 Bx=Ax;By=0;Bz=Az;
119 p1x=Cx; p1y=Cy+shift; p1z=Cz;
120 p2x=Dx_r; p2y=Dy_r; p2z=Dz_r;
121 n1 = ((p1y-By)*(p2z-Bz)) - ((p2y-By)*(p1z-Bz));
122 n2 = ((p2x-Bx)*(p1z-Bz)) - ((p1x-Bx)*(p2z-Bz));
123 n3 = ((p1x-Bx)*(p2y-By)) - ((p2x-Bx)*(p1y-By));
```

```
gamma = (p1x*p1x) - (p2x*p2x) + (p1y*p1y) - (p2y*p2y) + (p1z*p1z) - (p2z*p2z);

CCx=0;
CCz = (gamma - ( ( ((n1*Bx) + (n3*Bz))/(n2) ) * ((2*p1y) - (2*p2y)) ) ) /
((2*p1z) - (2*p2z) - ((n3/n2)*((2*p1y)-(2*p2y))) );
CCy = ((n1*Bx) + (n3*Bz) - (n3*CCz)) / n2;

Point(10) = {CCx,CCy,CCz,lc};

Bx=Ax;By=0;Bz=Az;
p1x=Cx; p1y=Cy+shift; p1z=Cz;
p2x=Dx; p2y=Dy; p2z=Dz;
n1 = ((p1y-By)*(p2z-Bz)) - ((p2y-By)*(p1z-Bz));
n2 = ((p2x-Bx)*(p1z-Bz)) - ((p1x-Bx)*(p2z-Bz));
n3 = ((p1x-Bx)*(p2y-By)) - ((p2x-Bx)*(p1y-By));
gamma = (p1x*p1x) - (p2x*p2x) + (p1y*p1y) - (p2y*p2y) + (p1z*p1z) - (p2z*p2z);

CCx=0;
CCz = (gamma - ( ( ((n1*Bx) + (n3*Bz))/(n2) ) * ((2*p1y) - (2*p2y)) ) ) /
((2*p1z) - (2*p2z) - ((n3/n2)*((2*p1y)-(2*p2y))) );
CCy = ((n1*Bx) + (n3*Bz) - (n3*CCz)) / n2;


Point(11) = {CCx,CCy,CCz,lc};

d_val = Sqrt((Dx-Ax)*(Dx-Ax) + (Dy-Ay+shift)*(Dy-Ay+shift) + (Dz-Az)*(Dz-Az));
beta = 1/(Sin((d_val*0.5)/R));
t = Cos(beta) * R; //dist

meanX = (Dx + Ax)/2.0; meanY = (Dy+(Ay+shift))/2.0; meanZ = (Dz + Az)/2.0;
meanDist = Sqrt((meanX-Bx)*(meanX-Bx) + (meanY-By)*(meanY-By) +
(meanZ-Bz)*(meanZ-Bz));
vx = (Bx-meanX)/meanDist;
vy = (By-meanY)/meanDist;
vz = (Bz-meanZ)/meanDist;



h_aux = Sqrt(R*R - (d_val*0.5)*(d_val*0.5));
v1x = -Tan(a_left - (Pi*0.5));
v1y = 0.0;
v1z = 1.0;

ddx = -v1z*(Dy-(Ay+shift));
ddy = -v1x*(Dz-Az) + v1z*(Dx-Ax);
ddz = v1x*(Dy-(Ay+shift));


P_cX = (meanX+h_aux*ddx);
```

```
P_cY = (meanY+h_aux*ddy);
P_cZ = (meanZ+h_aux*ddz);

teste = Sqrt((Ax-P_cX)*(Ax-P_cX) + ((Ay+shift)-P_cY)*((Ay+shift)-P_cY) +
(Az-P_cZ)*(Az-P_cZ) );
teste1 = Sqrt((Dx-P_cX)*(Dx-P_cX) + ((Dy)-P_cY)*((Dy)-P_cY) +
(Dz-P_cZ)*(Dz-P_cZ) );
teste2 = Tan(Pi*0.5-a_left)*P_cX - P_cZ + Az;


lambda = (-Az) / (Tan(-a_left+(Pi*0.5))*Tan(-a_left+(Pi*0.5)) + 1.0);
c_auxX = (lambda*(Tan(-a_left+(Pi*0.5))));
c_auxY = shift;
c_auxZ = -lambda;

Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((c_auxX-Ax)*(c_auxX-Ax) +
(c_auxY-Ay+shift)*(c_auxY-Ay+shift) + (c_auxZ-Az)*(c_auxZ-Az)),
Sqrt((c_auxX-Dx_r)*(c_auxX-Dx_r) + (c_auxY-Dy_r)*(c_auxY-Dy_r) +
(c_auxZ-Dz_r)*(c_auxZ-Dz_r)));
Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((c_auxX-Ax)*(c_auxX-Ax) +
(c_auxY-Ay+shift)*(c_auxY-Ay+shift) + (c_auxZ-Az)*(c_auxZ-Az)),
Sqrt((c_auxX-Dx)*(c_auxX-Dx) + (c_auxY-Dy)*(c_auxY-Dy) +
(c_auxZ-Dz)*(c_auxZ-Dz)));


Point(12) = {c_auxX,c_auxY,c_auxZ,lc};

teste = Sqrt((Ax-c_auxX)*(Ax-c_auxX) + ((Ay+shift)-c_auxY)*((Ay+shift)-c_auxY)
+ (Az-c_auxZ)*(Az-c_auxZ) );
teste1 = Sqrt((Dx-c_auxX)*(Dx-c_auxX) + ((Dy)-c_auxY)*((Dy)-c_auxY) +
(Dz-c_auxZ)*(Dz-c_auxZ) );
teste2 = Tan((Pi*0.5)-a_left)*c_auxX - c_auxZ + Az;
Printf("teste esq:: %g <-> %g -- isplane:%f -- lambda:%g",teste, teste1,
teste2, lambda);


d_val = Sqrt((Dx_r-Ax)*(Dx_r-Ax) + (Dy_r-Ay+shift)*(Dy_r-Ay+shift) +
(Dz_r-Az)*(Dz_r-Az));

beta = 1/(Sin((d_val*0.5)/R));
t = Cos(beta) * R; //dist

meanX = (Dx_r + Ax)/2.0; meanY = (Dy_r+(Ay+shift))/2.0; meanZ = (Dz_r + Az)/2.0;
meanDist = Sqrt((meanX-Bx)*(meanX-Bx) + (meanY-By)*(meanY-By) +
(meanZ-Bz)*(meanZ-Bz));
vx = (Bx-meanX)/meanDist;
vy = (By-meanY)/meanDist;
```

```
vz = (Bz-meanZ)/meanDist;


h_aux = Sqrt(R*R - (d_val*0.5)*(d_val*0.5));
v1x = -Tan(a_right - Pi/2.0);
v1y = 0.0;
v1z = 1.0;

ddx = -v1z*(Dy_r-(Ay+shift));
ddy = -v1x*(Dz_r-Az) + v1z*(Dx_r-Ax);
ddz = v1x*(Dy_r-(Ay+shift));

teste = (Dx_r - Ax)*Tan(a_right-(Pi*0.5)) - (Dz_r-Az);


P_cX = (meanX+h_aux*ddx);
P_cY = (meanY+h_aux*ddy);
P_cZ = (meanZ+h_aux*ddz);

teste = Sqrt((Ax-P_cX)*(Ax-P_cX) + ((Ay+shift)-P_cY)*((Ay+shift)-P_cY) +
(Az-P_cZ)*(Az-P_cZ) );
teste1 = Sqrt((Dx_r-P_cX)*(Dx_r-P_cX) + ((Dy_r)-P_cY)*((Dy_r)-P_cY) +
(Dz_r-P_cZ)*(Dz_r-P_cZ) );


P_cM = Tan(a_right-(Pi*0.5));
P_aux = (2.0*Dx_r + 2.0*P_cM*(Az-Dz_r));

P_cY = shift;
P_cX = ((Ay+shift)*(Ay+shift) - (2.0*P_cY*(Ay+shift)) - (Dx_r*Dx_r) -
(Dy_r*Dy_r) + (2.0*P_cY*Dy_r) - ((Az-Dz_r)*(Az-Dz_r))) / P_aux;
P_cZ = Az + P_cM*P_cX;

Printf("aux: %g", P_aux);


lambda = (-Az) / (Tan(-a_right+(Pi*0.5))*Tan(-a_right+(Pi*0.5)) + 1.0);

c_auxX = (-lambda*Tan(-a_right+(Pi*0.5)));
c_auxY = shift;
c_auxZ = -lambda;

Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((c_auxX-Ax)*(c_auxX-Ax) +
(c_auxY-Ay+shift)*(c_auxY-Ay+shift) + (c_auxZ-Az)*(c_auxZ-Az)),
Sqrt((c_auxX-Dx)*(c_auxX-Dx) + (c_auxY-Dy)*(c_auxY-Dy) +
(c_auxZ-Dz)*(c_auxZ-Dz)));
Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((c_auxX-Ax)*(c_auxX-Ax) +
```

```
(c_auxY-Ay+shift)*(c_auxY-Ay+shift) + (c_auxZ-Az)*(c_auxZ-Az)),
Sqrt((c_auxX-Dx_r)*(c_auxX-Dx_r) + (c_auxY-Dy_r)*(c_auxY-Dy_r) +
(c_auxZ-Dz_r)*(c_auxZ-Dz_r)));


AAy = Ay+shift;


Printf("PONTO 13");
Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((c_auxX-Ax)*(c_auxX-Ax) +
(c_auxY-(Ay+shift))*(c_auxY-(Ay+shift)) + (c_auxZ-Az)*(c_auxZ-Az)),
Sqrt((c_auxX-Dx)*(c_auxX-Dx) + (c_auxY-Dy)*(c_auxY-Dy) +
(c_auxZ-Dz)*(c_auxZ-Dz)));
Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((c_auxX-Ax)*(c_auxX-Ax) +
(c_auxY-(Ay+shift))*(c_auxY-(Ay+shift)) + (c_auxZ-Az)*(c_auxZ-Az)),
Sqrt((c_auxX-Dx_r)*(c_auxX-Dx_r) + (c_auxY-Dy_r)*(c_auxY-Dy_r) +
(c_auxZ-Dz_r)*(c_auxZ-Dz_r)));
Printf("Value test:: P(%g,%g,%g) -- C(%g,%g,%g) -- %g -- %g", P_cX, P_cY, P_cZ,
c_auxX, c_auxY, c_auxZ, Sqrt((P_cX-Ax)*(P_cX-Ax) +
(P_cY-(Ay+shift))*(P_cY-(Ay+shift)) + (P_cZ-Az)*(P_cZ-Az)),
Sqrt((P_cX-Dx_r)*(P_cX-Dx_r) + (P_cY-Dy_r)*(P_cY-Dy_r) +
(P_cZ-Dz_r)*(P_cZ-Dz_r)));

Point(13) = {c_auxX,c_auxY,c_auxZ,lc};


delta_A = Sqrt( ((Ax-c_auxX)*(Ax-c_auxX)) + ((AAy-c_auxY)*(AAy-c_auxY)) +
((Az-c_auxZ)*(Az-c_auxZ)) );
delta_B = Sqrt( ((Dx_r-c_auxX)*(Dx_r-c_auxX)) + ((Dy_r-c_auxY)*(Dy_r-c_auxY)) +
((Dz_r-c_auxZ)*(Dz_r-c_auxZ)) );
Printf("Ponto 13:: %g,%g,%g -- delta:%g,%g", c_auxX, c_auxY, c_auxZ, delta_A,
delta_B);


teste = Sqrt((Ax-c_auxX)*(Ax-c_auxX) + ((Ay+shift)-c_auxY)*((Ay+shift)-c_auxY)
+ (Az-c_auxZ)*(Az-c_auxZ) );
teste1 = Sqrt((Dx_r-c_auxX)*(Dx_r-c_auxX) + ((Dy_r)-c_auxY)*((Dy_r)-c_auxY) +
(Dz_r-c_auxZ)*(Dz_r-c_auxZ) );
teste2 = -Tan(Pi*0.5-a_right)*c_auxX - c_auxZ + Az;
Printf("teste dir:: %g <-> %g -- isplane:%f -- lambda:%g",teste, teste1,
teste2, lambda);

Circle(1) = {2,12,4};
Circle(2) = {6,10,5};
Circle(3) = {6,8,3};
Circle(4) = {2,13,5};
Circle(5) = {7,11,4};
```

```
Circle(6) = {7,8,3};
Circle(7) = {2,1,3};
Circle(8) = {2,1,6};
Circle(9) = {2,1,7};
Line(10)={2,9};
Line(11)={9,4};
Line(12)={9,7};
Line(13)={9,6};
Line(14)={9,5};
Line(15)={9,3};


Line Loop(1) = {8,3,-7};
Ruled Surface(1) = {1};
Line Loop(2) = {8,2,-4};
Ruled Surface(2) = {2};
Line Loop(3) = {9,6,-7};
Ruled Surface(3) = {3};
Line Loop(4) = {9,5,-1};
Ruled Surface(4) = {4};
Line Loop(5) = {10,14,-4};
Ruled Surface(5) = {5};
Line Loop(6) = {10,11,-1};
Ruled Surface(6) = {6};
Line Loop(7) = {13,3,-15};
Ruled Surface(7) = {7};
Line Loop(8) = {13,2,-14};
Ruled Surface(8) = {8};
Line Loop(9) = {12,6,-15};
Ruled Surface(9) = {9};
Line Loop(10) = {12,5,-11};
Ruled Surface(10) = {10};
Surface Loop(1) = {1,2,3,4,5,6,7,8,9,10};
Volume(1) = {1};


// coding
// code 1 : fixe vertices
// code 2 ; Chassignac
// code 3 : enregy surface
// code 4 : suturation left side
// code 5 : suturation right side
// code 6 : suturation bottom side
// code 7 : double suturation Left side
// code 8 : double suturation Right side
// code 9 : table match


Physical Surface(4) = {6};
```

```
Physical Surface(4+MINUS_ONE_DIM) = {6};
Physical Surface(4+MINUS_TWO_DIM) = {6};

Physical Surface(5) = {5};
Physical Surface(5+MINUS_ONE_DIM) = {5};
Physical Surface(5+MINUS_TWO_DIM) = {5};

Physical Line(7) = {11};
Physical Line(7+MINUS_ONE_DIM) = {11};

Physical Line(8) = {14};
Physical Line(8+MINUS_ONE_DIM) = {14};

Physical Line(1) = {15};
Physical Line(1+MINUS_ONE_DIM) = {15};

Physical Line(0) = {10};
Physical Line(0+MINUS_ONE_DIM) = {10};

Physical Line(9) = {1};
Physical Line(9+MINUS_ONE_DIM) = {1};

Physical Line(10) = {4};
Physical Line(10+MINUS_ONE_DIM) = {4};

Physical Surface(6) = {7,8,9,10};
Physical Surface(6+MINUS_ONE_DIM) = {7,8,9,10};
Physical Surface(6+MINUS_TWO_DIM) = {7,8,9,10};

Physical Surface(3) = {1,2,3,4};

Physical Point(1) = {9,3};

Physical Point(0) = {2};

Physical Volume(1)= {1};
```

Listing A.1: GMSH breast geometry script template