CONSTRUCTING A 3D MODEL BY USING 2D PARAMETER INPUTS

by

Umut Naderi

Engineering Project Report

Yeditepe University

Faculty of Engineering and Architecture

Department of Computer Engineering

2018

CONSTRUCTING A 3D MODEL BY USING 2D PARAMETER INPUTS

APPROVED BY:

Assist. Prof. Dr. Dionysis Goularas.....
(Supervisor)

Prof. Dr. Sezer Gören Uğurdağ.....

Assist. Prof. Dr. Esin Onbaşıoğlu.....

DATE OF APPROVAL: 14/01/2019

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Assist. Prof. Dr. Dionysis Goularas for his guidance, support, knowledge and patience during all my project phases.

Secondly I would like to thank Assist. Prof. Dr. Fethi Okyar for sharing his knowledge and support with me during my project.

I would also like to thank Research Assistant Yusuf Can Semerci for sharing his software knowledge and support during my project.

I would also like to thank M.Sc.Student Oğulcan Güldeniz for helping and guiding me during the earlier stages of my project.

ABSTRACT

CONSTRUCTING A 3D MODEL BY USING 2D PARAMETER INPUTS

The main goal is simulating the construction of the lumbar vertebrae in a 3D model. Every distinct individual has different pattern of spine form in terms of lumbar vertebral formation. In our project we have five different shapes, each representing a single vertebra in a lumbar vertebral column consisting five vertebrae. Given two different views from two different coordinate planes, these being frontal view and lateral view, which in our case we take the right view, user is expected to provide the coordinates of the specific vertices in a given sequence of guiding images. Having stored four different x and y coordinate values of vertices for each vertebrae in the first view, user is expected to the the same process for the second view.

In the final process, by the data obtained by the selections of the user, height, width, length and rotation is calculated by certain formulas. Placement of each vertebrae is determined by the values obtained and each are created by a Python macro, written into a ".py" file to be executed by FreeCAD program, providing us the 3D environment to simulate the lumbar vertebrae formation. Python macro consists of calls to preset 3D renders of vertebrae and their formation. Those commands are modified with the coordinate information provided by the user in the previous steps. Five vertebrae shapes are created according to those information and placed into the 3D environment.

ÖZET

İKİ BOYUTLU GÖRSELLERDEN PARAMETRELER İLE ÜÇ BOYUTLU MODEL YAPILANDIRMA

Projedeki ana hedef, lombar bölgede yer alan omurga parçalarını üç boyutlu bir model ile yapılandırarak simule etmek. Her bireyin omurga ve omur dizilimi ve düzeni birbirinden farklılık gösterebiliyor. Özellikle hastalıklı bireylerde bu farklılık daha göze çarpıcı bir şekilde görülebiliyor. Projemizde beş farklı omuru temsil eden şeklimiz mevcut. Bu şekillerin her biri omurganın lombar düzenindeki bir omuru temsil ediyor. Kullanıcıya önden ve yandan olmak üzere iki farklı açıdan, iki farklı görsel sunuluyor ve kendisinden istenilen noktaları seçmesi bekleniyor. Seçilen bu noktaların üç boyutlu uzayda denk geldiği noktalar saklanıyor. Her dört köşe için beş ayrı omurda bu işlem tekrarlanıyor. Daha sonra, diğer açıya sahip görsel için bu işlem tekrar olarak yapılıyor işlem tamamlanıyor.

En son işlem olarak ise kullanıcının seçtiği noktalardan elde edilen veriler ile oluşturulacak olan üç boyutlu modelin eni, boyu, kalınlığı ve dönme açısı hesaplanıyor. Üç boyutlu düzlemde bu şekillerin duracağı noktalar yine bu veriler ile hesaplanarak şekil düzleme yerleştiriliyor. Bu işlemlerin her biri seçim işlemi bittiğinde bir Python makrosu olarak ".py" formatındaki bir dosyaya kaydediliyor. Daha sonra da bu dosya FreeCAD programında çalıştırılarak üç boyutlu uzayda modelimiz oluşturuluyor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

FEA	Finite Element Analysis
OpenCV	Open Source Computer Vision Library
GUI	Graphical User Interface
CAD	Computer Aided Design
$\tan^{-1}(x)$	arctan(x) function
$\lvert x \rvert$	Absolute value of x
.py	Python file format
.cpp	C++ file format

1. INTRODUCTION

One of the most popular methods of studying the spine is transferring it to a three dimensional environment. This 3D environment provides us some benefits for analyzing and altering certain aspects of the objects and models. Having this kind of potential in our hands, we can make it work to our needs. It gives us the possibility of modeling complex geometries of spine. By creating this model, we will have the chance to simulate the construction and the movement by separately placing every vertebra in a lumber vertebral column.

Figure 1.1 3D Model of a Spine

Vertebra arrangement may have differences on different bodies. Not every individual has the same exact spine formation. Although they may have a similar type, some minor differences will conceive a completely different model than the others. As a matter of fact, these differences should be taken into account when the data are collected.

In this project, the lumbar vertebrae information will be given from two images, which are representing two dimensions of view. These views will be from frontal dimension and lateral dimension, which in our case is the right view to be used.

Table 1.1 Table for the selection guide

Vertebra Count	Vertices	View Levels
5	4	2

Lumbar vertebrae is represented in a basic way in this project. The base body shape of the vertebra is used in our 3D model. Using the possibilities that are provided us by a free 3D program(FreeCAD), this base shape is created by parametric python macro codes. These macro codes are built in a way that can be adjustable during the later stages according to the data provided by the user. Macro codes that that constructs initial lumbar vertebrae is connected with the parametric data that is taken from the user, which leads to give it its final form, presented to the user as an output. Using this kind of model, it grants us to modify its all kind of its geometrical aspects as desired. Height, width, length, three different rotation axises and rotation amount in degrees can be changed as provided.

Our model consists of five vertebrae which forms a lumbar vertabra column. User will be selecting four vertices of each vertebra. These vertices will help us collect the needed information to construct the shapes of each vertebra in a 3D environment to form a lumbar vertebrae model. User should do this process for two different dimensions, as previously mentioned, for frontal and lateral dimensions. These different dimension will help us calculate the rotational data as well as the measurement of the each shape in the model.

2. BACKGROUND

2.1. Parametric Modeling

Parametric modeling is modeling the objects with real-life behaviors and attributes by using modeling software. The important thing in parametric modeling is to simulate the characteristics of the components in the model. The interaction between the objects in a model has a very important role in parametric modeling. Parametric models are created by using mathematical calculations. Data given into these equations lead into the outcome which is presented in the end as a result.

Parametric methods are very useful for simulating dependent circumstances. Behavior of a substance can vary according to the situation it is in. In these kinds of situations, in order to obtain the outcome of that exact situation and observe the consequences, parametric modeling is the recipe that is approached.

Parametric modeling is a field that has a great use in the recent years. It provides a great range of ways to observe the model when necessary. Parts of the model can be separated from the whole and can be viewed and treated as an individual part in situations. It also grants a great opportunity visually in certain circumstances. Depth of the whole object can be seen as well as the parts that forms the whole model.

One of the most popular programs for parametric modeling is Solidworks. It is known for modeling solid objects and design products. It is also known for its parametric modeling use. An example of parametric modeling in Solidworks can be seen in **Figure 2.1**.

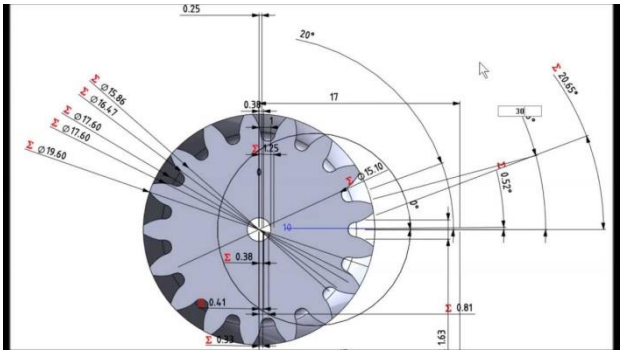


Figure 2.1 Parametric modeling of a gear in Solidworks

2.2. Related Works

There have been some researches made in past years for similar purposes.

One research has been made by a group involving Nicola Cappetti, Alessandro Nadeo, Arcangelo Pellegrino, Giovanni Francesco Solitro and Francesco Naddeo in January 2010. It is directly related with parametric modeling of lumbar vertebrae. They analyzed the morphology of vertebrae in their research. A single vertebra's structure has been identified separately in the research. They described the body as form of cylinder with a concave shaped side.

Construction of vertebra was the next subject in the research. They analyzed the size, weight and the proportions of the body. The entire model was built in a three-dimensional space, in which the plane x-y coincides with the sagittal plane, and the axes origin is at the bottom extremity of the body superior face [4]. Their model was organized both vertically and horizontally for them to manage the high number of graphical entities. They also grouped the elements according to their characteristics such as structure, edge and surface. Furthermore elements of the vertebra model were dealt separately. Five different elements were given shapes to form a whole vertebra model.

Similar to the grouping of the elements according to their characteristics, they have grouped the parametric elements in a similar manner. Structure group handled the parametric data involving dimensions, position and orientation while the edge group handled the tangency of the border curves [4].

After these processes they have made, they have successfully obtained a well structured lumbar vertebra model by the parameters they have given.

Another study was made by Mohammad Haghpanahi and Mehrdad Javadi in 2011. Differently, they made their research on whole lower cervical spine. By reviewing similar studies and previous parametric models, and by studying the geometry of the vertebra through CT-Scan images, 29 geometrical parameters were selected for defining each vertebra [3]. For assembling vertebras, two parameters for each motion segment have been chosen to identify the position of the vertebras in relation to each other [2]. One parameter identifies the distance between two opposite surfaces of two vertebras. The other parameter is the angle between the mentioned surfaces [2]. Other parameters which are given are for defining the thickness and the ends of each vertebra. They used the software called "CATIA V5" during their research. "Part Design" module was used during the modeling. They made this process for all 5 vertebrae. After the implementation is done, model of these 5 vertebrae was created.

Different from what we are trying to create in our research, they involved the modeling and implementation of the soft tissues of the vertebrae. Our research has a more basic purpose so for this time we did not want to involve the soft tissue forms in our project.

They made calculations on the rotating angles, calculating the rotation with the following formula;

$$\cos\theta = \frac{U \cdot U'}{|U|\,|U'|} \tag{2.1}$$

where U is a vector that connects two points from the upper vertebra before deformation, U′ is a vector that connects the same two points of the upper vertebra after deformation and Θ is the angle between U and U ′ , which indicates the rotation angle of the upper vertebra [2].

3. ANALYSIS AND DESIGN

3.1. The Lumbar Vertebral Column

The lumbar vertebral column consists of five separate vertebrae and they are the largest segments of the vertebral column in human body. We can separate their identities by naming them first, second, third, fourth and fifth vertebra from above to downwards. Numeration of each separate vertebra in the formation can be seen in figure below.

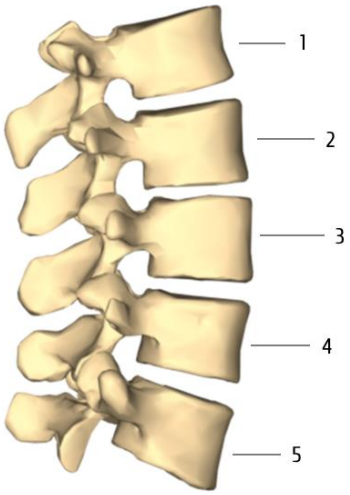


Figure 3.1 A Lumbar Vertebral Column from Lateral View

3.2. A Typical Vertebra

3.2.1. The Vertebral Body

The vertebral body is the main component of spine, a significant part in load-bearing of spine [3]. Each vertebra is made up cancellous bone core and cortical bone shell [5]. The lumbar vertebrae are roughly cylindrical with a lateral diameter of 40-50 mm and sagittal diameter of 30-35 mm [1]. The body shape of a single vertebra can be seen in the figure.

Figure 3.2 Frontal view of a single vertebra

3.2.2. The Geometry of a Vertebra

The actual geometry of a vertebra involves the body, the facet and the lamina. These three parts are the main points that form the vertebra. These can vary in depth, height and length in different kinds of situations. Five vertebrae in the lumbar vertebral formation are connected together by the facets. Each facet makes a bond with another to form the structure.

In order to simulate it we decided to draw a shape that acts as a representative in the process. 3D model has its height, width and length values which can be changed and differ from each other. It can also vary by the angle it has been rotated. We have the ability to deform it in two dimensions as desired during the process.

The main focus is to create a model that can simulate the look and feel of a real vertebra in a simple manner. For this particular intent, we had to decide what kind of structure is going to be used to represent a single vertebra in the 3D space.

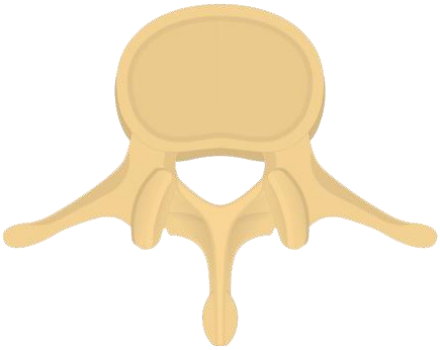


Figure 3.3 A single vertebra's view from top

3.3. Our Approach

Our main aim is to acquire a proper 3D model of lumbar vertebrae assembled by a data set, initially determined by the user input. Given the images of lumbar vertebrae, it consists of five separate elements in a structure of formation. In order for us to represent it in a 3D environment we need to create a 3D model which will be its representative in this exact environment.

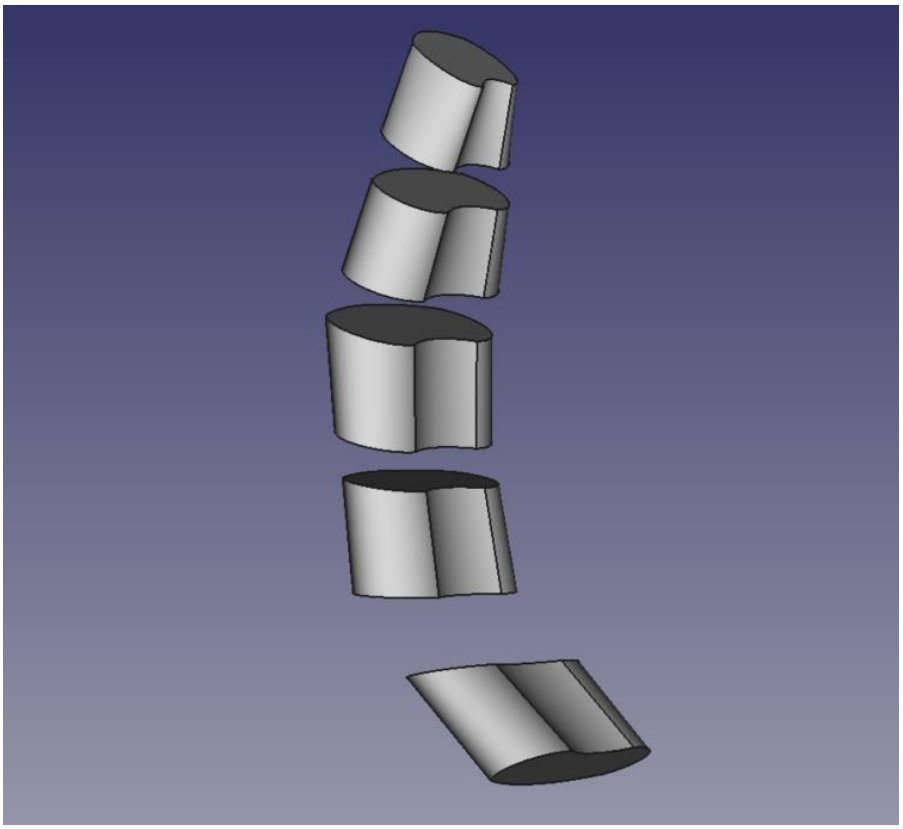


Figure 3.4 Parametric lumbar vertebrae in 3D environment

Initial step of parametric modeling is to learn and study a basic 3D model which can benefit on our behalf and have the knowledge to alter all its data by numerical data in the end. After understanding the basic form of parametric modeling, we need to create our own model and gather its parameters those which are modified by the user input. User should have the images of a real vertebra and present the data to the software. User input is always prone to some flaws but in our case we can neglect that kind of error rate.

Accurate data of information is necessary to come against a reliable result. Therefore data of vertebra positioning and placement should be directly taken from the user. Distortions and changes in these placements may occur. For the sake of accurate information user should be given a 2D image involving the formation of a lumbar vertebrae. For situations such as rotation, we need at least two view axes in order to apprehend the rotation of an element. In a 3D space, rotation may occur on each 3 axes.

Algorithm 1. Obtaining and collecting the data by the input.

```
    Function UpdateImageToPoint
    IF view is front
    THEN displayImage
         drawCircle to point x and y, color yellow
         point <- position
         spine ->
vectorVertebrae.at(vertexCount/4).vectorFrontal.at(vertexCount%4).setx(pos.x)
         spine ->
vectorVertebrae.at(vertexCount/4).vectorFrontal.at(vertexCount%4).setx(pos.y)
         drawCircle to point x and y, color yellow
         updateImage
    ELSE IF view is not front
    THEN displayImage
         drawCircle to point x and y, color yellow
         point <- position
         spine ->
vectorVertebrae.at(vertexCount/4).vectorLateral.at(vertexCount%4).setx(pos.x)
         spine ->
vectorVertebrae.at(vertexCount/4).vectorLateral.at(vertexCount%4).setx(pos.y)
         drawCircle to point x and y, color yellow
         updateImage
```

Collecting the data in a correct way is also the key to acquire an accurate outcome. Each element's data should be kept in separate in order to avoid any mistakes in the later stages. We should keep the data for height, width and length of an element, as well as their rotational data. In order to calculate the height data, we need to obtain the exact coordinates of the top level and the bottom level of an element. For this purpose, we need to acquire data relevant to the top level and the bottom level of the element.

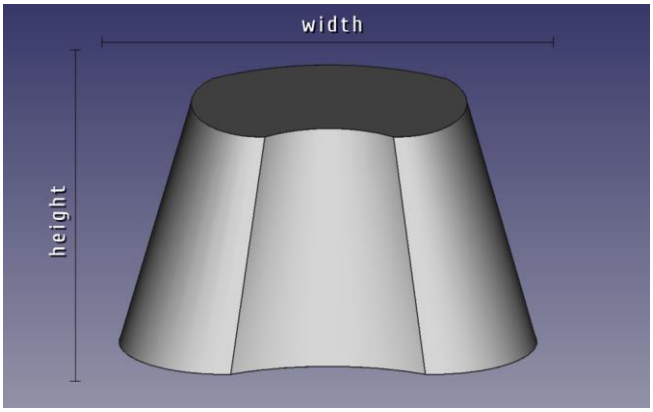


Figure 3.5 Image showing how height and width are addressed on model

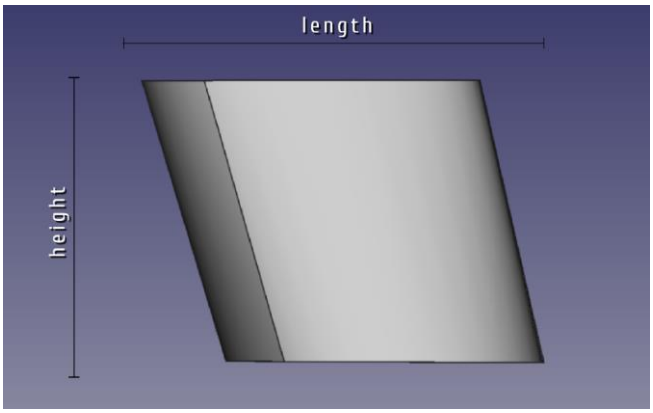


Figure 3.6 Image showing the height and length are addressed on model

Similarly, for width data we must have the knowledge of the data of the left hand side and the right hand side of the element. Obtaining these data from a 2D image will guide us for forming up an accurate 3D parametric model of what we are looking for. Fortunately, these exact values will also help us calculate the rotation of an element. We may know how a model element rotated in a certain axis by these values.

Finally having all necessary data for 5 different vertebrae in the formation, we will insert these data values into the parametric model and expect it to present us the result model.

3.4. General Structure

A reliable and well designed structure is a must in the project. Therefore we did pay attention to how the structure of our model is built up. The structure is formed in a

hierarchical order from top to bottom. Initial form of the structure can be seen in **Figure 3.7** below.

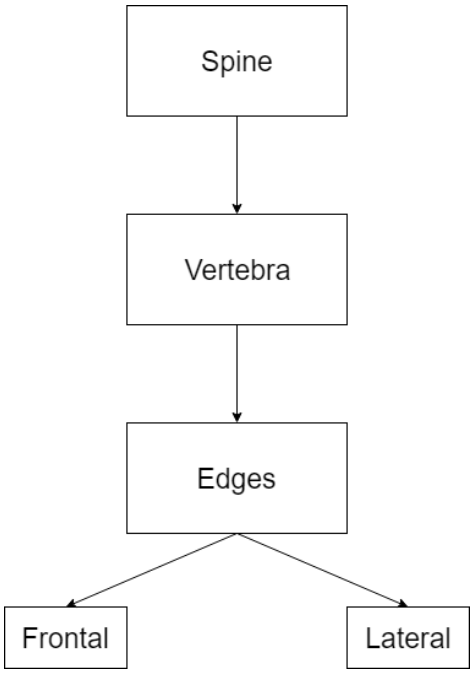


Figure 3.7 The structure of our model

Our main intent with creating this kind of structure is to access all necessary parts in a simple and clean way, therefore to avoid any possible redundant complexities. According to this we followed a simple path. We handled every structure element in a different class during the implementation. Spine class, being on the top of the structure has a subclass called Vertebra. Every Vertebra object has 4 edges for two different sides. Data is being collected for both frontal view and lateral view. It is important that these data sets are kept separate. Beyond these main sets, we also collect and calculate the rotation data and the size aspects.

3.5. Proposed Design

Analyzed features and goals should be put in a planned and designed structure in order to start shaping up our project.

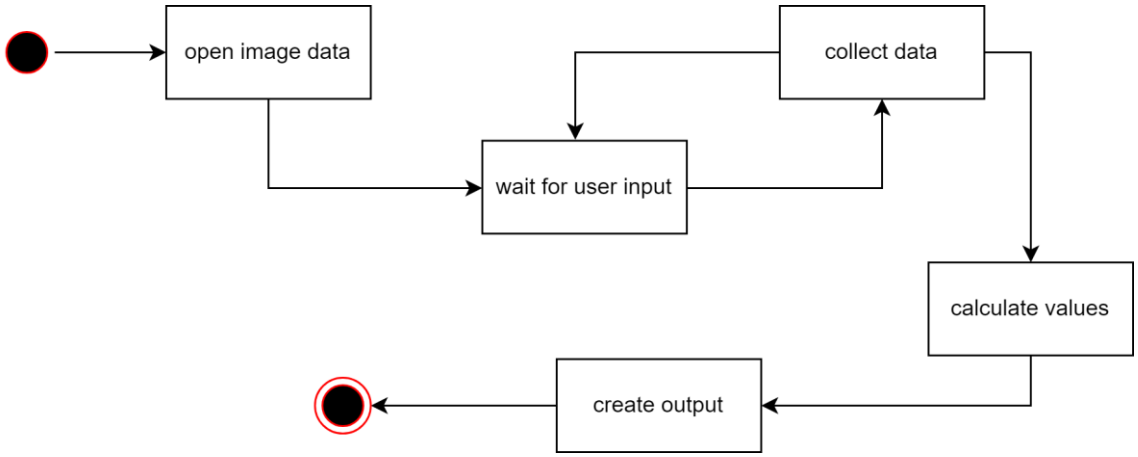


Figure 3.8 Actıvıty Diagram of the project

User will be giving an image data into the program. According to this image, program will wait for the user's input of click into the frame to decide where the vertex is. Vertex value will be collected inside an array, preferable a vector which has its size not preset. Value of the mouse coordinates will be collected at each step. Each value will be used in set of calculation methods to determine de scale, position, rotation of each element which will be presented to user as an output.

3.6. Calculations

We collect the values of mouse coordinates with the use of our software. Mouse coordinates of each vertebra edge are collected as a data set. These do not have sole use. However, by using some calculations and implementing them into methods we can figure out many aspects of the model such as scale, positioning and rotation angle.

We calculate the size of a model by this following formula,

$$x = \sqrt{|v1 - v2|^2 + |y1 - y2|^2} \tag{3.1}$$

where v1 and v2 are two top edge coordinates by the x-axis and y1 and y2 are the top edge coordinates of those same edges by the z-axis. We do this in order to cancel out the rotation if any to determine the width or length of a single vertebra.

Also we calculate the rotation angle with this following formula,

$$\arctan\theta = \frac{v2-v1}{y2-y1} \times 180/\pi \quad (3.2)$$

where v1 and v2 are the edge coordinates of the two adjacent edges in z-axis and y1 and y2 are two coordinates of the same edges in x axis. Multiplying the division with 180 and dividing it by π will give us the angle θ as the result.

4. IMPLEMENTATION

4.1. Software Used

4.1.1. OpenCV

OpenCV (Open Source Computer Vision Library) is a library that can be used on multiple platforms for multiple purposes. It was built to provide a common infrastructure for computer vision applications. It makes it easy to utilize and modify the code. It contains the methods and objects for the project.

4.1.2. Qt

Qt is a cross-platform application development framework for desktop, embedded and mobile. It is a framework written in C++. It has its own libraries, but it also supports libraries imported into its interface. In the project, CMake is used to import OpenCV and its components into Qt platform in order to maintain the graphical aspects of the application during the execution.

4.1.3. FreeCAD

FreeCAD is an open source parametric 3D modeling application, made primarily for designing real life objects. Parametric modeling describes a particular type of modeling where the shapes of the 3D objects that are designed are controlled by parameters. These parameters are part of object and continue to be modifiable at any time, even after the object has been created. It also benefits from its support of Python environment. Every object can be controlled and altered via Python scripts and macros. Console window in the program enables any user to enter a macro or a supported python script for particular uses such as changing the object's placement, rotating object in any three dimensions at any angle given to it.

4.2. Development

We used many different methods in order to achieve what we desired. One of the many important methods is to insert an obtained coordinate value into a data set.

Also by the using this exact method, we can update the image and put an indicator, which in our case is a yellow circle, to the point where user selects. Circle will be displayed as soon as the user makes the click action.

Our data array, in our case vector, starts from 0 therefore we use modulus operation to decide which data is collected in which part in an incremented action.

4.3. User Interface

Our software has a basic user interface. We avoided implementing any redundant feature in order to prevent any possible misleads when used.

Firstly the user is asked to select an image file from the system explorer by using the ‘File’ tab at the top shelf using our interface. It will present the user two different choices which the user will decide to select whether the front view of a lumbar vertebrae image or the lateral view of a lumbar vertebrae image. It supports image files such as .png and .jpg.

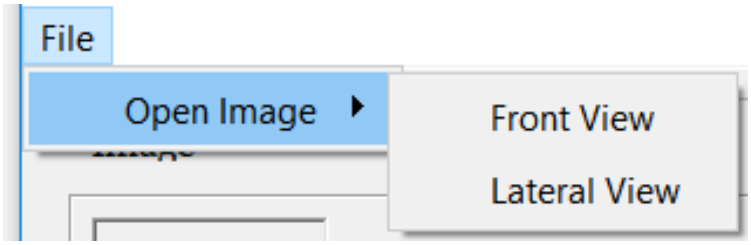


Figure 4.1 File tab from the user interface

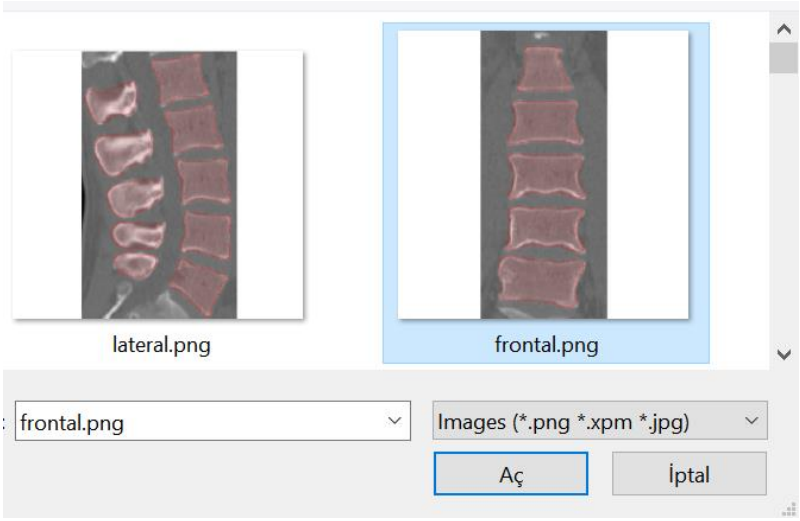


Figure 4.2 File selection screen of our software

User is given a preview of a selection guide on the right hand side of the screen after selecting the desired image file from the file selector. This guide image will help the user and lead on which point to be selected for the each stage of selection. Lumbar vertebrae has 5 vertebra and each will have 4 different points to be selected for their 4 edge points, which we call as vertices.

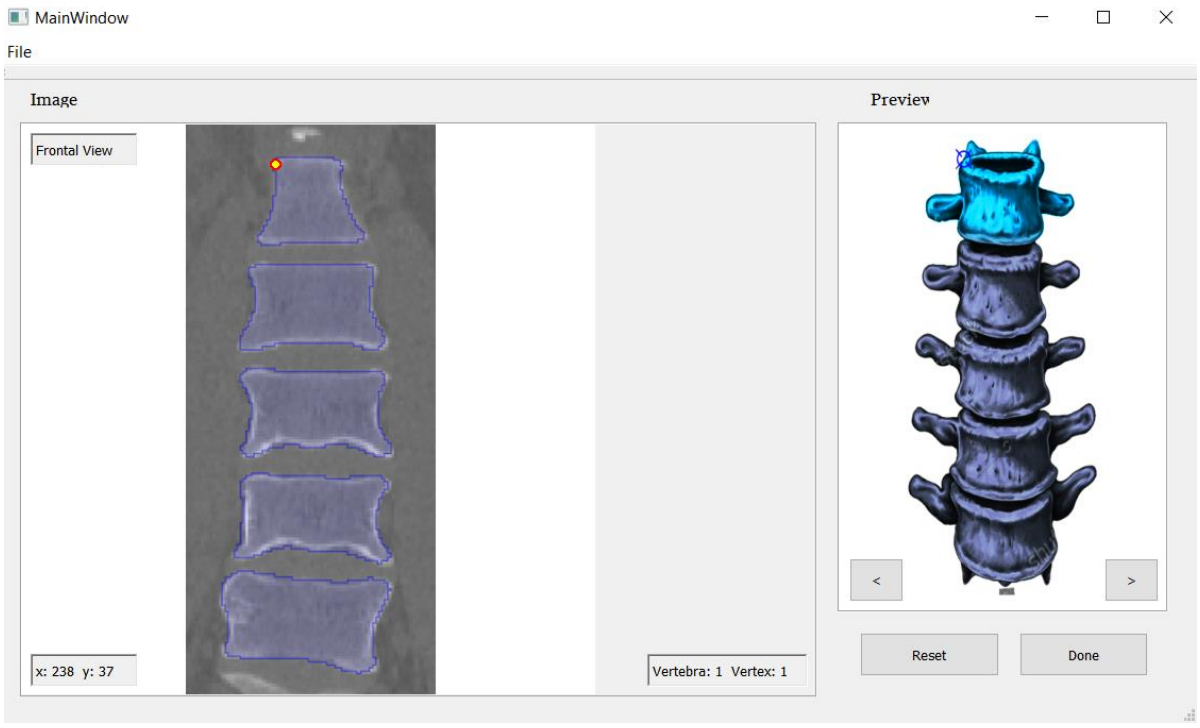


Figure 4.3 Overview of our software for frontal selection

Each selection by the left mouse button into the frame on the left hand side will produce a feedback of the point clicked to the user by a yellow circle mark. By this, user will have the information of where the click has been made. In addition to this, user is provided values and info of the progress made. On the left bottom of the left frame, user is provided the exact coordinates where the mouse is located at that moment. Also on the right bottom, user can see the information of which vertebra is next on the click set.

Figure 4.4 a. Mouse coordinates on frame **b.** Vertebra information on frame

4.4. 3D Model

4.4.1. Initial Estimation

Our 3D model has been designed in a basic shape in FreeCAD. We built the base structure of a vertebra in 3D environment. For these shapes to be parametric, they need to rely on values and have a form according to those given values. FreeCAD supports Python macros on its console.

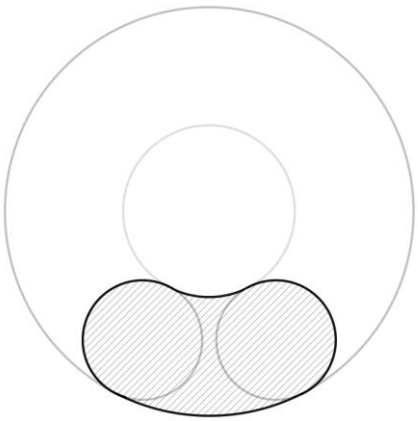


Figure 4.5 Sketch geometry of the vertebra base

Shapes can be created, edited, renamed by this console as well as their values can be edited when desired. Having analyzed the shape and geometry of a vertebra, we created a solid shape of the base of a vertebra for the use.

We used the sketching tools in FreeCAD to draw the base of a vertebra. It was obtained by having two different radius of circles of the same center and two symmetrical smaller radius circles placed between them. Cutting out the shape accordingly, we came up with the shape that will represent the vertebra in our model. It can be seen in the **Figure 4.5** above. This sketch was placed placed to the top and bottom and between these were filled as a solid afterwards. This progress is called “loft” in modeling literature. Each vertebra model was numbered accordingly. Finally they are scaled, rotated and positioned by a python script accordingly into the 3D environment.

4.4.2. Modeling in FreeCAD

Firstly we created a sketch object inside the 3D environment in order to begin our process of modeling. As we initially planned, we draw two circles which share the same center of radius. Inside circle has a radius of 15 mm and outside circle has a radius of 30 mm. Below you can see the python macro of this process and the outcome of it as a figure.

```
App.newDocument("Project")
App.setActiveDocument("Project")

from FreeCAD import Base
import Part,PartGui,Draft

sketch01 = App.activeDocument().addObject('Sketcher::SketchObject','Sketch01')
sketch01.Placement = App.Placement(App.Vector(0.000000,0.000000,0.000000),App.Rotation(0.000000,0.000000,0.000000,1.000000))

sketch01.addGeometry(Part.Circle(App.Vector(0.000000,0.000000,0),App.Vector(0,0,1),30),False)
sketch01.addConstraint(Sketcher.Constraint('Coincident',0,3,-1,1))

sketch01.addGeometry(Part.Circle(App.Vector(0.000000,0.000000,0),App.Vector(0,0,1),15),False)
sketch01.addConstraint(Sketcher.Constraint('Coincident',1,3,-1,1))
```

Figure 4.6 Python macro of sketching two circles sharing the same central point.

After creating these two circles, we need to also create two symmetrical circles located in between these two circles we created earlier. Two symmetrical circles with radius of 8 mm would be sufficient for our process.

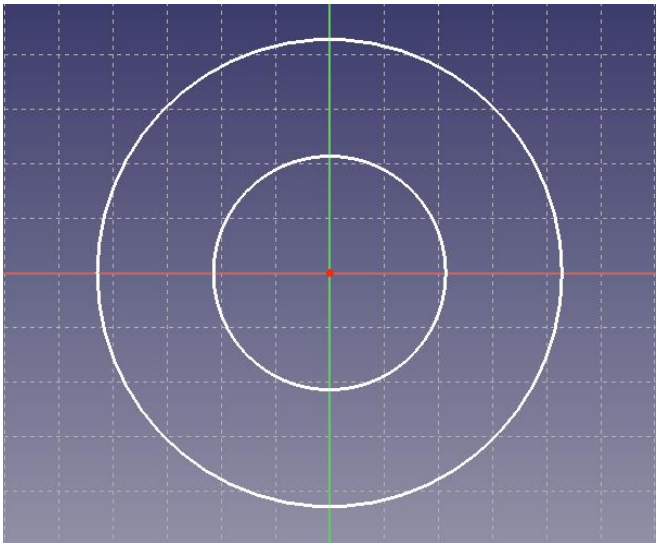


Figure 4.7 Sketching of two nested circles sharing the same central point

Below you can see the python macro of the placement and the sketching process of these two symmetrical circles. Also the result of this process can be seen in the figure below.

```
sketch02.addGeometry(Part.Circle(App.Vector(8,-21,0),App.Vector(0,0,1),8),False)
sketch02.addSymmetric([2],-2,0)
```

Figure 4.8 Python macro of sketching two symmetrical shaped circles.

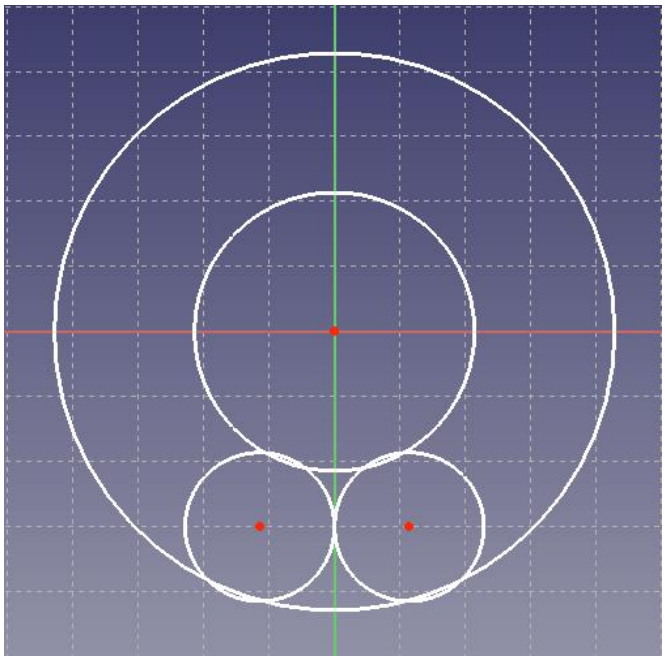


Figure 4.9 Form of the model after two symmetrical circles created.

In order to give our sketch its final form, we need to get rid of all the excess shapes and lines that we used to give it a shape. For this purpose, we use the trim action inside FreeCAD's sketch tools. Trim action is used for breaking and erasing the lines between any kind of joints. Any selected lines which are connected between two separate joints will be erased from our sketch afterwards. Here is how our python macro is shaped after the trimming process, which also produces our final sketch.

```
sketch01.trim(3,App.Vector(-4.671036,-13.775558,0))
sketch01.trim(2,App.Vector(4.397662,-13.864904,0))
sketch01.trim(3,App.Vector(-1.186513,-16.455961,0))
sketch01.trim(2,App.Vector(0.090503,-19.139975,0))
sketch01.trim(3,App.Vector(-11.270746,-28.304337,0))
sketch01.trim(2,App.Vector(10.701625,-28.702028,0))
sketch01.trim(0,App.Vector(-17.697254,-24.176645,0))
sketch01.trim(1,App.Vector(1.574561,15.313293,0))
sketch01.trim(1,App.Vector(0.207600,14.282619,0))
```

Figure 4.10 Python macro of the trimming process of excess lines.

After finishing the trimming process, our sketch model is shaped like in the figure below. We created the body of a vertebra using the sketch tools in FreeCAD, using macros.

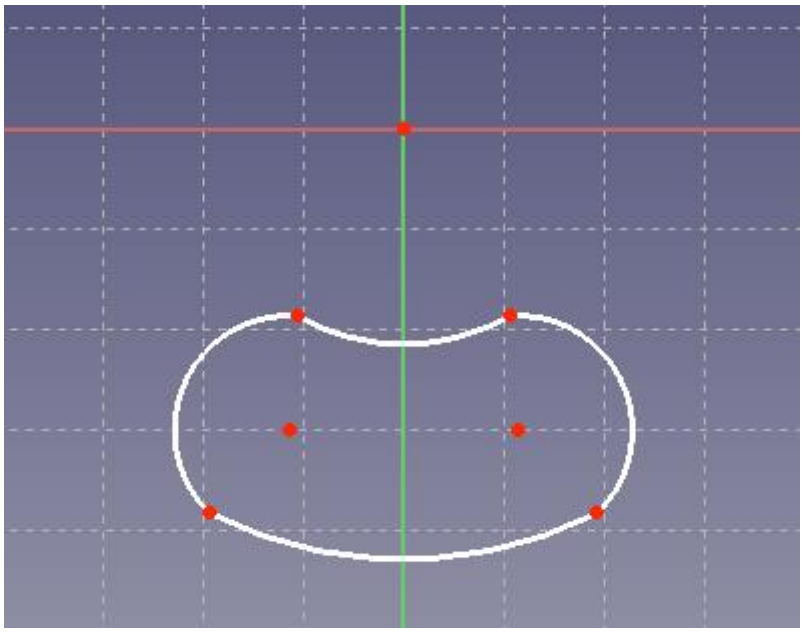


Figure 4.11 Final shape of the sketch after the trimming process.

Second step of building our model is to create a solid object by filling the area between two of the sketches we created earlier. This process is called “Loft” in design terms. In order to fill between two sketches, we need to use loft action inside FreeCAD’s Part Design section. Python macro of this process is stated below.

```
scale01 = Draft.scale([sketch01],delta=FreeCAD.Vector(2,2,2),center=FreeCAD.Vector(0,0,0),copy=False)
scale01.Label = 'Scale01'

scale02 = Draft.scale([sketch02],delta=FreeCAD.Vector(2,2,2),center=FreeCAD.Vector(0,0,0),copy=False)
scale02.Label = 'Scale02'

loft01 = App.getDocument('Project').addObject('Part::Loft','Loft01')
loft01.Sections=[scale01, scale02, ]
loft01.Solid=True
loft01.Ruled=False
loft01.Closed=False
```

Figure 4.12 Python macro of the loft process in order to create a solid object.

Final shape of our model is ready after the loft process. We filled the space between two sketches we created earlier. Final model can be seen in the figure below.

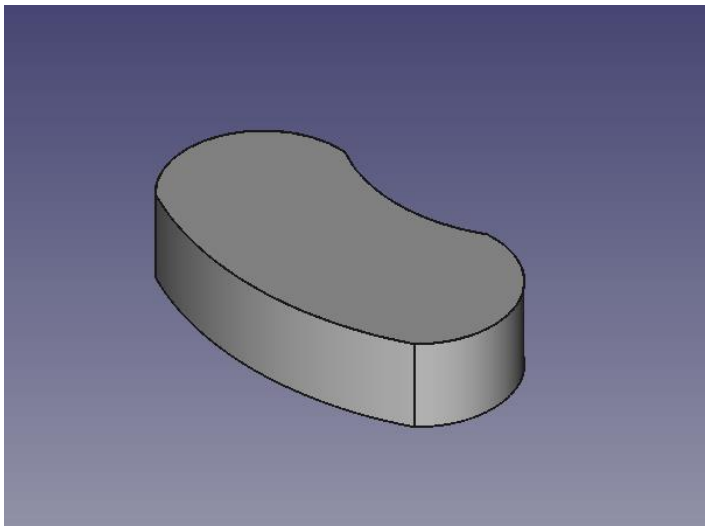
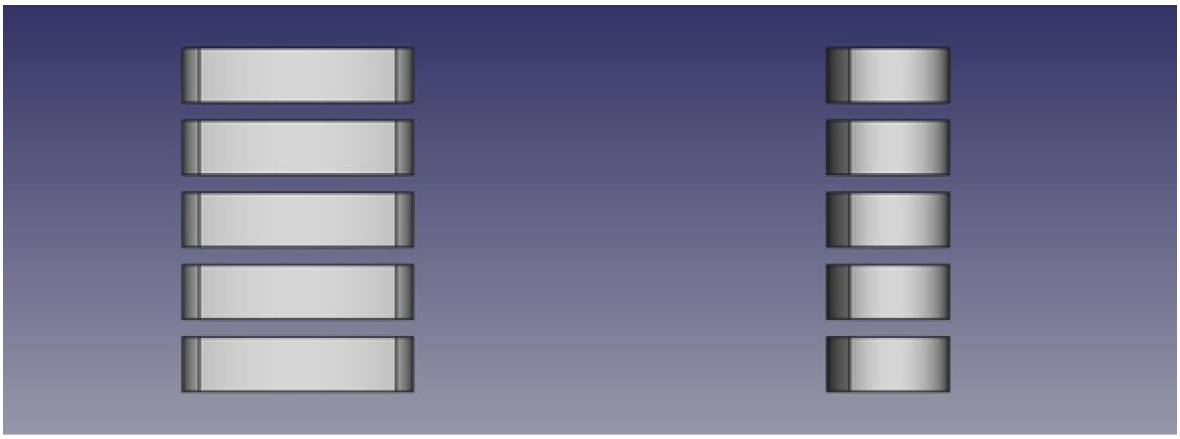


Figure 4.13 Final shape of our model after the loft process.

5.TESTS AND RESULTS

5.1. Test Case 1 – Straight Alignment

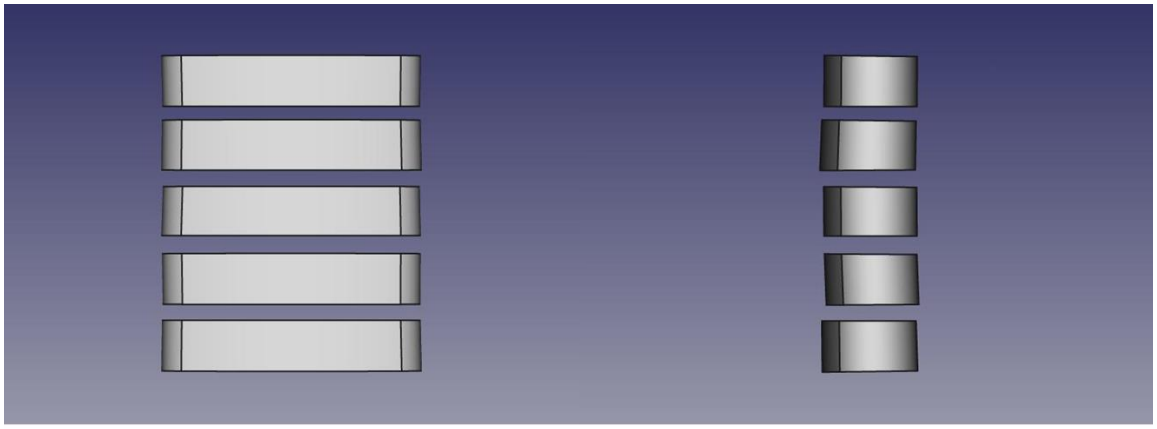
Initial formation of the first test model is aligned as in the figures below.



a b

Figure 5.1 a. Frontal view of the initial formation **b.** Side view of the initial formation.

Running our application by these two inputs given we conclude with a similar model with minimal errors. Due to error prone on mouse coordinates while selecting the corners there are small angle distortions. However, placement of the vertebral formation is pretty accurate. The result can be seen below;



a b

Figure 5.2 a. Frontal view of the result **b.** Side view of the result.

Table 5.1 Table for Test Case 1 – Straight Alignment

Vertebra Name	Initial Volume	Test Result	IV*/TV**	TR/TV***	Error %
#1	26586.945	64212319.658	0.1999	0.2004	0.0005
#2	26586.945	64636835.418	0.1999	0.2017	0.0018
#3	26586.945	62131917.909	0.1999	0.1939	0.0060
#4	26586.945	63685660.158	0.1999	0.1988	0.0011
#5	26586.945	65668078.707	0.1999	0.2049	0.0050
Total	132934.727	320334811.851	1.0000	1.0000	0.0144
Average	26586.945	64066962.370	0.2000	0.2000	0.0028

*IV: Initial Volume **TV: Total Volume ***TR: Test Result Volume

In this test case, there were some very minor differences between each vertebra volumes. However, the outcome was 100% accurate. This test gave us a 0.0028% of error rate which can be seen above. However, if we considered more fractions it would produce more error rates.

5.2. Test Case 2 – Displacement of Two Vertebrae

In this test case we tried to displace two vertebrae from their default locations by sliding them 10 mm to the right side looking from the front. In other words we shifted the Vertebra #1 and Vertebra #3 counting from the top positively in the x-axis. Initial placement of the alignment can be seen below in the figures.

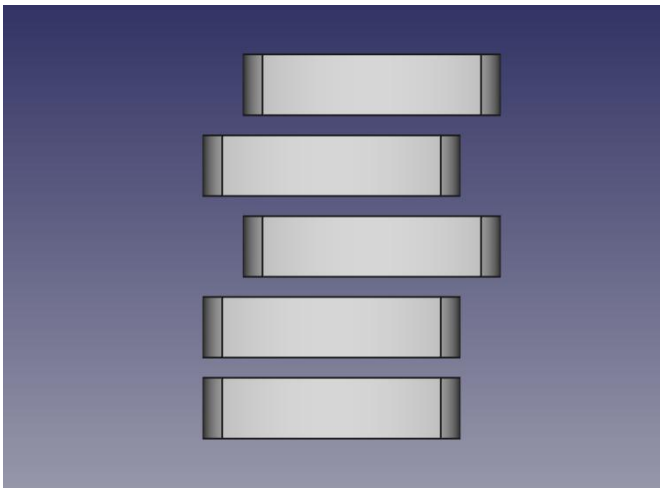


Figure 5.3 Initial alignment of the model for test case 2

Our shifting amount initially was 10 mm manually on the software. However since we enlarge the object, since mouse coordinates are obtained by the pixel amount on the screen, on bigger scales the affect of the same amount of difference is much smaller and less observable. 10 mm shift caused an approximate 48 mm shift on the enlarged scale as an output in this case. The result model can be seen below.

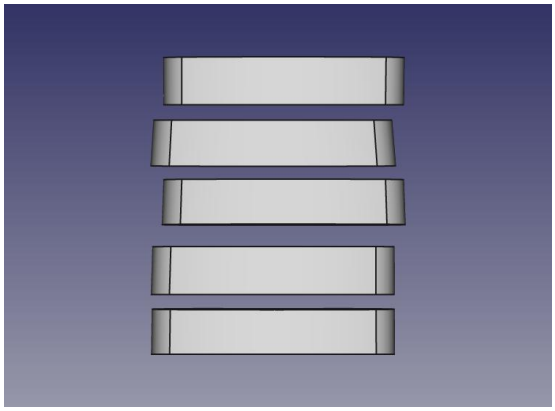


Figure 5.4 Result model for test case 2

Second test case gave an error rate of 0.01%. List of the volumes for this test case can be seen in the figure below. The accuracy of the volumes differ from the last test case, however the result is consistent like the previous one.

Table 5.2 Table for Test Case 2 – Displacement of Two Vertebrae

Vertebra Name	Initial Volume	Test Result	IV*/TV**	TR/TV***	Error %
#1	26586.945	59594159.914	0.1999	0.1894	0.0105
#2	26586.945	65358214.416	0.1999	0.2077	0.0078
#3	26586.945	61723816.255	0.1999	0.1962	0.0037
#4	26586.945	58761673.459	0.1999	0.1868	0.0131
#5	26586.945	69197325.556	0.1999	0.2199	0.0200
Total	132934.727	314635189.601	1.000	1.000	0.0551
Average	26586.945	62927037.920	0.1999	0.2000	0.0110

*IV: Initial Volume **TV: Total Volume ***TR: Test Result

5.3. Test Case 3 – Different Scales

In this test case, the the size of the model has been changed manually on different scales. From top to bottom, 1.60, 1.70, 1.70, 1.60 and 1.50 values were given into separate elements respectively. Initial view can be seen in the figure below.

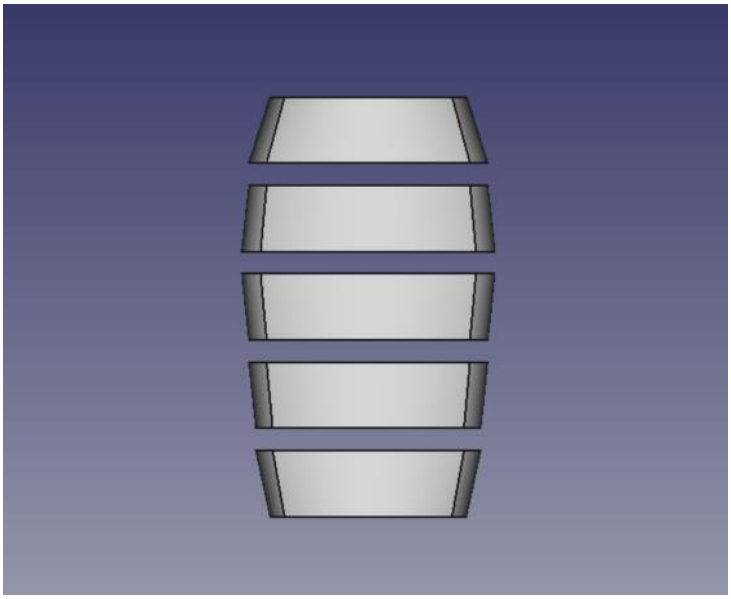


Figure 5.5 Frontal view of model for Test Case 3

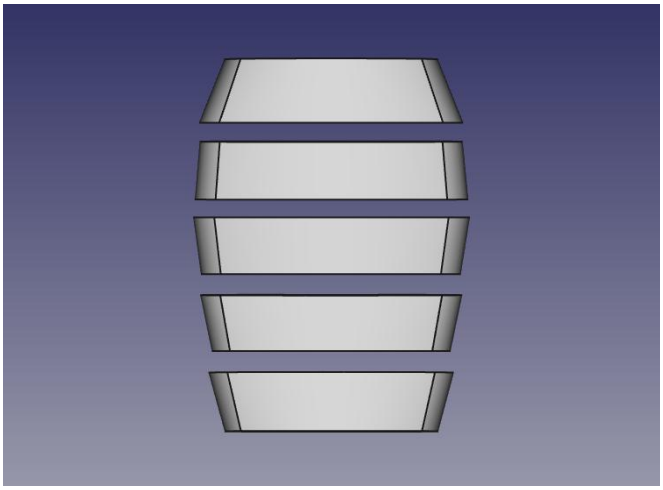


Figure 5.6 Front view of result model for Test Case 3

Table 5.3 Table for Test Case 3 – Different Scales

Vertebra Name	Initial Volume	Test Result	IV*/TV**	TR/TV***	Error %
#1	21269.554	55464554.829	0.2099	0.1894	0.0204
#2	23263.574	55102850.934	0.2085	0.2077	0.0008
#3	23263.574	55108892.470	0.2085	0.1962	0.0123
#4	21934.228	49803508.676	0.1884	0.1868	0.0016
#5	20604.881	48812399.096	0.1847	0.2199	0.0352
Total	110335.813	264292206.007	1.000	1.000	0.0703
Average	26586.945	62927037.920	0.1999	0.2000	0.0140

*IV: Initial Volume **TV: Total Volume ***TR: Test Result

The results were accurate after the test case. The model was similar to the initial parameters. It protected its predefined shape after the process. Second vertebra in the formation had the smallest error rate after the process with a ratio of 0.0008%. Biggest error rate came from the fifth vertebra with a ratio of 0.0352%. Average error rate was calculated as 0.0140%.

5.4. Test Case 4 – Scaled Elements

In this test case we made some scaling of elements as the initial condition. Initial alignment can be seen in the figure below.

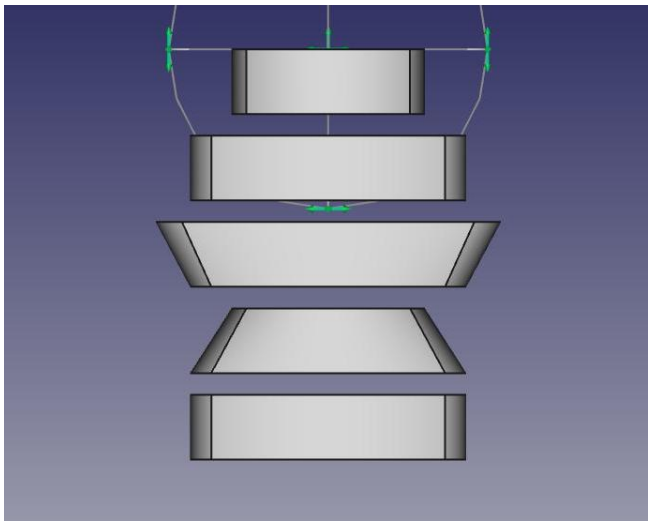


Figure 5.7 Initial alignment for Test Case 4

The result was accurately obtained by the program. Table below gives the information about this test case.

Table 5.4 Table for Test Case 4 – Scaled Elements

Vertebra Name	Initial Volume	Test Result	IV*/TV**	TR/TV***	Error %
#1	16586.45242	41273147.11200	0.13566	0.13746	0.00180
#2	26586.94557	64964431.52113	0.21745	0.21637	0.00107
#3	29898.29510	71262540.25725	0.21745	0.23735	0.00718
#4	22606.18384	53711669.41471	0.24453	0.17889	0.00599
#5	26586.94557	69026800.58438	0.18489	0.22990	0.01240
Total	110335.813	264292206.007	1.000	1.000	0.02844
Average	26586.945	62927037.920	0.2000	0.2000	0.00568

*IV: Initial Volume **TV: Total Volume ***TR: Test Result

5.5. Test Case 5 – Rotated Element

In this test case we rotated the first element in the formation to see the outcome. Also avoid any collusion between the vertebrae, we made the second element smaller than it should be for the case. We rotated the first element 30 degrees as the initial condition.

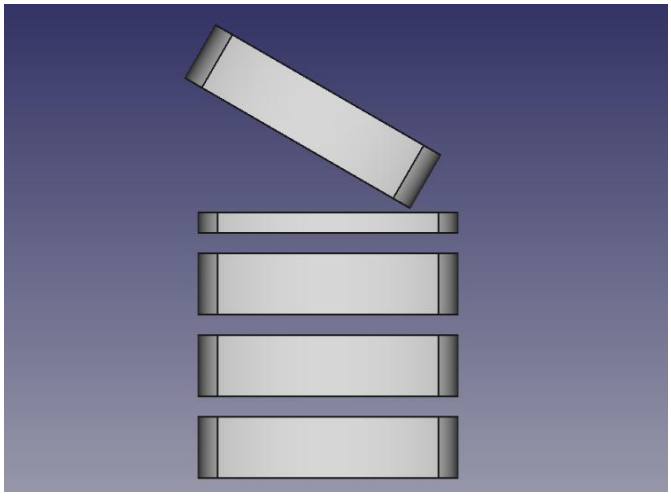


Figure 5.8 Initial alignment for Test Case 5

Table 5.5 Table for Test Case 5 – Rotated Element

Vertebra Name	Initial Volume	Test Result	IV*/TV**	TR/TV***	Error %
#1	26586.94557	37656657.80900	0.23077	0.15292	0.07785
#2	8862.31519	20800605.36592	0.07692	0.08447	0.00755
#3	26586.94557	59040019.01386	0.23077	0.23976	0.00899
#4	26586.94557	61981422.02898	0.23077	0.25170	0.02094
#5	26586.94557	66767755.57870	0.23077	0.27114	0.04037
Total	115210.09751	246246459.79649	1.000	1.000	0.15570
Average	23042.0195	49249291,9593	0.2000	0.2000	0.03114

*IV: Initial Volume **TV: Total Volume ***TR: Test Result

CONCLUSION

The results show that we have a minimal error rate for each test case. Overall view of this research shows that we have dealt with parametric modeling of lumbar vertebrae with using a basic approach. Model we used was not the very realistic. However, our intent was to represent the vertebra as basic as possible to avoid any possible flaws that may have mattered during the testing stages. We have implemented a software application that can make use of user input and with ease, it can turn these inputs into a set of models which represents a lumbar vertebral formation in a 3D environment.

Future Works

Improvements can be made on the model representing the vertebra. A realistic model can be taken into account in the future works. This could benefit the research in certain situations such as resemblance and model physics if some further tests will be done. A realistic model also will be good to observe more aspects from object in any manner.

Secondly, accuracy can be improved. In our research we took 4 points for each vertebra, which in our case were the corner points, can be increased. Obtaining 6 to 10 points will increase the accuracy and decrease the error rate for most circumstances.

Thirdly, the view angles can be increased. In our research we developed our application to handle two different view angles. However, if obtained, a third angle of view can be useful and also will give the opportunity to calculate the rotations of the axis-z in some situations.

Bibliography

[1] Kurutz, Marta. "Finite Element Modeling of the Human Lumbar Spine." Finite Element Modeling of the Human Lumbar Spine (2010).

[2] M. Haghpanahi, M. Javadi. "A three dimensional parametric model of whole lower cervical spine." (2011).

[3] MH, Pope. "Biomechanics of the lumbar spine." Annals of medicine (1989).

[4] Nicola CAPPETTI, Alessandro NADDEO, Arcangelo PELLEGRINO, Giovanni Francesco SOLITRO, Francesco NADDEO. "PARAMETRIC MODEL OF LUMBAR VERTEBRA." (2010).

[5] Panjabi M, White A. "Physical properties and functional biomechanics of the spine." Clinical biomechanics of the spine (1990).