2019•2020

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Semi-automated presurgical planning of maxillofacial reconstruction with fibula free flap: software development and evaluation

PROMOTOR :

Prof. dr. ir. Luc CLAESEN

PROMOTOR :

dr. ing. Yi SUN

Leander Van Cappellen, Remy Vandebosch

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding UHasselt en KU Leuven

2019•2020

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Semi-automated presurgical planning of maxillofacial reconstruction with fibula free flap: software development and evaluation

PROMOTOR :

Prof. dr. ir. Luc CLAESEN

PROMOTOR :

dr. ing. Yi SUN

Leander Van Cappellen, Remy Vandebosch

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

UHASSELT KU LEUVEN

Deze masterproef werd geschreven tijdens de COVID-19 crisis in 2020. Deze wereldwijde gezondheidscrisis heeft mogelijk een impact gehad op de opdracht, de onderzoekshandelingen en de onderzoeksresultaten.

Acknowledgement

In this chapter we would like to express our gratitude towards everyone who helped us in the realisation of this master's thesis. First we would like to thank our external supervisor Dr. Ing. Sun Yi at UZ Leuven campus Sint-Rafaël. His guidance and feedback has mostly shaped our thesis. Furthermore, we would like to thank our internal supervisor Prof. Dr. Ir. Claesen Luc and Ing. Swinkels Wout for lending us an office space and providing feedback and suggestions. Finally, we would like to thank the MeVisLab community. They always responded quickly to our questions on the forum and helped us solve the most challenging problems.

Contents

List of Tables

List of Figures

List of symbols

Axial view	From top to bottom.
Coronal view	From front to back.
CSO	Contour segmentation object.
CT	Computed Tomography.
Cutting guide	A 3D printed guide to tell the surgeon where to cut.
Cutting plan	The planning that decides where the osteotomy should be performed.
DICOM	Digital imaging and communications in medicine, a file format used in hospitals for medical images.
Double barrel	The act of stacking two fibula pieces on top of one another.
Fibula	The thin bone in the lower leg used for reconstruction.
Fibula free flap	The fibula bone connected to the soft tissue surrounding the bone.
GUI	Graphical user interface.
JSON	Javascript object notation, a standardised file format for storing variables.
Macro	A series of instructions that can be executed using a single instruction.
Mandible	The bottom part of the jaw that is able to move.
Mandible tangent plane	The plane that just touches the lowest part of the mandible.
MDL	MeVisLab definition language, a programming language to build a GUI in MeVisLab.
Osteotomy	Cutting a bone.
Peroneal artery	The artery that runs alongside the fibula bone.
Piecewise linear regression	A linear regression where all pieces all connected.
Polyhedron	An object with many faces.
Pre-surgical planning	The planning that is being performed before the surgery takes place.
Qt	Open-source toolkit for creating graphical user interfaces.
Sagittal view	from Left to right.
STL	Standard tesselation format, a file format that uses triangles to describe an object.
WEM	Winged edge mesh, a polyhedral object described by using vertices, edges and faces.

Abstract

When the mandible suffers from tumors, illnesses or physical trauma, reconstruction with the fibula free flab is a common method to restore both its functionality and shape. Currently, the presurgical planning of this reconstruction is done manually at the UZ Leuven. Because this task can be very time consuming and tedious, a new planning software with a semi-automated approach is necessary. This thesis explains the design decisions and structure of the new created software, which is programmed in the MeVisLab environment. The general approach is to split the reconstruction into smaller steps where the software provides tools and suggestions to aid the user. The software supports one, two and three segments reconstructions. A double barrel approach is also possible. The evaluation is performed by a medical expert from the UZ Leuven. It is concluded that the quality of the new software is high enough to perform reconstructions on patients. Also the step by step approach eases the user experience. However, the user experience and ease of use are not optimal. The user interface is sometimes crowded and confusing. Future work exists of adding new features and improving the existing ones. The current software forms a good framework for future additions.

Abstract in Dutch

Wanneer de onderkaak lijdt aan tumoren, ziekten of een fysiek trauma is kaakreconstructie met de fibula-free flab een veelgebruikte methode om zowel de functionaliteit en vorm te herstellen. Momenteel gebeurt de prechirurgische planning van deze reconstructie handmatig aan het UZ Leuven. Omdat deze taak erg tijdrovend en vervelend kan zijn is een nieuwe planningssoftware met een semi-geautomatiseerde aanpak nodig. Deze thesis bespreekt het ontwerp en de structuur van de nieuwe software, die is geprogrammeerd in MeVisLab. De algemene benadering is de reconstructie opsplitsen in kleinere stappen, waarbij de software tools en suggesties aanbiedt om de gebruiker te helpen. De software ondersteunt reconstructies met één, twee en drie segmenten. Een double-barrel reconstructie is ook mogelijk. De evaluatie wordt uitgevoerd door een medisch expert van het UZ Leuven. Uit evaluatie blijkt dat de kwaliteit van de nieuwe software hoog genoeg is om reconstructies bij patiënten uit te voeren. Ook de stapsgewijze aanpak verlicht de ervaring voor gebruikers. Het gebruiksgemak is echter niet optimaal, de interface is soms druk en verwarrend. Toekomstig werk bestaat uit het toevoegen van nieuwe functies en het verbeteren van de bestaande functionaliteiten. De huidige software vormt een goed raamwerk voor toekomstige aanvullingen.

Chapter 1

Introduction

1.1 Scope of the problem

The mandible is an important organ in day to day activity. It plays a crucial functional role in speaking and eating. The mandible also contributes to the shape of the face, so it also plays a crucial role in the appearance. Unfortunately, the mandible can suffer physical trauma and can be affected by a tumor or bone disease. These illnesses can interfere with the function and appearance of the mandible. The best method to restore the function and appearance of the mandible is to reconstruct it with the patient's fibula bone. The fibula bone is not necessary for normal activities and has a good shape to perform the reconstruction. During the surgery, osteotomy will be performed to remove the ill part of the mandible. Next, the fibula will be removed and cut to fit in the gap left by the osteotomy. Finally, the fibula will be placed in the mandible. Because the fibula must fit perfectly in the mandible, the surgeon cannot plan the reconstruction during the surgery. The best solution is to create a planning before the surgery that contains the exact dimensions and locations of the cutting.

The pre-surgical planning is created by medical experts and surgeons from CT-data. The planning is unique for each patient, so a lot of consideration is needed to create the best planning for each patient. The planning phase takes over one half hour to create, depending on the case. This planning time can add up quickly, so the use of a different planning method can release pressure from the surgical team. This brings forward the central question of this thesis. Can we create a planning program that can reduce the planning time of mandible reconstruction with the fibula free flab?

1.2 Analysis of the existing method

The current pre-operative routine exists out of multiple steps. First, 3D models of the mandible and fibula are extracted from CT-data. Then the pre-surgical planning is preformed to reconstruct the mandible. Finally, a cutting guide is created that can be used during surgery to cut the bones like specified in the planning. This thesis will focus on the second step: the creation of the pre-sugical planning. Currently, the UZ-Leuven uses ProPlan to create the planning. ProPlan is a Cranio-maxillofacial planning tool. It has a lot of functions to create pre-surgical planning. The extensiveness of ProPlan allows the medical team to create a planning for even the most

complex cases and surgeries, but it also results in a steep learning curve for beginners. ProPlan also requires a mostly manual approach, but has also semi-automatic steps. The medical team has to go through the same manual planning routine each time, a task that can become tedious and can take a lot of time.

1.3 Reconstruction methods

The result of the pre-surgical planning is always unique for each patient. The location, size and nature of the patients condition play a role in the decision making of the medical team. There are multiple approaches to the reconstruction, but this thesis will be limited to the most common methods. The general approach is always the same. The medical team will try to cut and place the fibula so it best mimics the natural shape of the mandible. Depending on how much of the mandible is removed, a different number of pieces are required. Figure 1.1 shows the most common configurations. It is important to note that these examples just show the possible reconstructions and are not created by a medical expert. (a), (b) and (c) show one, two and three piece reconstructions respectively. (d) show the addition of a vertical piece.

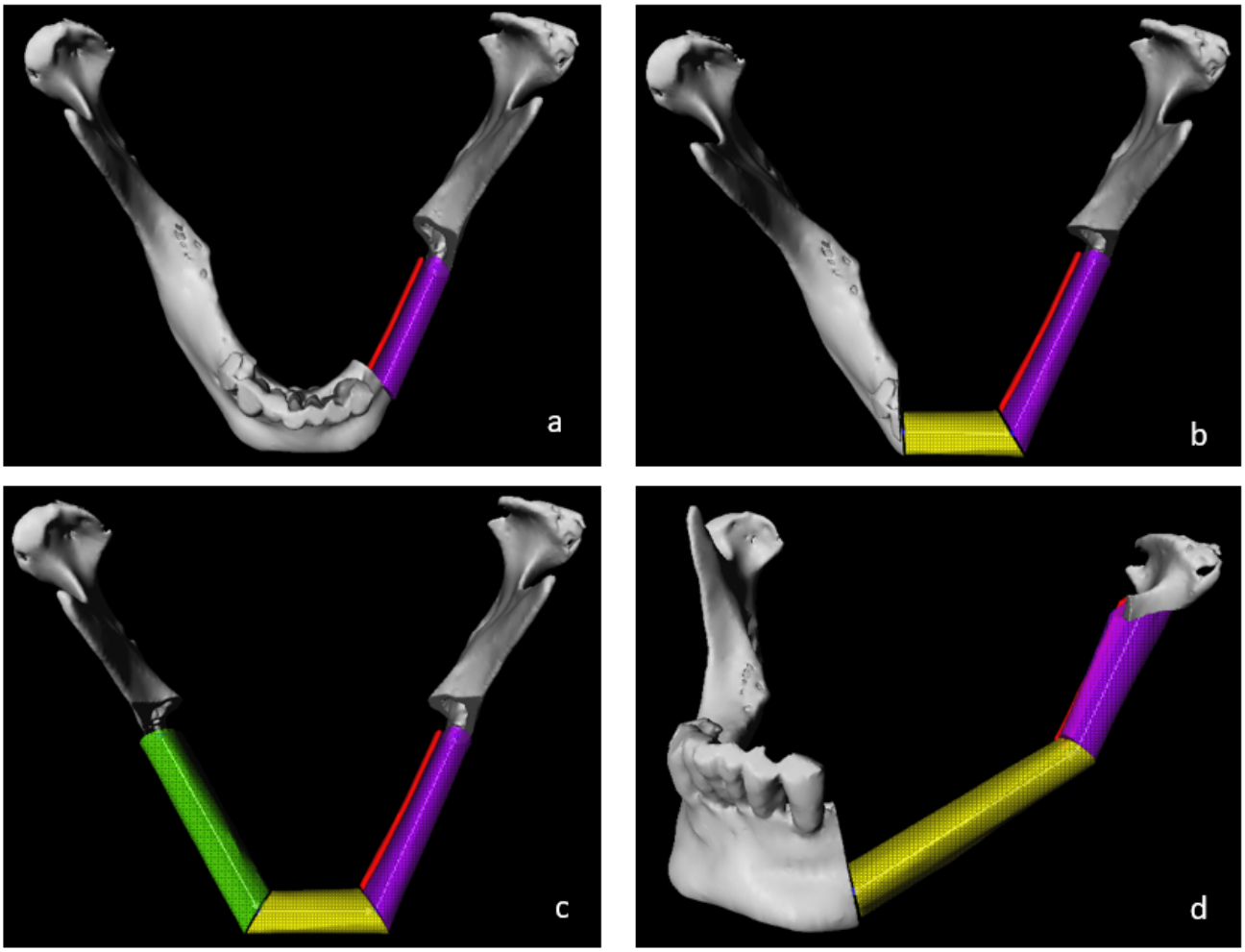


Figure 1.1: Examples of possible reconstructions with (a) one segment, (b) two segments, (c) three segments and (d) a vertical piece.

The reconstruction involves the use of the fibula free flap. This means not only the fibula bone

is used, but also a piece of the tissue connected to the fibula. This tissue is not cut during the osteotomy, so even tough the bone is cut through, the bone pieces are still held in place with the free flap. Because of this, fibula pieces cannot be interchanged by each other. The peroneal artery is also moved with the free flap to provide blood supply when the fibula free flab is placed in the mandible. This artery must be connected to either left or right of the mandible to the veins of the surrounding tissue. Because a free flap is used and the artery must be connected, the possible fibula free flap orientations are limited within the mandible.

The medical team can also choose to perform a double barrel reconstruction. Figure 1.2 shows how this technique stacks two fibula pieces on top of each other. The double barrel can also be performed with more than two fibula pieces. During the double barrel, the placing limitations caused by the free flap stay present.

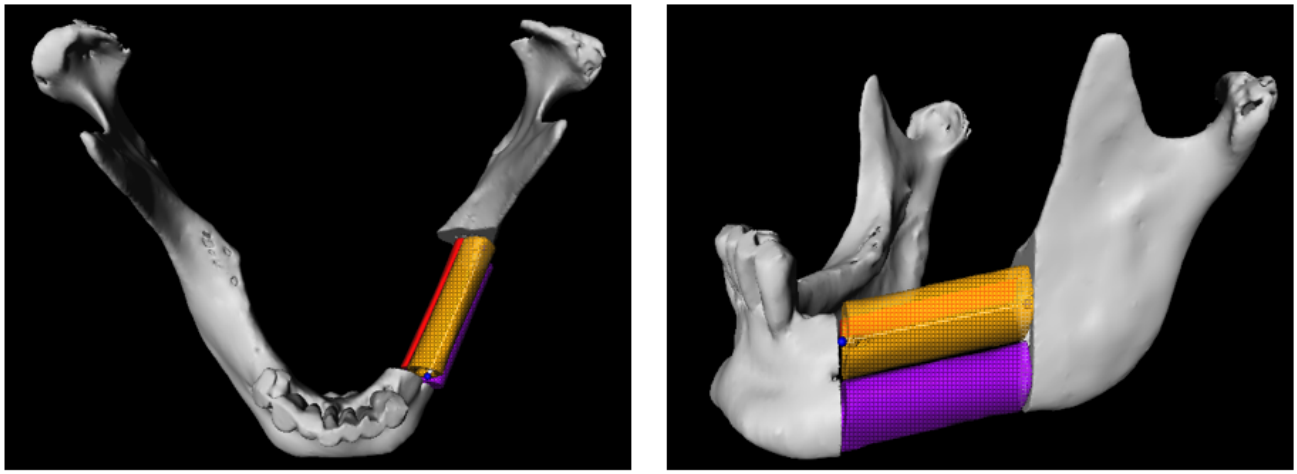


Figure 1.2: Example of a double barrel reconstruction.

The pieces for the reconstruction can not be taken from a random place in the fibula. The fibula is connected to the knee and ankle. Pieces are taken as close as possible to the ankle. However, a piece of approximately 70mm must be left connected to the ankle. The peroneal artery runs from the knee to the ankle on the interior side of the fibula. This also leaves the medical team the choice to place the interior side of the fibula facing outwards or inwards in the mandible.

During the planning, there are a couple of standards that need to be taken into account to achieve reliable results. These requirements are as follows:

- The minimal distance between pieces is 2mm on the fibula. This is to take the saw thickness into account.
- With double barrel, the minimal distance between a piece that will be placed on the bottom and a piece that will be placed on the top must be 15mm on the fibula. This is because the free flap tissue must have enough length to bend 180°.
- The shortest length of a fibula piece is 20mm. This distance is measured in reference to the length of the bone on the shortest side.
- The distance between the ankle and the first piece is at least 70mm.

1.4 Related work

Using technological solutions in medical applications is a hot topic. Unfortunately, the pre-surgical planning of the mandible reconstruction is a topic where there is no real industry standard. There are multiple papers published where researchers create algorithms to fully automate the planning [3][4]. The results of this research shows that automated reconstruction is possible, but has a few drawbacks. Automated algorithms suffer high variance in their accuracy. They also have trouble dealing with highly deformed mandibles. However, intervention of the medical staff and preprossessing the data improves the results. These methods are in full research and there is no real industry standard to perform the reconstruction. The best solution is to create a semi-automated planner where an algorithm can give initial solutions for the reconstruction and the software provides all the tools to allow the surgeon to change the reconstruction at any time.

1.5 Aim of the project

The new reconstruction planner software should improve the currently used methods. The current used software, ProPlan, is extensive, mostly manual and has a steep learning curve for beginners. The new software should be able to provide the following benefits to the user.

- Provide an easy and fast workflow for each reconstruction. Providing an easy workflow makes a low entry level possible. The software should also only contain the necessary functions to complete the reconstruction, eliminating the overflow of options that traditional software provides.
- Give suggestions for multiple steps in the workflow. This can eliminate the tedious routine tasks that the users perform nowadays, but can also give suggestions that can aid the user to create the reconstruction.
- Provide the user with all the tools to easily change or fine tune the suggestions given by the software. This is important because it gives the user all the tools to perform the reconstruction and so eliminates dependency on the automated algorithm. These tools should be intuitive and easy to use.
- Provide the same level of quality as the currently used software in a shorter planning time. The current planning can take over one half hour, so reducing the time without sacrificing quality is important.

Because the software must meet the requirements above, it cannot include all possible reconstruction cases. A selection must be made of the most common cases that are applied in reconstruction. The software will support up to three piece reconstructions with the option for vertical piece placement. There will also be support for double barrel reconstructions with a maximum of six pieces total.

There also needs to be clear agreements about the input and output of the reconstruction software. STL files of the mandible, fibula and peroneal artery will be used as input data. DICOM data of the CT-images can also be loaded in for additional visual aid during the reconstruction, however this is not mandatory. The output consists out of STL files of the reconstructed mandible and STL files of the fibula pieces. These outputs can then be used to create the cutting guide.

1.6 Outline of the thesis

Chapter 2 will discuss the tools of MeVisLab. After giving a brief introduction on how to work with MeVisLab, the different data types will be explained. This will also provide more insight on the workings of MeVisLab. Chapter 3 explains the implementation of the reconstruction software. All of the different components that are developed will be discussed. To provide a look on why certain components are build in a certain way, the most critical design choices are also explained. This chapter also explains the user interface. Chapter 4 will discuss the evaluation of the software and the future work that can be performed.

Chapter 2

Tools

2.1 MeVisLab

MeVisLab is a modular framework designed for image processing research and focuses on medical imaging. It comes with a lot of software modules designed for image segmentation, registration and volumetry. To achieve all of this, MeVisLab uses a number of third party libraries. The application framework Qt and Open Inventor are the most important ones. There are also the scripting language Python and the graphics library OpenGL that are integrated. MeVisLab is developed by MeVis Medical Solutions AG in close cooperation with the Fraunhofer MEVIS research institute [5].

2.1.1 Development inside MeVisLab

There are three different levels on which development inside MeVisLab can be achieved. There is the visual level that is the easiest and quickest way of development. It is done by connecting different modules together in a simple plug and play fashion to create something that is called a network. Development can also be done on a scripting level using Python. This is more complex and also slower than the visual level, but allows to create more complex interactions between modules and implements dynamic functionality inside a network. Last there is the C++ level, this level of development allows a developer to create his own modules and implement new algorithms. These three levels of development focus mainly on functionality. Beside this MeVisLab has created the MeVisLab Definition Language (MDL). This XML variant allows a developer to create a graphical user interface (GUI) to hide the complexity of the underlying network. MDL contains basic GUI elements like buttons and check boxes. Furthermore there are GUI elements for visualisation like labels and different types of viewers. A useful feature of MDL is parameter binding. An example of this is binding a checkbox of the GUI with the on-off field of a SoToggle module to turn rendering of a sub scene on and off [2].

The visual level of development is based on modules. There are three different types of modules that are separated using colours. Blue modules are ML modules. ML modules are used for processing voxels. Green modules are Open Inventor modules to process visual scene graphs. Brown modules are macro modules. Macro's are combinations of modules. They are networks that are represented by one single module. Most modules have inputs and outputs to make connections. There are also three different types of connectors to pass data between modules. A

connector can have the shape of a triangle, this means it can pass ML images. A second possibility is a half circle. A half circle connector can pass inventor scenes. The last option is a square. Square connectors are designed to pass base objects, an example of a base object is a Winged Edge Mesh (WEM). Furthermore there are also two different ways to connect modules together. A connection can be made between connectors on the modules and these type of connections are data connections. When a connector is selected to make a connection, the connectors that this certain data type can connect to will light up. Data connections can not only be made by dragging the mouse between two connectors, there is another method called connecting by proximity. Connecting by proximity is done by dragging a module close to the top or bottom of another module that it has to be connected to and MeVisLab will automatically connect them. This technique is particularly useful to insert a module into an existing connection. By hovering the module over the connection and releasing the mouse button, the module will be inserted into the connection. Besides data connections there are also parameter connections that connect parameters between different modules together or even inside the same modules in a unidirectional way. Parameter connections can also be made bidirectional by using two parameter connections. This way when one of the parameters change, the other one is updated automatically. Every module has a panel. The automatic panel is always available and displays all of the fields of a module along with their data type and value. There is the option for a module to have another panel that is written in MDL [2].

Macro modules as stated before contain an image processing pipeline within and capture their macro behaviour in a single module. A macro module consists out of three different files. A *.mlab file that defines the modules of the internal pipeline. This file can be edited visually in the MeVisLab GUI. A *.script file that describes all the inputs, outputs and parameters of the module. In this file the GUI can also be written using MDL. This GUI is displayed upon opening the macro module. Lastly there is a *.py file where all the Python code is written into [2].

The scripting level of development is done using Python scripts. The Python functions can be triggered using field listeners or user interface controls. To make use of Python the mevis library has to be imported into the Python script. By calling the ctx.field() function the script can manipulate field parameters of modules. On C++ development level a developer does not have to start from scratch. To design a module with a certain function in mind like for example an image processing module, the module can inherit from different classes that are defined in the ML library. These classes contain predefined functions for basic manipulation of the input data [2].

Because MeVisLab contains more than 2000 modules, there is an extensive help resource that describes every module, scripting function and MDL function available. By selecting a module in the MeVisLab GUI and pressing F1, MeVisLab will open the help page for that specific module. The same works for scripting functions and MDL functions inside the MATE IDE. The entire help documentation can also be accessed under the developer tab at mevislab.de [2].

2.1.2 Data types

MeVisLab has multiple data types. Each of these data types can be converted into one another. One of these types is a winged edge mesh, abbreviated as WEM. This data type contains surface information in the form of faces, edges and vertices to render an object in 3D. Another datatype for this purpose is Open Inventor. Open Inventor uses polygons to render objects. Besides WEM

and Open Inventor there are also Voxels. Voxels can be seen as pixels in 3 dimensions. Essentially Voxels divide a 3D space into cubes. MeVisLab has a special data type for contours. This data type is called CSO and this acronym stands for Contour Segmentation Objects. It allows for automatic or interactive generation of contours [2]-[6].

A winged edge mesh or also called polygon mesh is a polyhedral object that is described using vertices, edges and faces. The object is usually constructed using triangles, but quadrilaterals or simple polygons can be used as well. There are many different operations that can be performed on meshes. These include operations like boolean operations and smoothing. A representation of how a polygon mesh is constructed from vertices all the way up to an object can be seen in figure 2.1 [7]-[8].

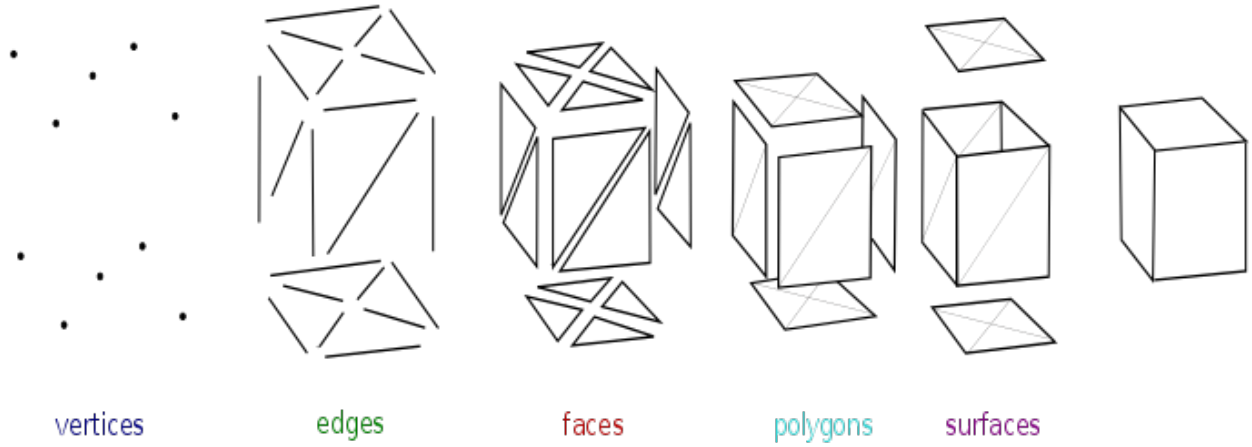


Figure 2.1: Representation of a polygon mesh [1]

Open Inventor is called a datatype in MeVisLab but is essentially an abstraction layer to program OpenGL. Open Inventor was created to address an issue when programming OpenGL. The order in which the instructions are send to the OpenGL engine determines the performance significantly because the programmer has to avoid that objects are rendered that are invisible in the resulting image. In the year 2000, Open Inventor was released under open source license. This version has been tweaked by MeVis Medical Solutions and is still in use by MeVisLab today. Inside MeVisLab, the Open Inventor modules can be used to build what is called a scene graph. The modules are called nodes and represent a 3D object that can be drawn. The properties of this 3D object that can also be altered. The order in which the different nodes are rendered depends on the position of there connection. The traversal order of the nodes is from top to bottom and from left to right as seen in figure 2.2. Any changes of the fields of Open Inventor modules are stored in a queue and are executed asynchronously [2], [9].

The last datatype to discuss are voxels. In contrast to WEM and Open Inventor scene's, voxels do not have their position encoded trough coordinates in a world coordinate system, but its position is based relative to other voxels. Various properties can be represented by a voxel. In CT scans, which are used for mandible reconstruction, the value of a voxel is in Hounsfield units. The Hounsfield scale is used to describe radiodensity of an object. Voxels are used mainly in medical imaging to represent volumetric data. Besides this purpose, voxels are also used in games to build terrain maps, think of games like Minecraft or Crysis [10]-[11].

MeVisLab can also handle multiple file types. Only the most important two are discussed here.

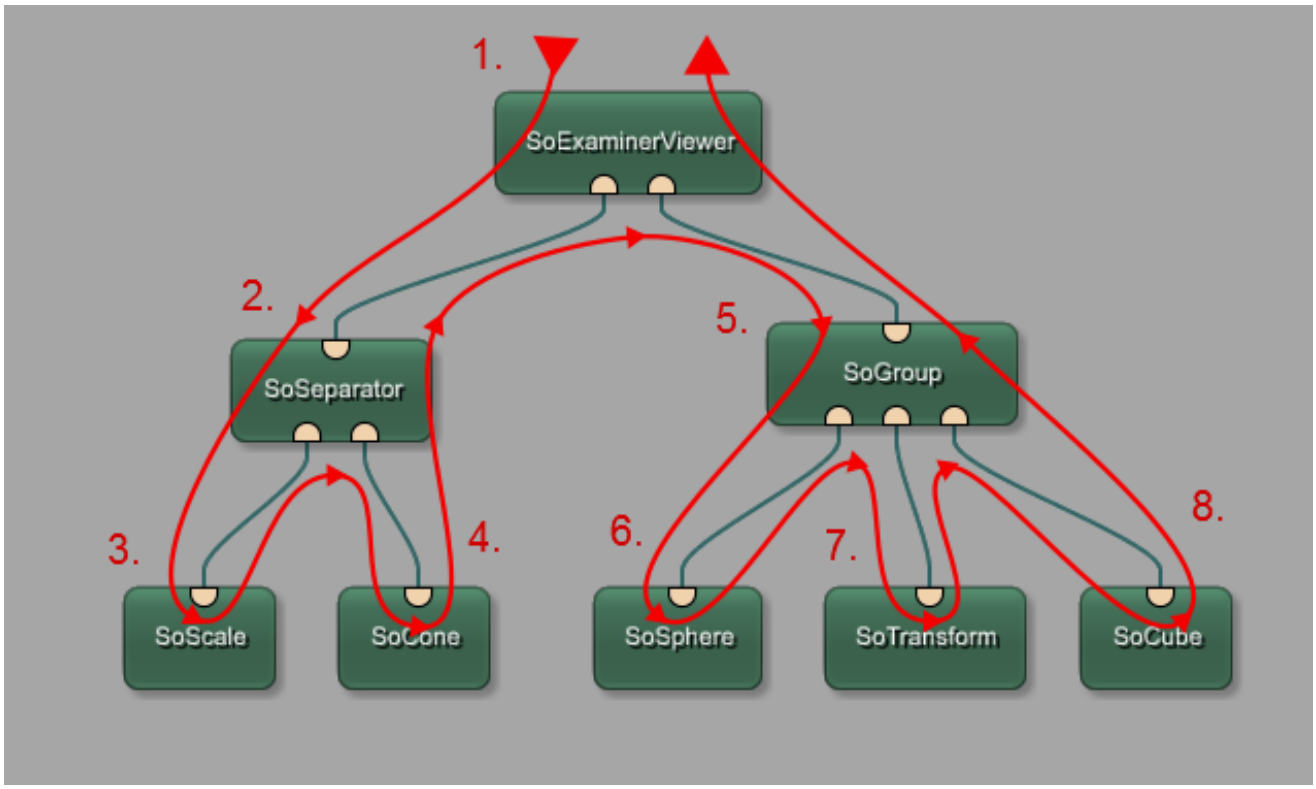


Figure 2.2: The traversing order of Open Inventor nodes [2]

First there is the STL file format. STL is an abbreviation for stereolithography but there are many different names for this format. In MeVisLab it is called Standard Tesselation Format. The STL format is used to describe a 3D object based on triangles, this property makes the STL format ideal to load into MeVisLab as a WEM object. The second file format is the DICOM format. This again is an abbreviation and stands for Digital Imaging and Communication in Medicine. It is the standard for storing and sharing medical image data. It is most commonly used to store computed tomography (CT) data or magnetic resonance imaging (MRI) data. By using a DirectDicomImport module DICOM files can be loaded into MeVisLab as ML images [12]-[13].

Chapter 3

Implementation

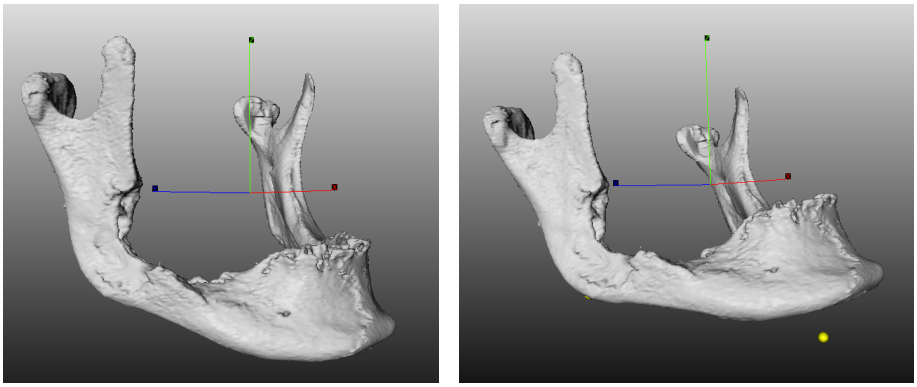
The 3D-reconstruction of the mandible with the fibula free flab is a complicated process and doing it in one step is way too complicated. Because of this, the software is designed to split the reconstruction into multiple smaller steps with easier to solve problems. With each step, the complexity of the reconstruction is reduced. The general flow of the software goes as follows. First, the user will cut the 3D mandible with cutting planes to remove the dysfunctional part. Next, the center line of the of the mandible is calculated. This will reduce the complexity from a 3D volume to a 3D line. Next, the 3D line is projected on planes the fit the shape of the mandible in the vertical and horizontal direction. Now the problem is 2D, it is easy for the computer to create an automatic reconstruction with a small calculation time. The software will place lines trough the 2D centerline of the mandible that will function as the centerline of the fibula pieces. It is also easy for the user to adjust the segment's fit in the 2D editor. Next, the software will reverse the steps. The 2D placement becomes 3D and the software will place pieces of the fibula on the centerlines that were determined in the previous step. Now the software has placed 3D fibula piece volumes inside the 3D mandible and the user can make final adjustments to fine tune the result before exporting the reconstruction.

This chapter will go over the design of all the internal macros. The internal network will be discussed and explained. Furthermore, the Python code that accompanies every macro individually is also discussed. It will not explain all the functions and modules that were used in detail because this would be overwhelming. The internal working of the MeVisLab functions and modules can be found on the MeVisLab website. The functions that where custom created for this project are documented in the project itself with the use of comments. The internal network can be seen in figure A.1. The complete application is implemented as a single macro that can be loaded into MeVisLab. This single macro consists out of 11 custom built macros and a SettingsManager module that saves all the necessary parameters upon saving the project. These macros are connected in a way to achieve the flow that is described above. It is important to note that the images that contain MeVisLab networks are not very clear. This is because the networks are quite large and it is hard to include them in an A4 format. The images are an aide for the explanation. For details, it is better to look at the original MeVisLab code.

3.1 Processing network

3.1.1 Loading the files

In order to load the desired STL files into the program, two different macros have been constructed. The first macro is called "LoadMandible". The network of how this macro is constructed can be seen in figure A.2. Starting with a WEMLoad module to read in the data from the specified STL file, the amount of polygons is reduced by 50% with a WEMReducePolygons module. This increases the speed of further calculations. The location of the mandible in 3D space is different per use case when first loaded in. That is why a WEMModify module is used to center the mandible. This means that the center of the mandible is now at the origin of the coordinate system. After the centering, the WEM is rendered as an inventor scene to enable the possibility for easy visualisation of the mandible. All the viewers in MeVisLab are for either inventor scenes or images. So to visualise a WEM object it has to be rendered as an inventor scene. Unfortunately the angle of the mandible is also different per use case which complicates the use of automated tools for the reconstruction. To solve this issue a system has been developed. This system is build into the LoadMandible macro and can be seen in figure A.2. It takes advantage of user input by letting the user select three points on the bottom of the mandible by using a So3DMarkerEditor module. Using these three points the mandible tangent plane can be calculated. Rotating the tangent plane so that it lines up with the xy-plane assures that the position of the mandible is the same for each reconstruction. The result of this operation can be seen in figure 3.1.



(a) (b)

Figure 3.1: Rotation of the mandible tangent plane to the xy-plane. (a) Mandible before rotation. (b) Mandible after rotation.

In the top of figure A.2 it can be seen that the LoadMandible macro has two outputs, the inventor scene of the mandible is brought out of the macro as well as the WEM format of the same mandible. The second macro that has been constructed for the purpose of loading the necessary files is called "LoadFibula". The network inside this macro can be seen in figure A.3. Like the LoadMandible macro, this macro also starts with a WEMLoad module, but in the LoadFibula macro there are 2 WEMLoad modules. The first one is for the fibula bone and the second one is for the peroneal artery. When the fibula bone is loaded into the program, a WEMDemergePatches module is used to seperate the outside of the bone from the inner channel that holds the bone marrow. This is done to speed up further calculations. Only the outside of the bone has to be visualised to perform the mandible reconstruction, that is why the inner

channel can be removed safely. After this step the amount of polygons is reduced to 15% in order to speed up further calculations. This is feasible because there is no need for high resolution in the fibula bone. After this reduction the fibula is centered using a WEMModify module. It is followed by rendering the fibula bone into an inventor scene and determining the longitudinal axis and performing a rotation on the inventor scene to align this axis with the z-axis. When this alignment is performed, the flow of the artery is from top to bottom. This assures that the starting point for the reconstruction is always the same. As mentioned before the second WEMLoad module loads the peroneal artery into the program. The amount of polygons is also reduced here to speed up calculations. Furthermore, the translation and rotation performed on the fibula bone are also applied on the artery. This assures that they are both aligned properly and that no distortions are present that could cause a faulty reconstruction. This macro has four different outputs. These include the fibula as an inventor scene and as a WEM as well as the artery as an inventor scene and a WEM. Besides this network there is also a Python function that is executed. This function calculates the centerline of the fibula. Because every fibula is different and they are never perfectly straight, it would be wrong to assume the centerline of the entire fibula can be approximated by a straight line. This is done by taking a plane that has the same alignment as the xy-plane and moving it from the bottom of the fibula to the top in 0.5mm increments. At every sampling point the cross section of the plane and the fibula bone is sampled using the WEMClipPlaneToMarkers. Using the MarkerStatistics module, the center of gravity of this cross section is determined. All of these points are stored in a list with the appropriate Python functions to retrieve the necessary points further down in the reconstruction. The centerline that is calculated can be seen in figure 3.2.

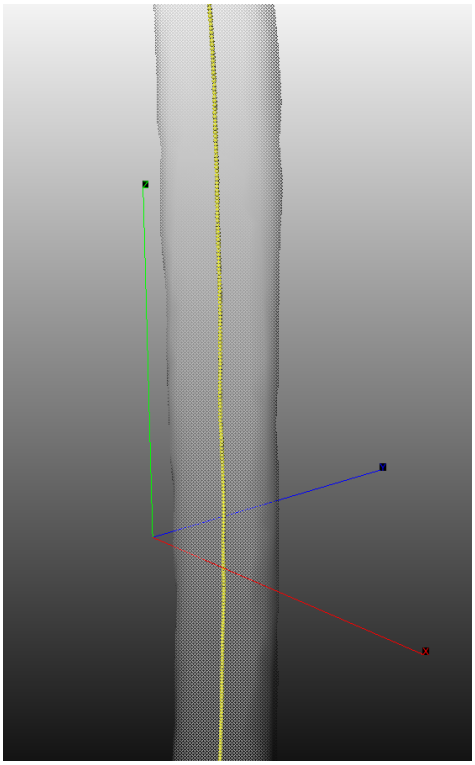


Figure 3.2: An enlarged view of the fibula and the centerline in yellow

3.1.2 Marking the artery side of the fibula bone

It is necessary for the user to know on which side of the fibula bone the artery is located further down the reconstruction. So for this purpose a specialised macro has been created named "DrawArtery". The network of how this macro is constructed can be seen in figure A.4. The macro has four inputs. The peroneal artery as an inventor scene and in WEM format as well as the fibula bone in WEM format and as inventor scene. As stated before, the goal of the DrawArtery module is to mark the side of the fibula bone where the peroneal artery is located. This can be achieved in two different ways. The first one is automatic and the second one is manual. The automatic way of marking starts with retrieving the position of the peroneal artery. After this, a copy of the fibula bone is taken and translated 2mm into the direction of the artery. Formula 3.1 and 3.2 are used to calculate this offset in the xy-plane. The variable r is the slope of the position vector in the xy-plane. R is defined as dy divided by dx as illustrated in figure 3.3. D is the distance that the object has to move in mm, in this case two.

$$x = \sqrt{\frac{d^2}{r^2 + 1}} \tag{3.1}$$

$$y = x \cdot r \tag{3.2}$$

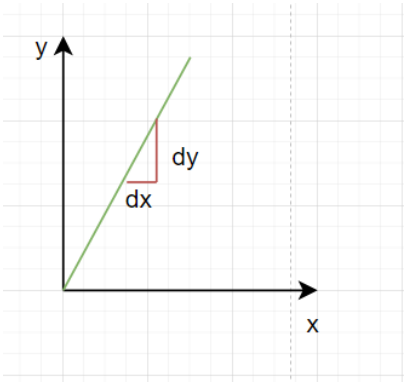
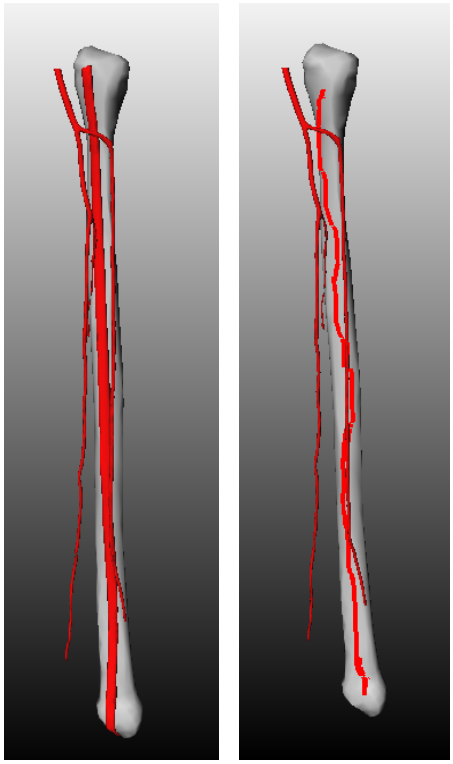


Figure 3.3: Slope r in the xy-plane.

To obtain the marker from these two bones the original fibula bone is subtracted from the translated fibula bone using a WEMLevelSetBoolean module. If this result would be used as a marker it would cover half of the fibula bone, this is to much and will cause confusion in the further steps of the reconstruction. By using a box and another WEMLevelSetBoolean module the result from the previous Boolean operation is trimmed to obtain a 5mm wide and 2mm thick line that will be used as the marker.

Besides this automatic method, the option to manually draw a marker is also available. A SoCSODrawOnSurface module allows a user to draw a line onto any surface that is visible in the viewer that the module is connected to. All the lines that are drawn in the viewer are stored as CSO's in a CSOListContainer. The resulting CSO's are then rendered as separate objects so they can be coloured differently, in this case red, and processed separately in further steps of the reconstruction. Using this technique, the user can mark whatever side of the fibula bone he desires. The result of both the marking techniques can be seen in figure 3.4. The DrawArtery macro has two outputs. One of these outputs is the marker in WEM format and the other output is the marker as an inventor scene.



(a) (b)

Figure 3.4: Marking the artery side of the fibula bone. (a) Fibula bone marked automatically. (b) Fibula bone marked manually.

3.1.3 Designing the cutting plan

A crucial part of any mandible reconstruction is the ability to remove a defect in the mandible. That is why in presurgical planning a cutting plan is designed. For the design of this cutting plan a specialised macro named "CuttingPlanes" has been constructed. The network of how this macro is designed can be seen in figure A.5. To start the design of the cutting plan the user is greeted with the mandible that needs reconstruction. Furthermore there is a cutting plane used as an indicator that the user can move. The indicator plane is 50 X 50 X 1 mm in size and indicates where a cut should be made. By using the manipulator that is attached to the plane, it can be translated and rotated into any position. Besides the manipulator there is also the option to perform all of these actions by using the arrow keys as well as the A and Z key on the keyboard. When the indicator plane has been positioned where a cut should be performed the position and angle can be fixed and the indicator plane is free to move to the next location. When a SoWEMConvertInvertor is used with auto-apply turned off in combination with a SoWEMRenderer and a SoSeparator, it is possible to make a buffer where different objects can be added to. This technique allows to place an infinite amount of cutting planes with only one indicator plane. Every cutting plane that is fixed in place will not only be stored as an inventor scene in the buffer, but the location and rotation of the plane is also stored in a 2-dimensional Python list. When the user decides that a fixed cutting plane should be moved or not used at all, it can be selected using the mouse and it will be deleted in both the buffer and the Python list. After all the necessary cutting planes have been fixed, the buffered planes will be used to make all the cuts with a single Boolean operation. For this Boolean operation a WEMLevelSetBoolean module is used. The advantage that is created by making all of the cuts with a single Boolean

operation is calculation speed. A Boolean operation takes time to calculate and by reducing the number of Boolean operations performed, a significant gain in speed can be achieved.

To achieve all of this, the macro needs two inputs: the mandible in WEM format and the mandible as an inventor scene. The macro also has only two outputs. These two outputs are two different WEM's, the first one is the cut mandible and the second WEM is the cutting plan itself. All of the different steps are demonstrated in figure 3.5. It is not easy to exactly determine where the cut should be made because certain anomalies like tumors are not visible in the STL file of the mandible. They are however visible in the DICOM files. To allow the user to see all of the details that are shown in the DICOM files, a separate macro is built to handle the DICOM files and viewers. The name of this macro is "DicomViewer" and the internal network can be seen in figure A.6. The modules in the bottom are for rendering the different overlays that can be displayed in the OrthoView2D module. One of these different overlays is the cuttingplan so that the user can see on the DICOM viewer exactly where the cuttingplan is located with respect to the soft tissue of the patient. The DICOM view is in 2D, so the 3D CT scan can be viewed from three different directions. The axial view or from top to bottom. Then there is the coronal view which is from front to back and the sagittal view which is from left to right. Furthermore, a slice of the DICOM image is taken in every direction and rendered in 3D so that is possible to overlay the DICOM data with the inventor scene of the cuttingplan as well as the mandible as shown in figure 3.6.

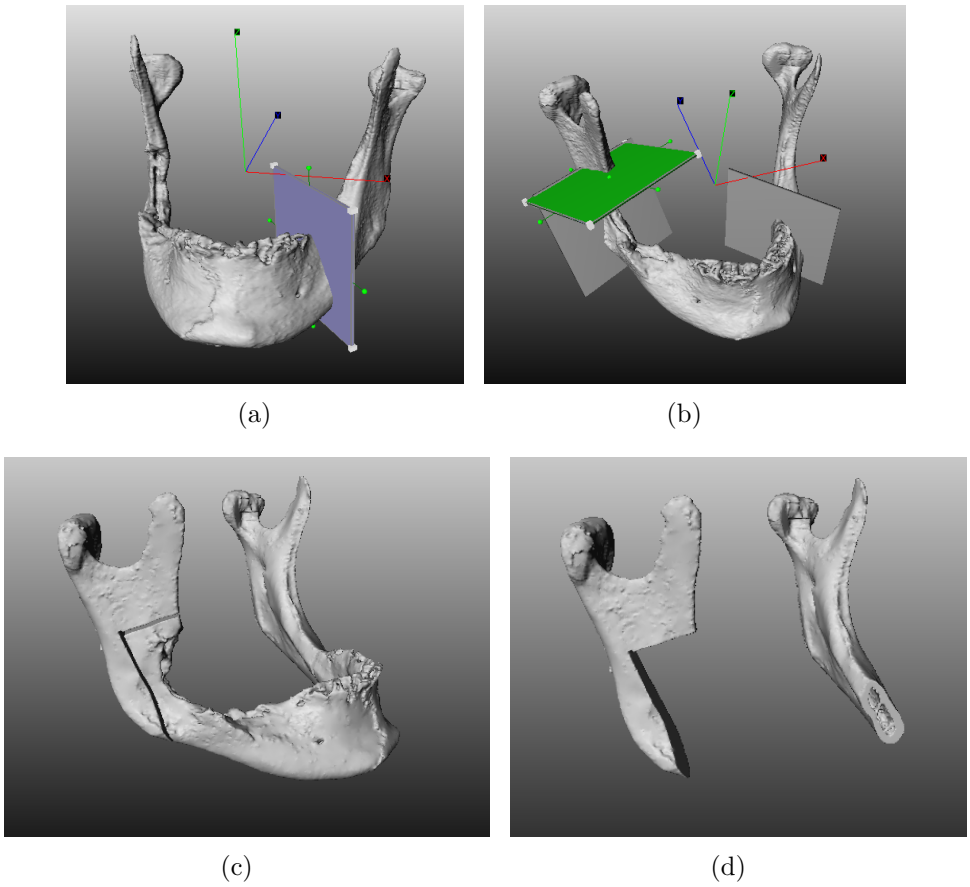


Figure 3.5: Designing a cutting plan and resecting the mandible. (a) The indicator plane in blue. (b) The finished cutting plan. (c) The mandible after cutting. (d) The resected mandible.

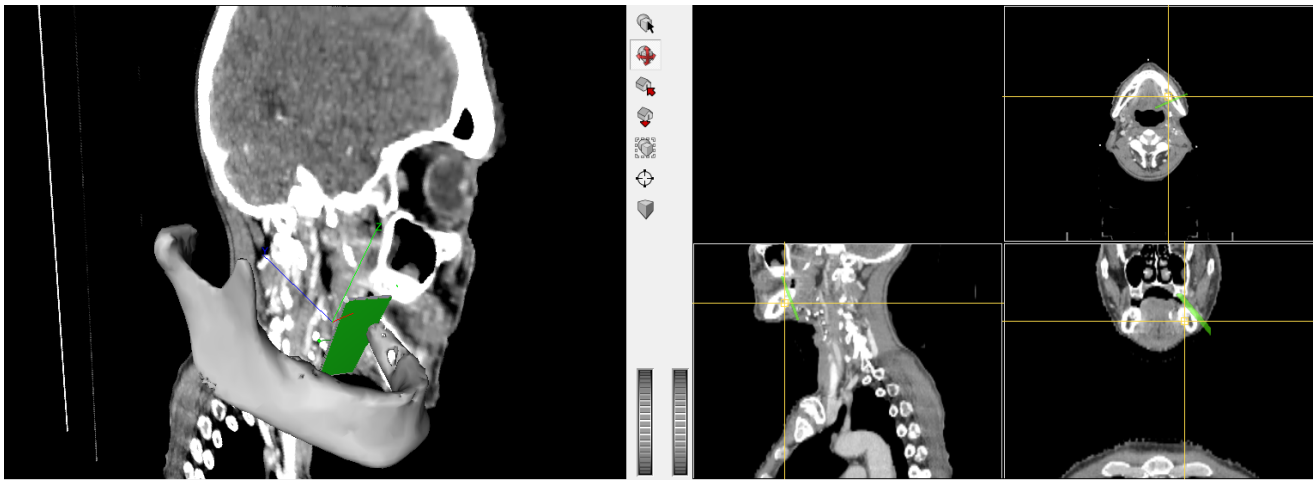


Figure 3.6: The orthoviewer for the dicom files

3.1.4 Resecting the mandible and defining the original contour

After the cutting plan has been designed and the mandible is now in multiple pieces, the original contour of the mandible has to be defined so that it can be reconstructed. The user has to select the piece of the mandible that has to be removed. Since it is very hard to consistent determine which mandible piece is in between the planes and needs to be removed, user input is still required in this step. By clicking on a piece it will be removed from the view and stored so it can be viewed later in the reconstruction process. To achieve all of this functionality, a specialised macro has been constructed called "SelectPartToWorkWith". The internal network of this macro can be seen in figure A.7. The cut mandible goes first through a WEMDemergePatches module so that the multiple different pieces can be addressed using their own unique ID. By setting a threshold value inside the module, small anomalies in the mandible will be removed as well. These anomalies are glitches in the CT scans and can be seen as small floating irregular objects. After this a WEMSelectPatches module is used to select the piece that the user wants to remove. This piece is then rendered and served as an output for future use. A second WEMSelectPatches module is used in combination with a WEMComposePatches module to select one by one the pieces that should not be removed. These pieces are also rendered as an inventor scene and provided as an output. The reason to choose this approach over a Boolean operation is calculation speed. The Boolean operation is slower in this particular step because the pieces are quite large. There is also Python code used to determine the ID of the piece that has been clicked on, as well to fill in the correct values in the parameters of the WEMSelectPatches modules. The result of all these calculations can be seen in figure 3.5.

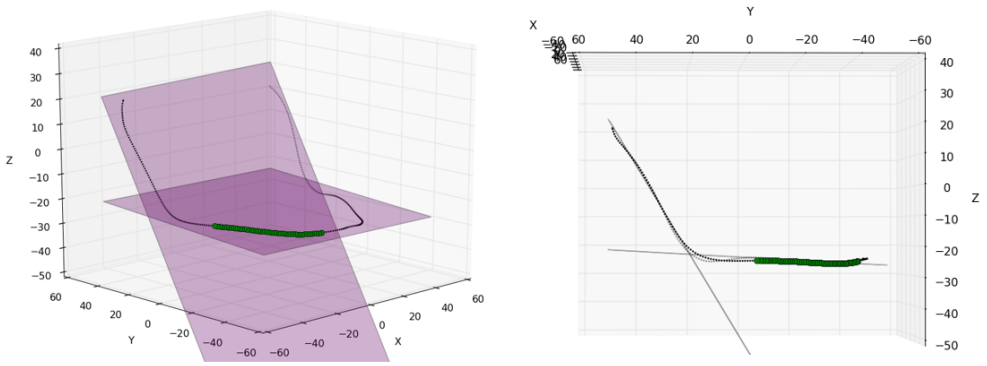
The SelectPartToWorkWith macro does not only remove a certain piece from the view, it also has a small three module network that is used to determine the exact intersection point between the cutting planes and the centerline. How this centerline is fabricated will be discussed further on in this chapter. The WEMLevelSetBoolean module can be set to calculate the intersection of two WEM's instead of the difference. When the WEM format of the centerline is used as input number one and the cutting planes as input number two, the result is two small discs that are the intersection of the centerline with the cutting planes. By determining the exact location of the intersection, the accuracy of the reconstruction is increased. This is because the lengths of the fibula pieces that are used for the reconstruction are calculated more accurately.

After the mandible has been resected, the contour of the original mandible is going to be defined. For this purpose a dedicated macro has been build. The name of this macro is "ContourMandible" and the internal network of this macro can be seen in figure A.8. In order to achieve the desired functionality, that is calculating the contourlines, some Python code is also needed. There are two different contourlines that will be calculated. These are the bottomline and the centerline of the mandible. Both of the lines are calculated according to the same principal. There is only a small difference. The main principle is a plane that has its normal vector in the xy-plane and turns around the z-axis. The plane turns around in 100 steps to achieve a good enough resolution without the process becoming too slow. The normal vector will be set in the ComposePlane module to create a plane. This plane will then be used in a WEMClipPlaneToMarkers module. By using the complete mandible in WEM format as an input the WEMClipPlaneToMarkers module will calculate the intersection of the plane and the mandible. This intersection is provided as a list of markers. These markers are 1 mm apart and are loaded into Python. From each slice the lowest marker is then determined to calculate the bottom line. To calculate the centerline a MarkerStatistics module is used. By providing the markerlist from the slice as an input, this module can calculate the center of all of these markers. From each slice, the centerpoint is determined. These points are then used to calculate the centerline. The resulting centerline has a lot of fluctuations. Since the bottomline follows the desired contour more accurately, this line will be used as the centerline by moving it up. To know how far it has to move upwards, the average distance between the orignal centerline and the bottom line is calculated. When the pieces are placed on the centerline of the mandible, they are always located too high. To accommodate for this, the average distance is divided by two. This gives a better initial placement. This new centerline now has the desired mandible shape and is at the correct position. To make sure that all the markers of the centerline are evenly spaced, a PathToKeyFrame module is used. This module interpolates the given centerline with a 1 mm resolution, meaning that there is a marker every 1 mm regardless of the density of the input markers. Besides interpolating and evenly spacing the markers, the PathToKeyFrame module also smooths the centerline so that small fluctuations are removed. The macro has only one output. This is the centerline that has been converted to a WEM format. This output is going to the SelectPartToWorkWith macro that was discussed earlier in this chapter.

3.1.5 Making the reconstruction plan

Once the contour line of the mandible and the cutting planes are known, the actual planning can begin. In this step, the software will guide the user through multiple steps to fit lines on the contour. These lines will be used as centerlines for the fibula pieces in the next steps. Fitting straight lines on this 3D contour line so it mimics the natural shape of the mandible is a complicated task. The approach to this is to simplify the problem into smaller problems and tasks where the user and software work together to get the result. The next section will explain the workflow and techniques used for the reconstruction planning. The software of this section is completely written in python because is is very specific to this application and Mevislab does not provide the necessary modules to create this step. For the specific details of the written functions, the python file ReconstructionMath.py and ReconstructionPlanner.py contain comments with explanation of all the written code.

Because a 3D reconstruction can be both difficult and complex for both machine and user, the first step is to split the problem into two smaller 2D problems. The first step is to split the contour data points into two data sets. To split the data set, the natural shape of the mandible is used. One data set will contain all the data points on the vertical part of the mandible, the other data set will contain all the data points that run horizontally between the vertical points. The mean of the z-coordinates (height) of the contour points is used to determine in what data set the contour points belong. Next, a linear regression can be performed to draw the best fitting plane through each data set. The result is shown in figure 3.7.



(a) (b)

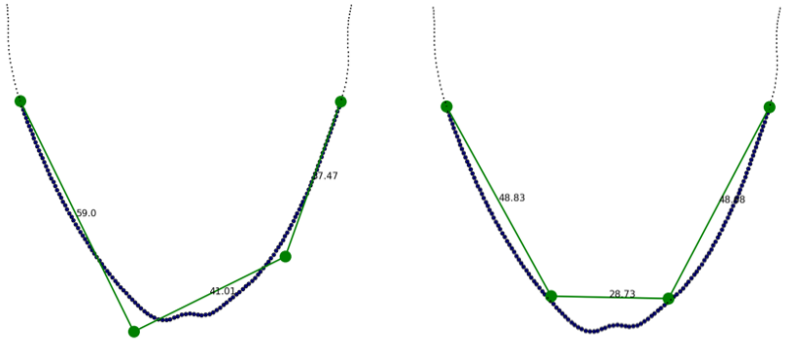
Figure 3.7: The planes that are fit trough the vertical and horizontal data. (a) a front view and (b) a side view showing the plans follow the shape of the mandible

The vertical data points are only needed when the reconstruction involves a vertical placed fibula piece. The next step is to project the data points perpendicular on the concerning plane. Now the data points lay within the 2D plane and have new coordinates in this plane.

Because the data points in the vertical dataset lay on a straight line, only one piece is needed to perform a reconstruction of this part of the mandible. This makes reconstruction of the vertical part rather easy. The challenge lies within the reconstruction of the horizontal data points. Here, multiple configurations are possible. Now the medical staff must decide the number of segments that need to be used in the horizontal reconstruction. Here, the software will give the initial position of the pieces. There are multiple methods, and multiple research teams have done dedicated research on this topic [3][4]. Searching for the most optimal solution for this problem is not the main topic of this paper. However, a fitting algorithm will be implemented that will suffice for a semi-automated planning.

Because the data points are now 2D, regression techniques can be used to fit lines through the data points. The pwlf Python library can perform piecewise linear regression where all pieces are connected. The pwlf algorithm performs as expected and place the pieces through the dataset. The result is good in certain cases, but can also be really bad. Traditional regression algorithms will try to reduce the cost function and fit the line as best as possible to the dataset. The best mathematical fit will not always result in the most natural shape for the mandible. Because of this, the use of linear regression can result in unnatural shapes. An example is shown in figure 3.8. (a) shows a three piece reconstruction with the use of linear regression, (b) is the expected result. The figure shows that the middle line is not straight. The same problems can occur in two piece reconstructions. A way to solve this problem is to create a second algorithm that only needs

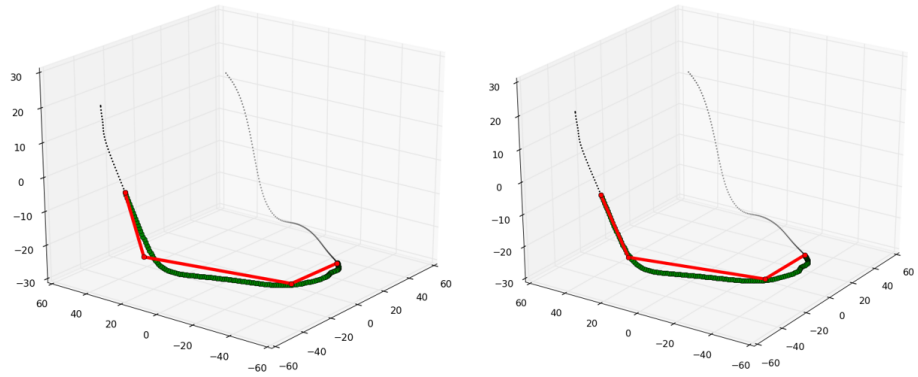
to be used when a horizontal piece on the bottom of the mandible is needed. This algorithm will scale a standard reconstruction shape to the right size for the mandible. These shapes are shown in figure 1.1 (b) for two segments and (c) for three segments. Finally, a function is implemented that uses the shape of the mandible and dimensions of the gap left by the cutting planes to decide which algorithm is best used in this case. This approach works really well and does not require a lot of computation power. Even if the algorithm makes a mistake or the medical team prefers an other shape, the user has the ability to drag around the breakpoints to change the shape as they desire.



(a) (b)

Figure 3.8: Three segment reconstruction shows why piece wise regression is not perfect. (a) shows the result of the regression algorithm, (b) the result of the custom algorithm.

Finally the software will place all the lines in a 3D view to show the final result. Both vertical and horizontal lines will be combined into one plot. If both vertical and horizontal lines are used, a new problem can occur. The shape of the reconstruction has difficulty following the shape of the mandible. More precise, the angle between the horizontal pieces and vertical pieces is too steep. Figure 3.9 (a) shows the problem, (b) shows the same reconstruction with an additional correction for the angle. The new angle is the angle between the two planes. Now the initial fit of the centreline is completed. The breakpoints of the lines, including the begin and end point, are exported to the next step of the planning process.



(a) (b)

Figure 3.9: Shows the reconstruction in 3D. (a) with the fault at the angle between the vertical and horizontal pieces, (b) with the corrected angle.

3.1.6 Cutting the fibula bone

When all the lines are determined for the reconstruction, the fibula bone needs to be cut. This section also has its own macro called "CuttingPlaneFibula". The internal network of this macro can be seen in figure A.9. This macro has three different inputs. The first two are the fibula bone in WEM format and as an inventor scene. The third input is the marker for the artery side in WEM format. This macro has only two outputs. Both of the outputs are in WEM format. The first output is the cut fibula bone and the second output is the cut marker. There are two main parts in this macro, the first one is the plane. This is only one single plane that is 100 X 100 X 1 mm in size. This plane has different rotation modules as well as different translation modules attached to it. All of the transformations can be done in a single transformation, but the choice is made to perform the transformations separately in different modules. This simplifies the debugging process as well as the Python code that calculates these transformations. Future changes made to the software will also be easier because the transformations are separated. The second important part is the buffer for the planes. This is achieved by using a SoWemConvert-Inventor with auto apply turned off in combination with a SoWemRenderer. This is the same technique as discussed in subsection 3.1.3. Then there are two WEMLevelSetBoolean modules that are used for cutting. One of them uses the buffered planes to cut the fibula bone into pieces. The other one uses the buffered planes to cut the marker, that is used to mark the artery side, into pieces.

For determining the position and angle of the planes, Python is used. The first thing that is done is to calculate all the necessary planes for cutting. Besides the planes from the cutting plan there are also cuts made at the breakpoints of all of the lines. The normal vector of these cutting planes are calculated in the following way. First the vectors of both of the lines are normalised. Then the cross product is made to calculate the normal vector of the plane the lines reside in. The normalised vectors of the lines are added together, this results in the vector that is exactly in the middle of the two lines as illustrated in figure 3.10. By taking the cross product of this vector with normal vector of the plane the lines reside in, the normal vector of the cutting plane is calculated. When all the planes are calculated they can be placed for cutting. The placement of these planes is done in a loop using Python. Most of the modules are Open Inventor nodes as explained in subsection 2.1.2, the changes are stored in a queue. As long as the Python code is not finished, the fields that it changes are not updated. The queue has to be processed before buffering the plane. This can be triggered while the Python code is running by using the MLAB.processInventorQueue() function in Python. Because the longitudinal axis of the fibula bone matches the z-axis, all of the distances can be mapped out in the z direction. These are calculated prior to the placement. This list of distances starts with the margin that is taken from the foot side of the fibula bone. The typical value for this margin is 70 mm, but it can be altered to accommodate the needs of the user. This margin is followed by the lengths of the lines, which correspond to the required lengths of the pieces. In between the lengths of the pieces there is another margin that is needed for proper cutting during the surgery. The typical value for this margin is 2 mm, but like the other margin it can be altered to suit the needs of the user. This margin applies to the closest distance between the pieces. Since the centre of the plane is used as the reference point for the placement, the shortest distance between the pieces is nearly always smaller than the required distance. Increasing the distance between the centres of consecutive planes will resolve this issue. To know how much distance has to be added, there are two more WEMLevelSetBoolean modules. The first module calculates the intersection

of the previously placed cutting plane and the fibula bone. This intersection is the actual cut performed. The second module calculates the intersection of the plane that is being placed. By using a WEMSurfaceDistance module the actual distance can be determined between the consecutive cuts and therefore the extra distance needed to achieve the required distance can be determined as well. Besides the angle of the cutting plane and its position, there is one more parameter that has to be taken into account. This parameter is the rotation that the user gives to the pieces to line them up correctly in the mandible. This parameter is initially zero but can be changed in a later stage of the reconstruction.

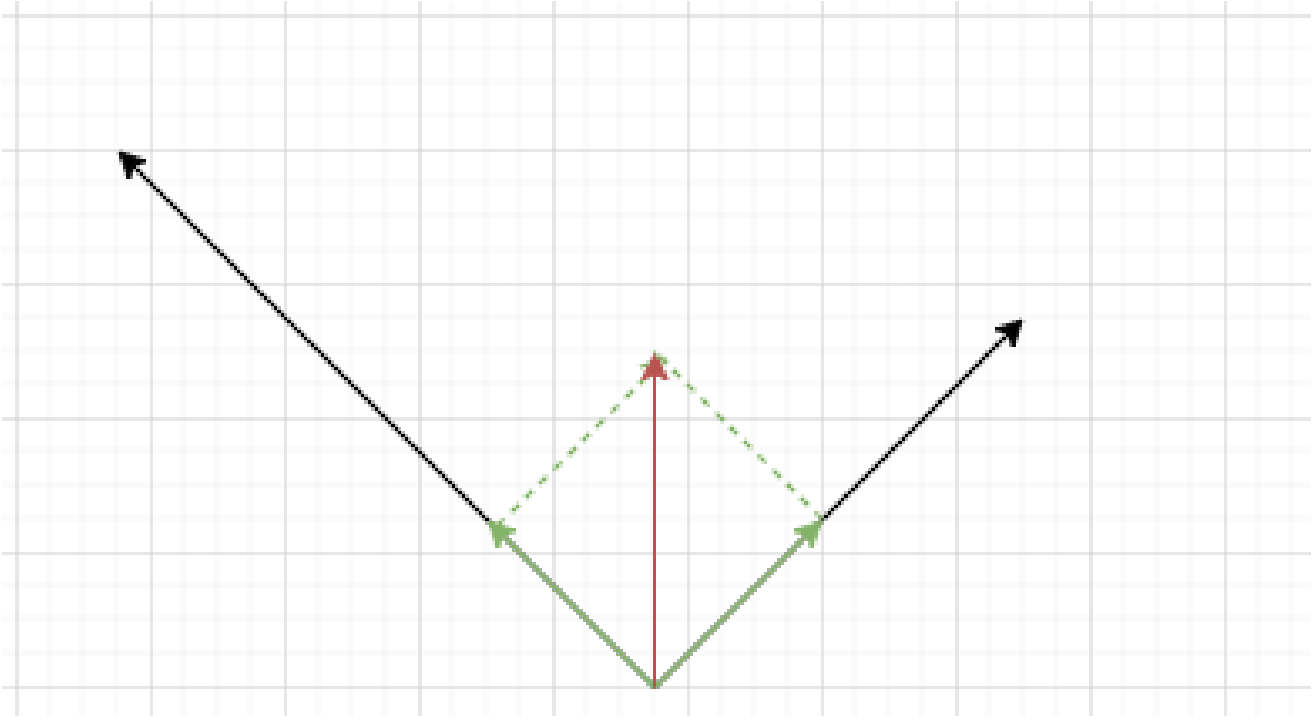


Figure 3.10: Calculating the middle of two vectors

There exists also a different technique of reconstruction that requires more complex cutting of the fibula bone. This technique is called double barrel reconstruction. In a double barrel reconstruction two pieces of fibula bone are stacked on top of one another to acquire more height. Occasionally this is necessary to make the reconstructed mandible aesthetically more appealing. Since all the pieces are connected using soft tissue, the cutting order of the pieces is inverted for the top layer in comparison to the bottom layer. An other important aspect is that the knee side of the fibula bone is located in the bottom layer. The reason for this is to allow easier attachment for the artery. This implies that while starting in the bottom, the inverted pieces have to be cut first and then the regular pieces follow. An extra margin is introduced here and that is the margin between the inverted pieces and the regular pieces. This margin has a typical value of 15 mm in order to be able to flip the inverted pieces on top. Like all the other margins this parameter can be adjusted to suit the preference of the user. After all the planes are placed, the fibula is ready for cutting and can go to the next step.

Now that the bone is cut into many different pieces, the correct piece has to be selected to be placed in the correct location of the resected mandible. A small macro has been designed for this purpose called “SelectBonePiece”. The internal network of this macro can be seen in figure A.10. This macro has three inputs as well as three outputs. The first two inputs are the cut

fibula bone and the cut marker, both in WEM format. The last input is the artery as an inventor scene. The outputs are all three in WEM format. The first output is the selected piece of the fibula bone. Important to note is that this piece of fibula bone is translated so its centred at the origin of the coordinate system. The second output is the piece of the marker that accompanies the piece of the fibula bone. The third output is the selected piece of the fibula bone but this time it resides at its original location. The selection of both the piece of the fibula bone and the piece of the marker work in the same way. A WEMDemergePatches module makes every piece addressable by its own unique ID. The same is done for the marker. This module is followed by a WEMSelectPatches module. With this module, the correct piece can be selected and provided to the output. The same is also done for the marker. The selection process itself is executed in Python. The first step is to provide the number of the two planes that are used to cut the specific piece. The next step is to get the position on the z-axis that was given to both of the planes to make the cut. Now all that is left to do is to loop through all of the pieces and check which one has its top and bottom between the two provided planes. The same is done for the marker. It is also very beneficial for the user to see exactly which piece is selected for each location in the resected mandible. This is done by taking the output of the WEMDemergePatches module that demerges the fibula bone and rendering it. A regular rendering will cause everything to be grey. To add a bit of colour a tableLut module is used. In a tableLut module all the separate pieces can be given a custom colour by addressing their unique ID. This colourful fibula bone is displayed in a viewer for the user to view.

3.1.7 Placing the fibula bone in the mandible

To correctly place the pieces of the fibula bone into the resected mandible there are two different macros. One for the regular reconstruction ranging from one piece to three piece reconstructions that is called “PlaceFibulaInMandible” and a second macro for the double barrel reconstruction that is called “DoubleBarrel”. Both of the macros are nearly identical to each other except that the DoubleBarrel macro has two additional inputs. The inventor scene from the PlaceFibulaIn-Mandible macro as well as the controls for this inventor scene. Each macro deals with one layer of the reconstruction. The PlaceFibulaInMandible macro takes care of the bottom layer of the reconstruction while the DoubleBarrel macro deals with the top layer of the reconstruction. In case of a regular reconstruction there is only one layer and the PlaceFibulaInMandible macro can handle the complete reconstruction. The internal network of this macro is show in figure A.11. In the bottom there are nine buffers. The three on the left are for the pieces of the fibula bone in their original position. These pieces are used to calculate, in the Calculator module, as precise as possible the center of the pieces. The three buffers in the middle are for the pieces of the fibula bone that have their center in the origin of the coordinate system. The three buffers on the right are for the pieces of the marker. The modules in the middle are for manipulating the piece that is selected. These modules take the piece and apply rotations and translations to position it in the correct location in the resected mandible. Besides these transformations the user can also apply a set of tranformations to position the piece more accurately where it should be placed. There are three more buffers besides the nine in the bottom to store the manipulated piece for further display when it is not selected. There are four groups of modules to perform the user defined transformations. The user can perform a rotation along the longitudinal axis of the piece. Then there are also three translations possible for the selected piece: along the longitudinal axis of the piece; up and down and towards or away from the inside. To calculate

the translation needed in every direction formula 3.1 and 3.2 are used.

Everything has to be flexible in positioning, so the breakpoints and the attachment points on the mandible can be adjusted as well. The breakpoints can be moved freely in all dimensions. The default dragger to move these markers is very unhandy to use. To solve this a SoTransform-BoxDragger is attached to the selected marker. This dragger is a cube that is wrapped around the marker. Selecting the sides of the cube allows the user to move it inside the selected plane. Now all there is left to do is to synchronise the position of the dragger with the position of the marker. The attachment points on the mandible have a restriction to this movement, they are only allowed to move on the cutting plan to avoid creating a large gap or that the point by accident ends up inside the mandible. For each piece there is an arrow between the beginning and end point that indicates the flow of the artery. A 3D measurement tool is also built using a So3DMarkerEditor module. MevisLab has no build-in tools to measure inside an inventor scene. A So3DMarkerEditor allows to place markers on any surface in an inventor scene. Each of these markers can have a vector who's endpoint can also be placed anywhere on any surface in an inventor scene. In the appearance setting of this module there is the option to display the length of the vector. This way it is possible to measure distances in an inventor scene. To export a completed reconstruction to STL format there are nine WEMSave modules. Those are the six blue modules at the top of the network and the three blue modules in the bottom left. These modules take a WEM and save it into an STL file. The complete reconstruction is exported as well as the complete fibula bone. Furthermore all the pieces of the fibula bone are exported to separate STL files. Also the mandible is exported to two different files, one for the piece that has to be removed and one for the rest that has to stay. All of the planes in the cuttingplan are exported to one single STL file. Furthermore in the LoadFibula macro there are also two more WemSave modules. These are used to export the complete fibula bone as well as the artery.

3.2 Graphical user interface

The user interface is built using the MeVisLab Definition Language (MDL). In order to follow the flow that is used for creating a mandible reconstruction, the interface is separated into six different steps using a wizard design pattern.

3.2.1 Step 1: Load image

The first of these six steps is to load the desired images for the reconstruction into the program. The view that the user is greeted with when opening the program can be seen in figure B.1 a. To the far left is a step list and the current step is indicated in bold. At the top there is an instructions box containing the instructions needed to know how to correctly complete the first step. Below the instructions box are four string fields to provide the different file paths to the different files. Next to each string field is a browse button that opens up the file explorer so that the selection of the required file is easily done. Even further to the right there is a progress bar for every string field. The progress of the loading of the file is displayed using this progress bar. These string fields are connected to the WemLoad modules in the LoadMandible and LoadFibula macro. Below these string fields is a checkbox to indicate if the user wants to use a manual marker instead of the automatic marker to mark the artery side of the fibula bone.

And below this checkbox are two viewers, the viewer on the left displays the loaded mandible and the viewer on the right displays the fibula and artery as well as the marker on the fibula bone. The left viewer, where the mandible is shown, is also where the three points are placed on the bottom of the mandible to align the tangent plane of the mandible to the xy-plane. The left viewer is linked to the SoExaminerViewer that can be found in the LoadMandible macro. The viewer on the right is of the same type and is located in the DrawArtery macro. Completely in the bottom is a button that will take the user to the next step in the reconstruction.

3.2.2 Step 2: Making the cutting plan

In this step the cutting plan is made by the user. The interface of this step is shown in figure B.2. Just as in step 1 at the top there is an instructions box to inform the user on how to operate the interface in this step. Below the instructions box there are two viewers. The viewer on the left is used to design the cutting plan and is linked to the SoExaminerViewer found in the CuttingPlanes macro. By positioning the blue indicator plane the user can indicate where the cut shall be performed in the mandible. The indicator plane can be moved in three different ways. The plane can be moved using the manipulator that is wrapped around the plane. The second method is using the fine tune controls on the bottom of the window. There are two parameters about the indicator plane that can be adjusted, the position and the rotation. These controls are two vectorInputs that are linked to respectively the SoTranslate and SoRotate modules that are connected to the blue indicator plane. The last method is by using the keyboard, the key combinations are explained in the instructions box. These keys are intercepted by SoGenericCommandAction modules, one for every key. Clicking on the button below the viewer with the title "Generate cut", the cutting plane will be fixed in place and the blue indicator plane can be moved to the next position. This locking in place is done by storing the plane in the buffer that was discussed in subsection 3.1.3. Next to the button to generate the cut there is a button to reset everything. Besides this there is also a way to remove a single plane if the user thinks its position or angle should be adapted. This is very easily done by clicking on the cutting plane that has to be altered. The fixed plane will be removed and the blue indicator plane will take its position and angle. There is a checkbox that allows the user to toggle the OrthoView2D on and off. This OrthView2D module is located in the DicomViewer macro. The viewer will be displayed on the right and the cuttingplan is displayed onto the DICOM image as an overlay. Furthermore, there are also three checkboxes to overlay a 2D slice of the DICOM data in the 3D view of the cuttingplan. For every orientation the user can move through the different slices of the DICOM data in the right viewer by using the scroll wheel and the visible slice is then also displayed in the 3D view. After all the cutting planes are placed correctly, the button below the right window can be pressed to cut the mandible. The result of this cutting will be displayed in the viewer on the right. This viewer is linked to the SoExaminerViewer in the SelectPartToWorkWith macro. In this right viewer the piece of the mandible that has to be removed can be selected here by clicking on the piece. After this has been done, step 2 is complete and the user can proceed to the next step.

3.2.3 Step 3: Calculate the contour

Calculating the contour is a straightforward step and does not require any instructions. There are only three buttons. One to go to the next step, one to go to the previous step and one to

calculate the contour. Pressing the last button will generate the bottom line and the centre line of the contour. This is done by executing a Python function. The details of this function are explained in subsection 3.1.4. These are then displayed in the viewer so that the user can verify that the contour is correct. When the user in step 1 forgot to place the three points on the bottom of the mandible the bottom line will most likely not span to back of the mandible. The result before and after pressing the "Generate contour" button can be seen in figure B.3.

3.2.4 Step 4: Make the reconstruction plan

The next step is to use the contour line to create an initial fit of the centreline of the fibula pieces. Multiple smaller steps are used to accomplish this. Figure B.4 shows the first screen of the planner. On the left is a menu with all steps. On the right is the centreline. The black points are the contour of the complete mandible, the green points is the piece that is about to be reconstructed. Next, the options in the menu will be explained. It is expected that the user goes through the menu from top to bottom.

The first step is the "Split data". Here the user can choose the orientation of the reconstruction. There is the option between horizontal, vertical and both horizontal and vertical. The vertical option is only necessary when there is a vertical piece used in the reconstruction. An example of the user interface and the visual representation of the plan is given in figure B.5. Next, the points will be projected on the planes and the user can go to the next step: "Horizontal reconstruction". This step is only used when there are horizontal pieces in the reconstruction. In the case of only a vertical piece, this step can be skipped. The user can choose one to three pieces, then the software will give an initial fit. The user can than drag the points on the plot to correct the software. Figure B.6 shows this step. Finally, the "Final result" option in the menu shows the final result of the planning in 3D. In the next step, the lines shown will be served as centrelines for the reconstruction. The user can at any time return to previous steps to make adjustments. Figure B.7 shows the final result.

3.2.5 Step 5: Cut and place fibula

This is one of two possible endings. Step 5 is the ending that is used when the user desires a reconstruction of only one single layer. A complete reconstruction of this type can be seen in figure B.8. The instructions on how to adapt the automatic reconstruction are given inside the instructions box. Here are also two different viewers. The viewer on the left is linked to the PlaceFibulaInMandible macro and displays the reconstruction of the mandible where all the pieces can be manipulated. These manipulations are rotating the pieces along its longitudinal axis to assure that the correct side is facing the anterior of the mandible. The breakpoints and the points attached to the mandible, displayed in blue, can be moved by using the mouse. When one or more pieces have been moved, the angles do not line up anymore between the different pieces or with the mandible. To re-cut the fibula pieces the user can simply hit the enter button on the keyboard. The viewer on the right side displays the fibula bone. The pieces are colour coded the same way as in the reconstruction viewer on the left. The artery is displayed as well and this allows the user to verify that the fibula bone is cut in the correct location and that the orientation of the pieces with respect to vascular direction is correct.

Below the viewers are a few more parameters that can be tuned to the user's needs. The margin

to the foot side of the fibula to the first cut can be set. A typical value for this margin is 70 mm. A second parameter that can be set is the margin between the pieces on the fibula bone. This is typically 2 mm and is needed for surgical purposes. The parameter on the right should not be used, but can be in case of a failure. With this parameter, every pieces can be given a separate offset in length. Now the user can adapt the length of all the different pieces individually. One row lower are the same controls for the OrthoView2D and the DICOM slice overlays as seen in step 2. The only difference here is that instead of the cuttingplan, the individual pieces are shown as overlays on the 2D view in their respective colours. This OrthoView2D is the same viewer that is used in step 2, only with different overlays. Below these parameters there is a combo box to select the different pieces. To the right of this combo box are four check boxes. The first two show or hide the complete or resected part of the mandible respectively. This is done by using SoBypass modules to disconnect the sub scenes from the viewer. The third checkbox allows to rotate all pieces on the same layer at the same time. Checkbox number four lets the user flip the vascular direction of the pieces. It is important to determine the correct flow of the artery so it can be attached correctly. Checkbox number three and four are not linked to any MeVisLab code but are used as Boolean inputs for the Python code. The last checkbox enables or disables the measurements. This checkbox is linked directly to the on-off checkbox of the So3DMarkerEditor module used for measuring as discussed in subsection 3.1.7. When this checkbox is checked the user can measure on any surface in the left viewer using the mouse.

3.2.6 Step 6: Double barrel

The alternate ending of a mandible reconstruction is a double barrel reconstruction, where two pieces of fibula are stacked on top of one another. The interface of step 6 and step 5 is nearly identical, there is only one major difference. This difference is the extra parameter in the bottom to set the margin between the layers of the double barrel on the fibula bone because there has to be enough soft tissue to flip everything over. The typical value for this margin is 15 mm. A complete double barrel reconstruction can be seen in figure B.9.

3.2.7 Export and save

When the reconstruction is complete and the user is satisfied with the end result, he can export it. There is a second tab in the top that brings the user to the save and reload interface that can be seen in figure B.10. There are two options, the first one is the save and reload. The user can save the complete project to a directory of choice when using this option. All the parameters from the modules are saved in a text file using the SettingsManager module. Furthermore all of the Python parameters are also saved, this is done using JSON files. When there is a previous project stored in the selected directory, it will be overwritten. When the load button is pressed, the project that is saved in the selected directory will be loaded. The software will in that case automatically move to step 5 or step 6 depending on the choice if a regular or double barrel reconstruction was saved respectively. The user is fully able to move back through the steps and adapt the planning if necessary. Normally the project is only saved to be reviewed later, so adapting the planning is usually not done. The second option allows to export the necessary STL files, as explained in chapter 4.1.7, so they can be provided for the next step in the planning.

3.3 Maintainability

Another important aspect is the maintainability of the software. In the future, MeVisLab can update. This can cause unexpected bugs in the software because some modules and functions can be changed. This problem has been kept in mind while designing the software. The project contains multiple (macro) modules that are independent of each other. So when one module fails, it can be replaced with a new one. This does not only eases maintainability, but also allows future developers to make adjustments to the software. When a better solution is found for one of the reconstruction steps, that module can be switched out for a better one. The file structure also follows this pattern. Each macro module contains a *.mlab file, *.script file and python file when necessary. This keeps structure in the project during development.

Chapter 4

Results and evaluation

This chapter will discuss the results of the project as well as evaluate the quality of the presurgical planning using the new software. These topics will be discussed based on a planning performed under the guidance of Dr. Ing. Sun Yi. Evaluating the performance of the software in terms of speed is not performed because the speed is determined significantly by the hardware it is deployed on. It is also important to notice that due to time restrictions and busy weeks at the hospital, the evaluation of the software is rather brief. Most of the evaluation was done on weekly basis over Skype meetings. Extensive testing by experts after the software was completed is not done. So it is hard to decide if the software performs better and shortens the reconstruction time. However, some evaluation could be done.

4.1 Ease of use

The user experience is an entirely subjective matter and differs from user to user. There are however general aspects that define a good user experience. One of those aspects that is used is the wizard design pattern. This design pattern insures that no steps are skipped and provides an overview where the user is located in the reconstruction process. Another advantage of the wizard design pattern is that the complex reconstruction is broken down into small and easy to understand steps. Since every step has instructions at the top of the page, the learning curve is not very steep and an inexperienced user can learn to use the software with a minimum of training. Dr. Ing. Sun Yi pointed out when first using the software, that the buttons used in step 4 have confusing names and that the steps within step 4 are quite complicated and could be made simpler. It is also important to point out that the interface is a bit crowded. This is not necessarily an issue. Using a menu approach would unclutter this, but this was not implemented due to timing constraints. One thing to look out for when implementing menus is that to many little menu's can annoy and disorient the user.

4.2 Quality

The quality of the reconstruction is reviewed by Dr. Ing. Sun Yi. During the weekly Skype meetings all the functions where discussed that where implemented since our last meeting. To properly demonstrate how these functions will work in the operational environment a reconstruction with guidance of Dr. Ing. Sun Yi was performed. During this reconstruction he would

approve if everything was working correctly or if any issues needed to be addressed. Some of those reconstructions where also exported to STL files and sent to Dr. Ing. Sun Yi for verification. He was able to verify that the finished software can provide a high enough quality for the reconstruction so that it can be used in surgical planning.

4.3 Future work

In the future the software has to be updated to add new features or to improve already existing features. The possibility for a four piece reconstruction can be added. A four piece reconstruction is not used very often but it will be useful if this type of reconstruction can be handled by the software as well. During the testing of the software a suggestion to improve was already made. Simplifying the interface was the suggestion that came up. More concretely, this means making the interface less crowded like discussed in chapter 5.1 as well as redesigning step 4.

Chapter 5

Conclusion

The main goal of this thesis was to build new reconstruction planner software that could replace the current program. A couple of requirements where lined up as discussed in chapter 2. Most importantly, the new software has to be semi automatic and reduce planning time without losing quality. The general approach to the reconstruction is to split the reconstruction problem into smaller problems that are easier to solve by the software and user. The new software is programmed in the MeVisLab environment and uses a wizard design pattern that guides the user trough all the steps. The results look promising. The software is able to perform one, two and three segment reconstruction, including a double barrel approach. The expert from the Hospital of Leuven confirms that the quality of the reconstructions are high enough to use the software on patients. Unfortunately, due to time limitations and busy weeks at the hospital, evaluation of the software is rather brief. Extensive testing is not done. However, some testing has been performed. The user interface allows for a fast guided workflow but is not ideal and can be confusing for inexperienced users. The primary task of future work is to improve the user interface followed by adding new functions and improving the existing ones. The current version of the new reconstruction software is a good base for future development and improvements.

Literature

[1] "Polygon Mesh." Mar. 29, 2020 [Online], Available: https://en.wikipedia.org/wiki/Polygon_mesh#/media/File:Mesh_overview.svg, [Accessed: May 15, 2020].

[2] "Getting Started." Mar. 5, 2020 [Online], Available: https://mevislabdownloads.mevis.de/docs/current/MeVisLab/Resources/Documentation/Publish/SDK/GettingStarted/index.html, [Accessed: May 14, 2020].

[3] H. Kim, T.-G. Son, H. Cho, E. Shim, B.-Y. Hwang, J.-W. Lee, and Y. Kim, "Automated maxillofacial reconstruction software: development and evaluation," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 8, no. 2, pp. 115–125, 2020. https://doi.org/10.1080/21681163.2019.1608308.

[4] "Introduction of an algorithm for planning of autologous fibular transfer in mandibular reconstruction based on individual bone curvatures." Feb. 2018 [Online], Available: https://www.researchgate.net/publication/323053551_Introduction_of_an_algorithm_for_planning_of_autologous_fibular_transfer_in_mandibular_reconstruction_based_on_individual_bone_curvatures.

[5] "MeVisLab: About MeVisLab." Mar. 5, 2020 [Online], Available: https://www.mevislab.de/mevislab, [Accessed: May 14, 2020].

[6] "Computer Assisted Surgical Planning." 2018 [Online], Available: http://depot.lias.be.kuleuven.ezproxy.kuleuven.be/delivery/DeliveryManagerServlet?dps_pid=IE10649537.

[7] "Use of polyhydra in computer vision." May 1975 [Online], Available: https://web.archive.org/web/20050829135758/http://www.baumgart.org/winged-edge/winged-edge.html, [Accessed: May 15, 2020].

[8] "A Mesh Data Structure for Rendering and Subdivision." [Online], Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.3444, [Accessed: May 15, 2020].

[9] J. Wernecke *et al.*, *The Inventor mentor: programming object-oriented 3D graphics with Open Inventor, release 2*, vol. 1. Citeseer, 1994.

[10] R. M. Lewitt, "Alternatives to voxels for image representation in iterative reconstruction algorithms," *Physics in Medicine & Biology*, vol. 37, no. 3, p. 705, 1992.

[11] U. Schneider, E. Pedroni, and A. Lomax, "The calibration of ct hounsfield units for radiotherapy treatment planning," *Physics in Medicine & Biology*, vol. 41, no. 1, p. 111, 1996.

[12] "STL (STereoLithography) File Format Family." Sep. 12, 2019 [Online], Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml, [Accessed: May 14, 2020].

[13] "1 Scope and Field of Application." [Online], Available: http://dicom.nema.org/medical/dicom/current/output/chtml/part01/chapter_1.html#sect_1.1, [Accessed: May 14, 2020].

[14] "(PDF) Automated Planning with Multivariate Shape Descriptors for Fibular Transfer in Mandibular Reconstruction." Oct. 2016 [Online], Availble: https://www.researchgate.net/publication/309475771_Automated_Planning_with_Multivariate_Shape_Descriptors_for_Fibular_Transfer_in_Mandibular_Reconstruction.

[15] "Virtual Surgical Planning for Mandibular Reconstruction With the Fibula Free Flap: A Systematic Review and Meta-analysis. - PubMed - NCBI." Jan. 2020 [Online], Available: https://www.ncbi.nlm.nih.gov/pubmed/31633539.

[16] "Developing a semi-automatic application to design a fibular cutting guide for mandibular." 2018 [Online], Available: http://depot.lias.be.kuleuven.ezproxy.kuleuven.be/delivery/DeliveryManagerServlet?dps_pid=IE11056266.

Appendix A

Appendix - MeVisLab networks

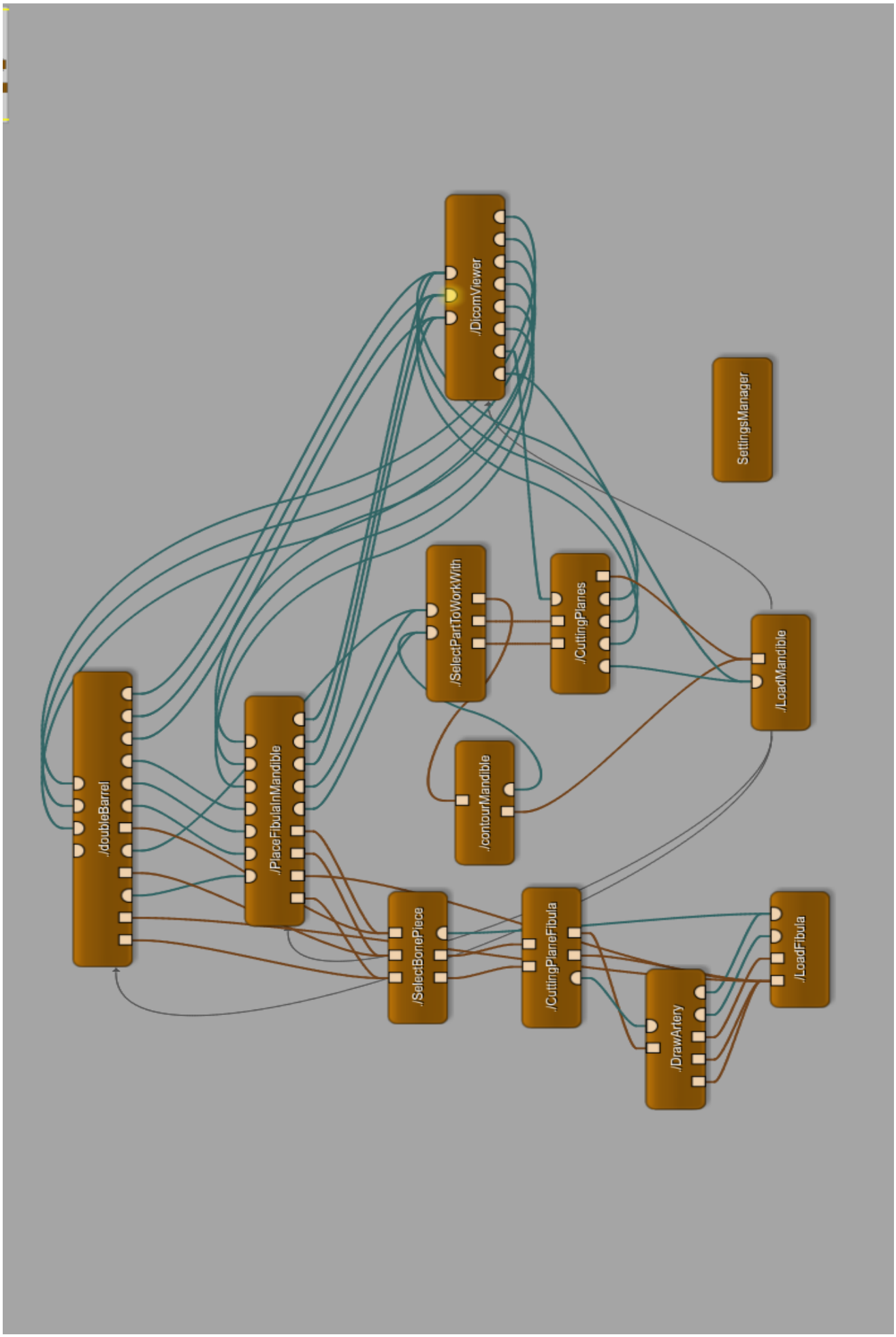


Figure A.1: The global structure of the macros

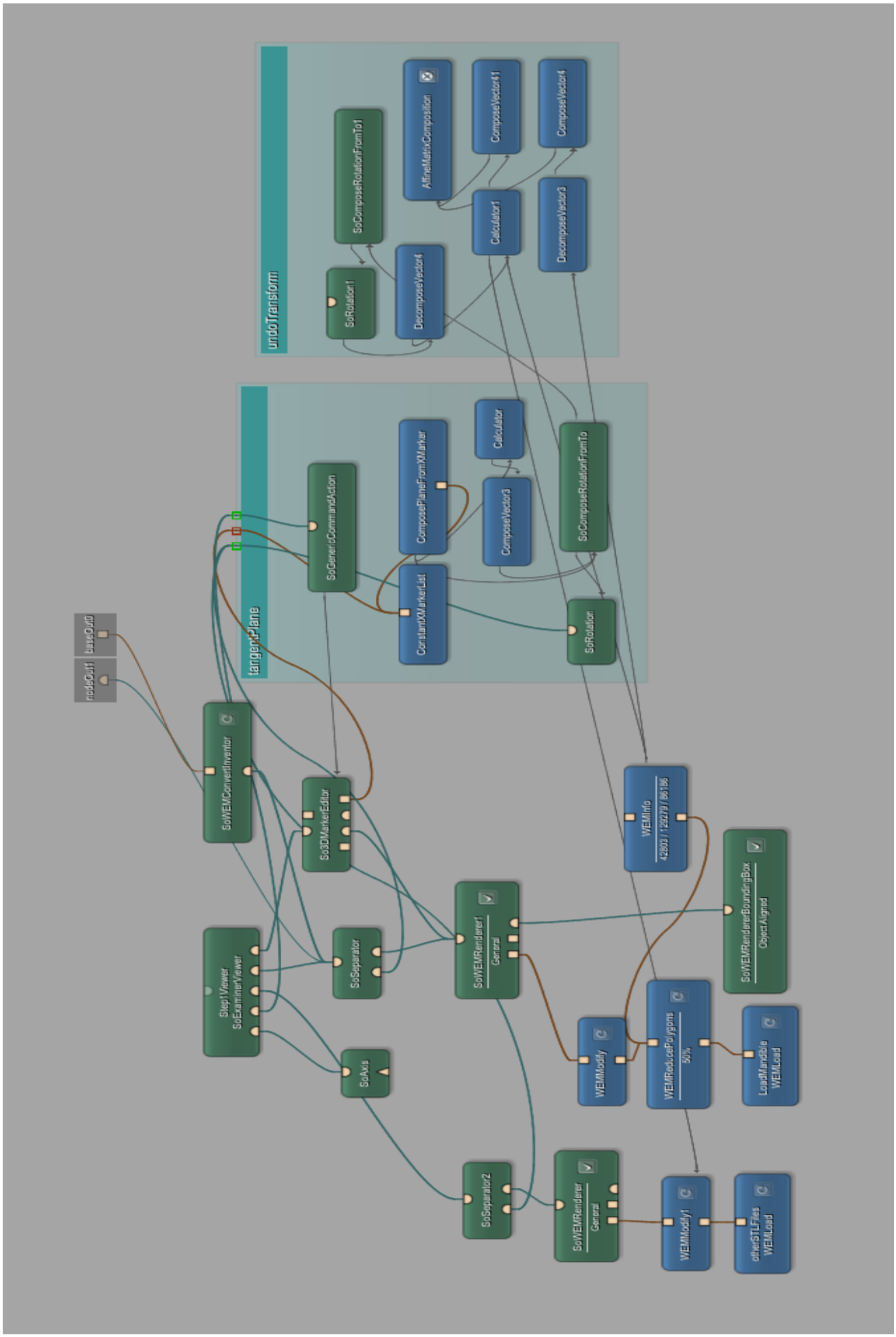


Figure A.2: The network of the LoadMandible macro

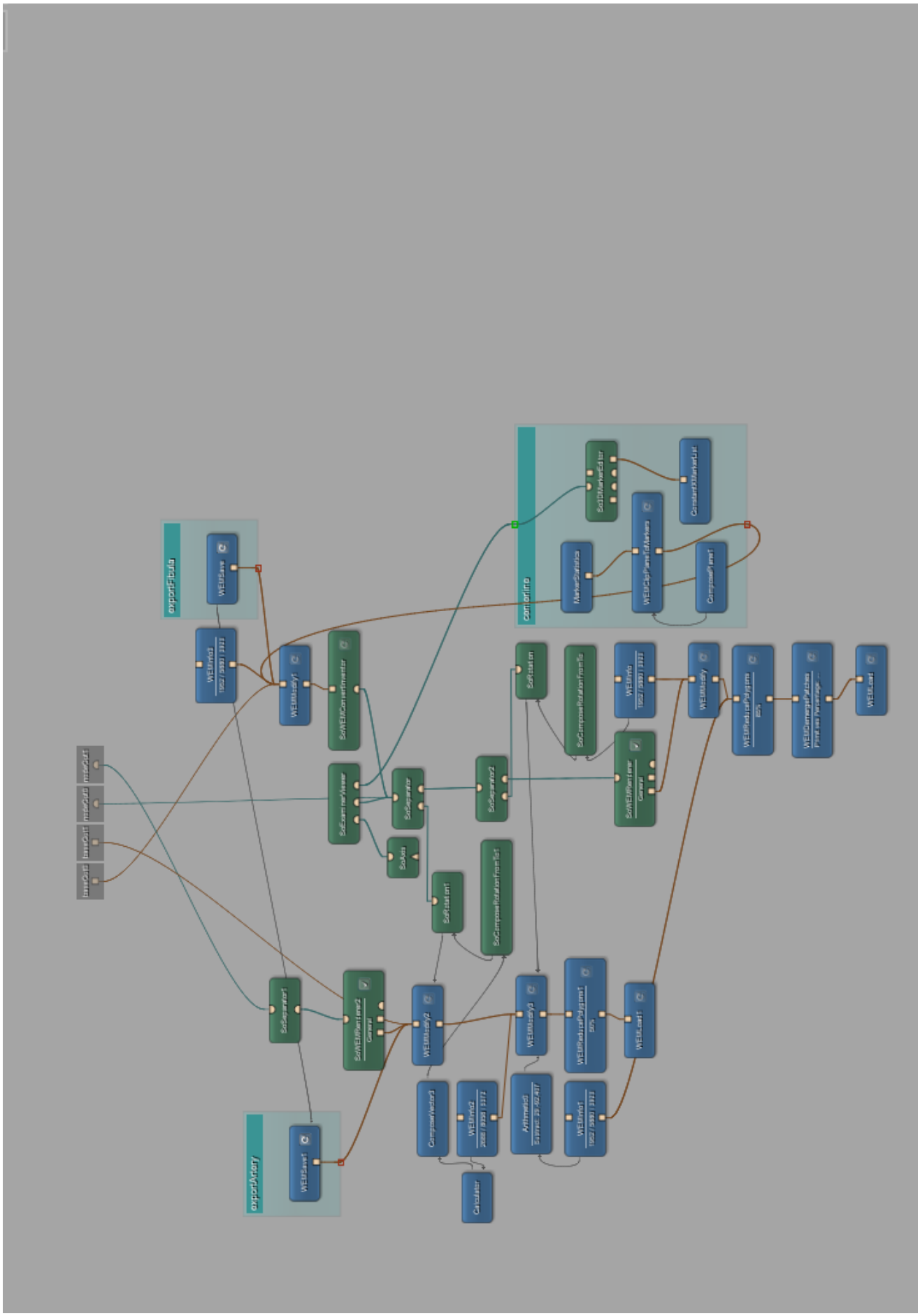


Figure A.3: The network the LoadFibula macro

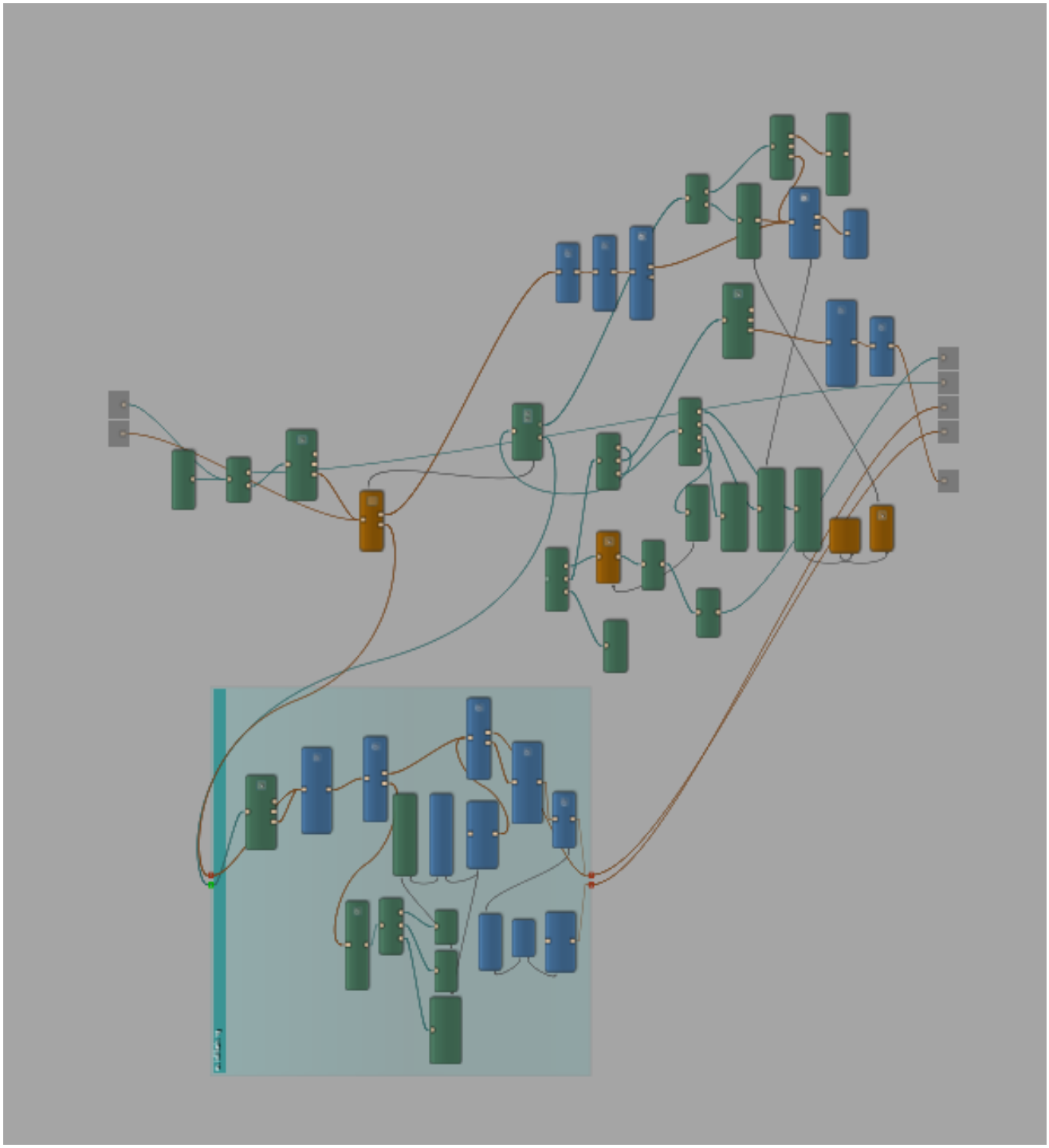


Figure A.4: The network of the DrawArtery macro

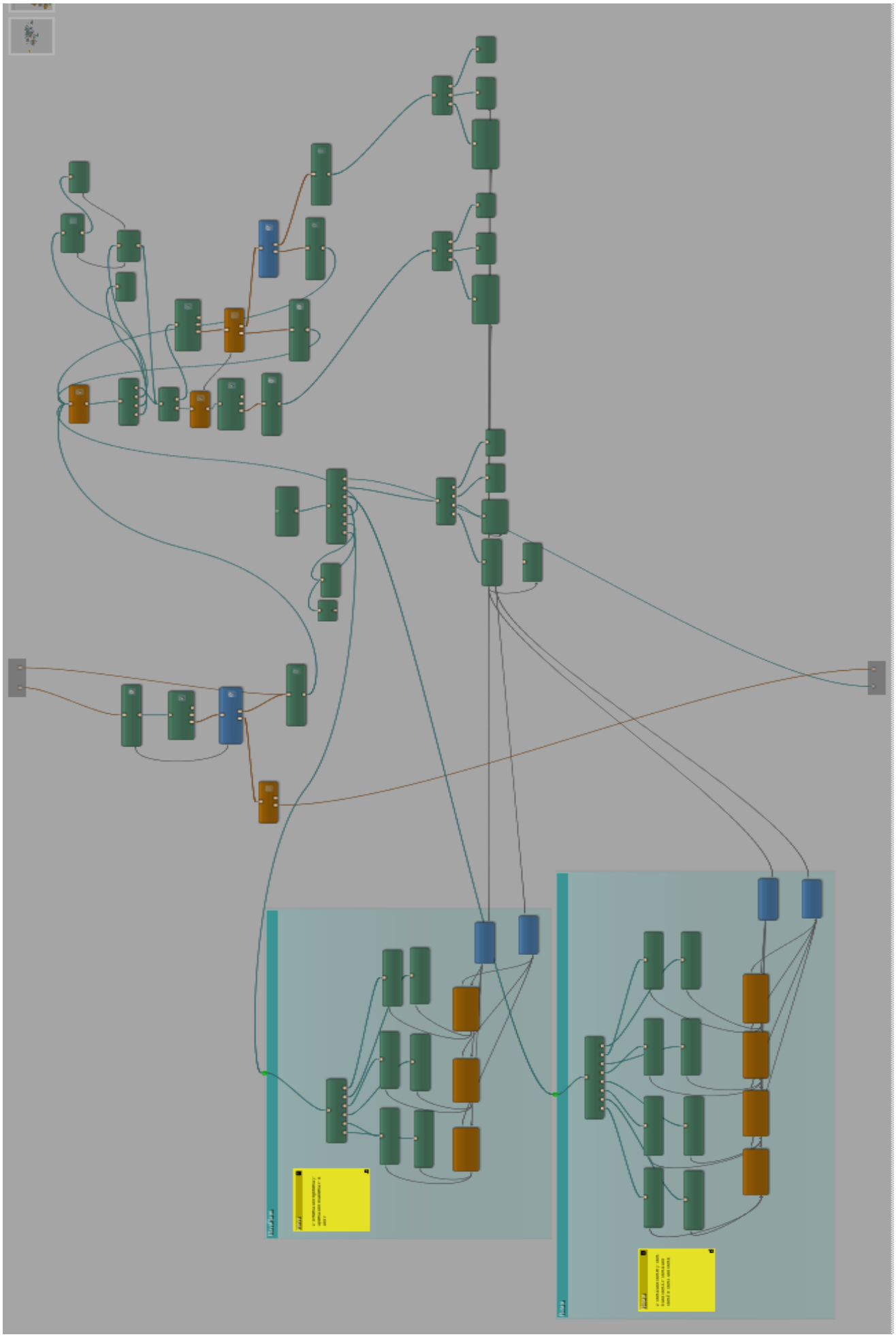


Figure A.5: The network of the CuttingPlanes macro

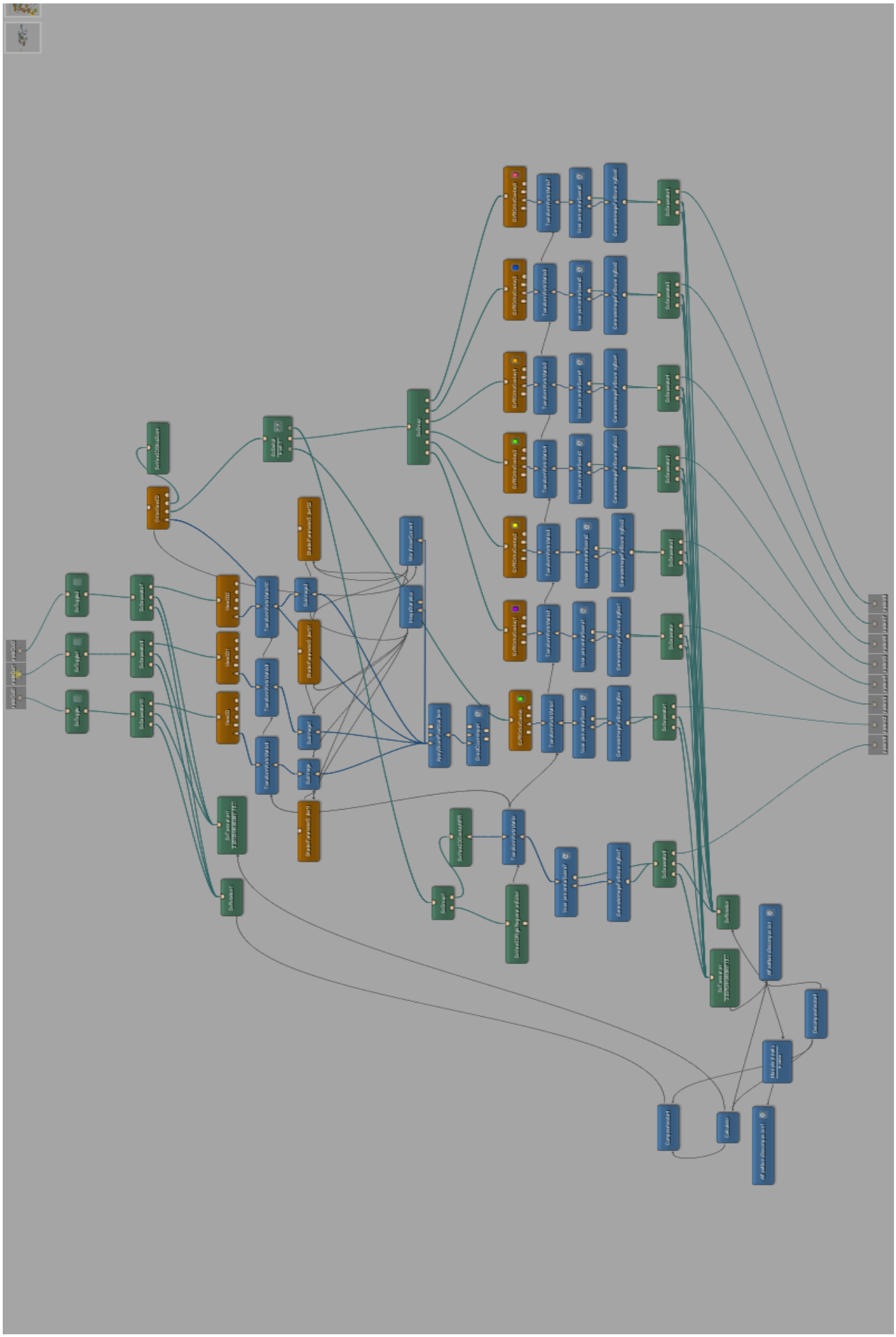


Figure A.6: The internal network of the DicomViewer macro

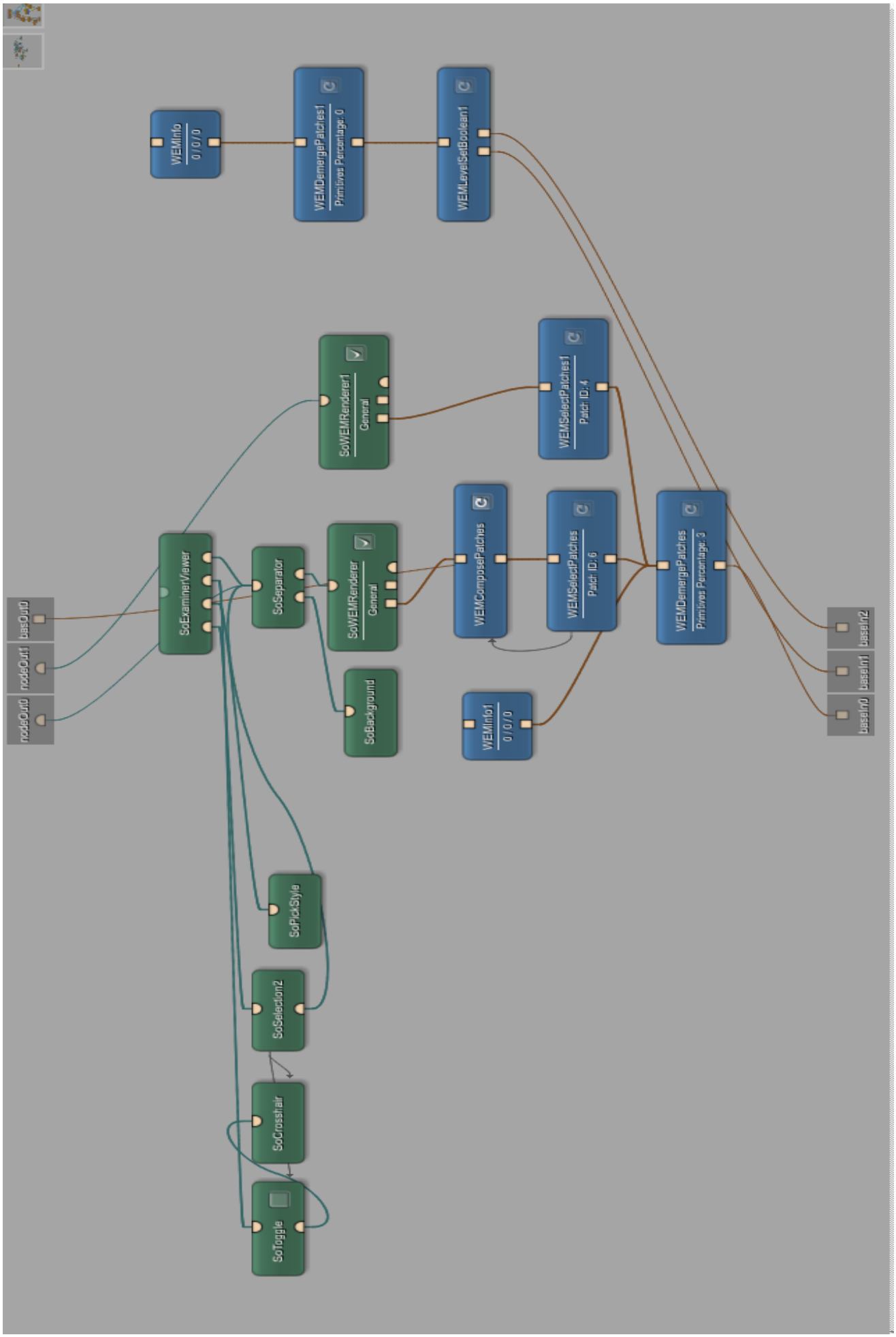


Figure A.7: The internal network of the SelectPartToWorkWith macro

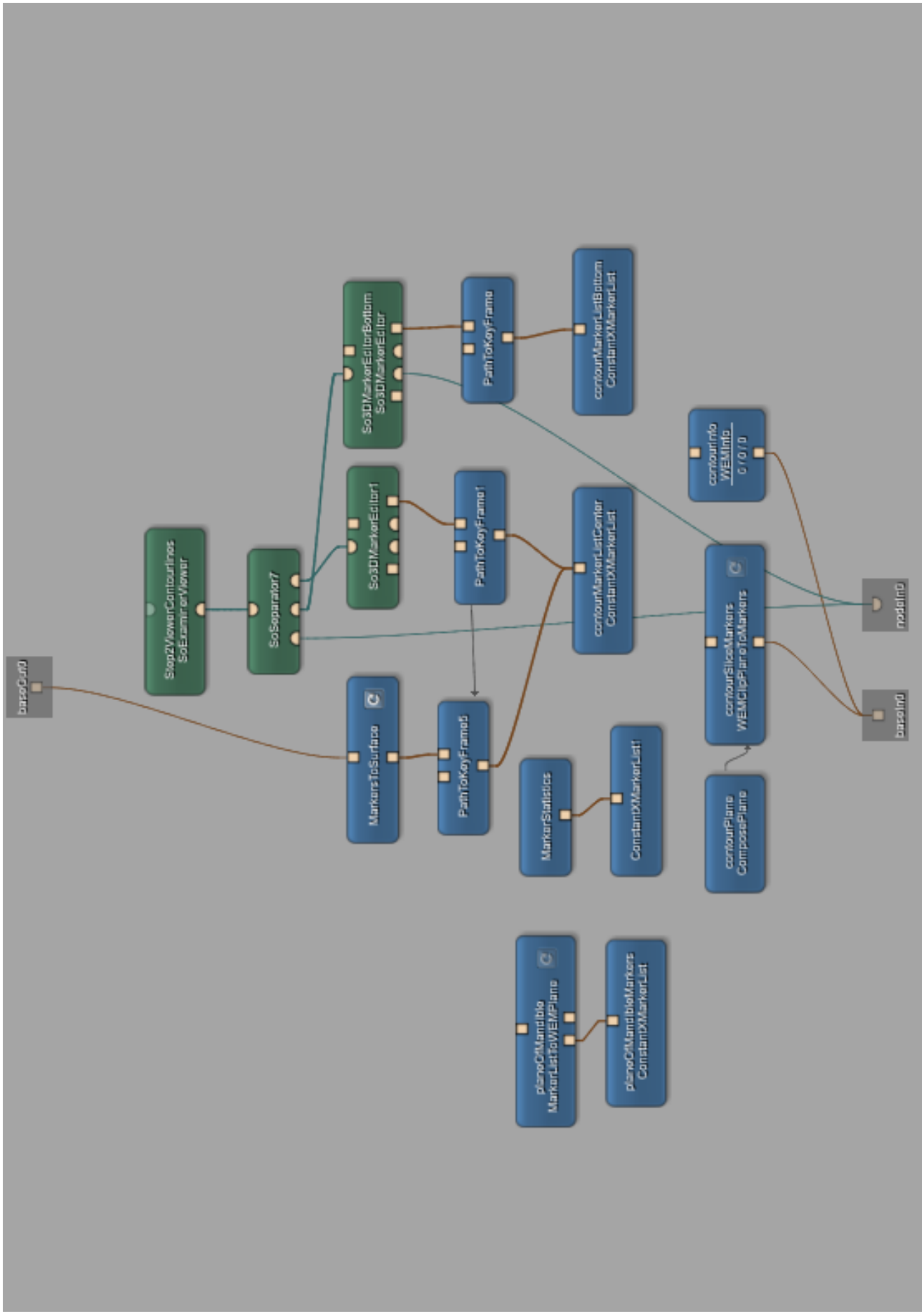


Figure A.8: The internal network of the ContourMandible macro

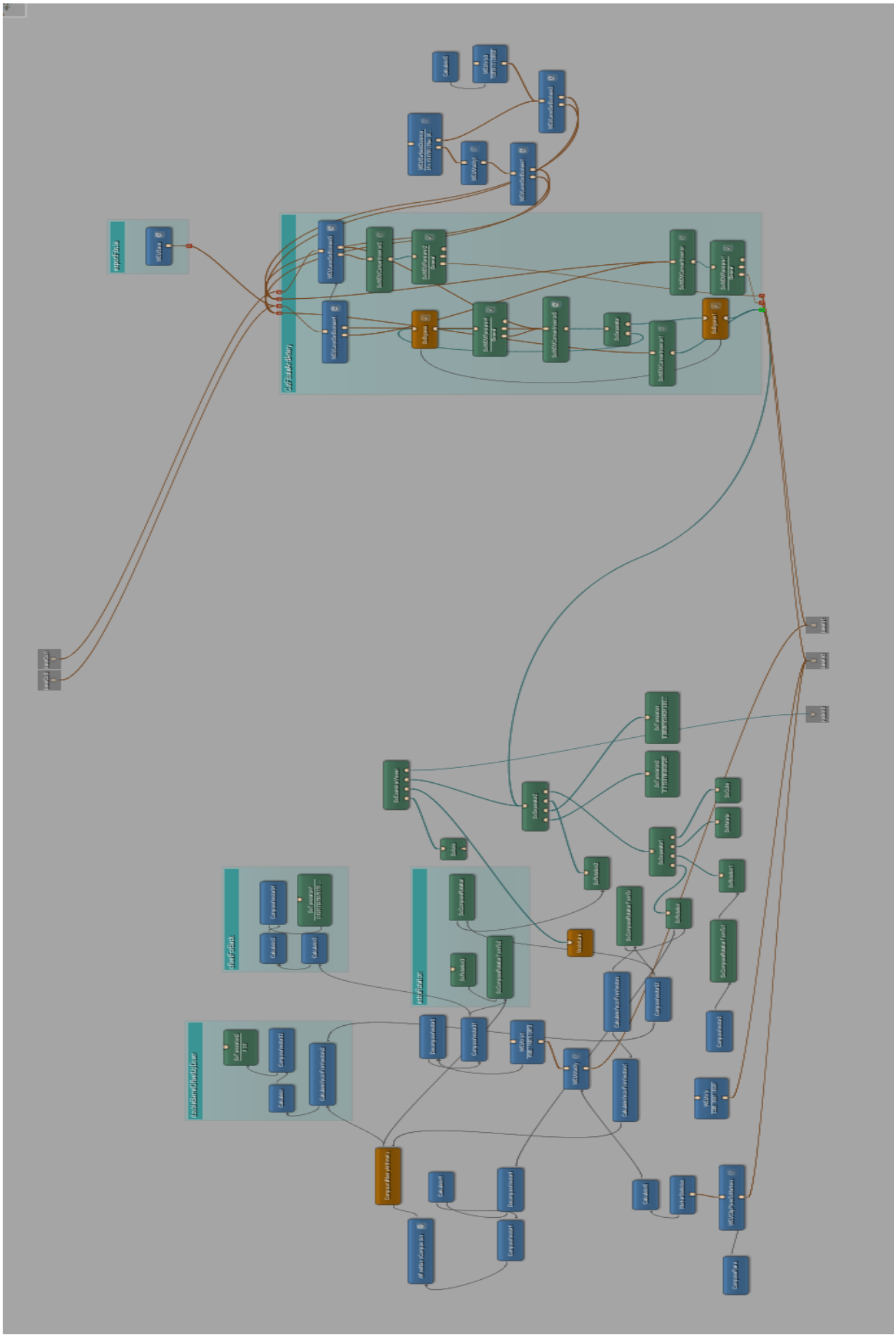


Figure A.9: The internal network of the CuttingPlaneFibula macro

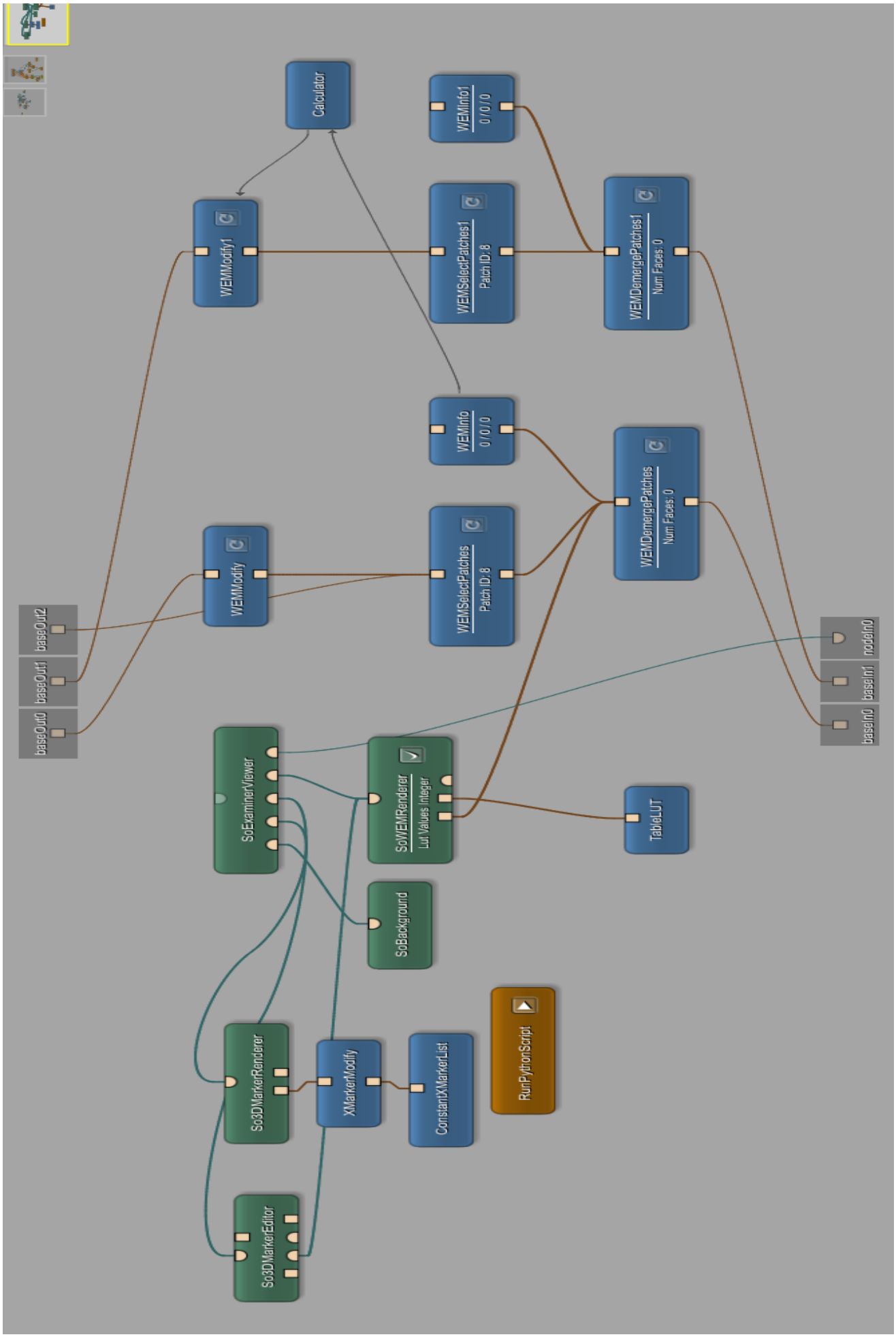


Figure A.10: The internal network of the SelectBonePiece macro

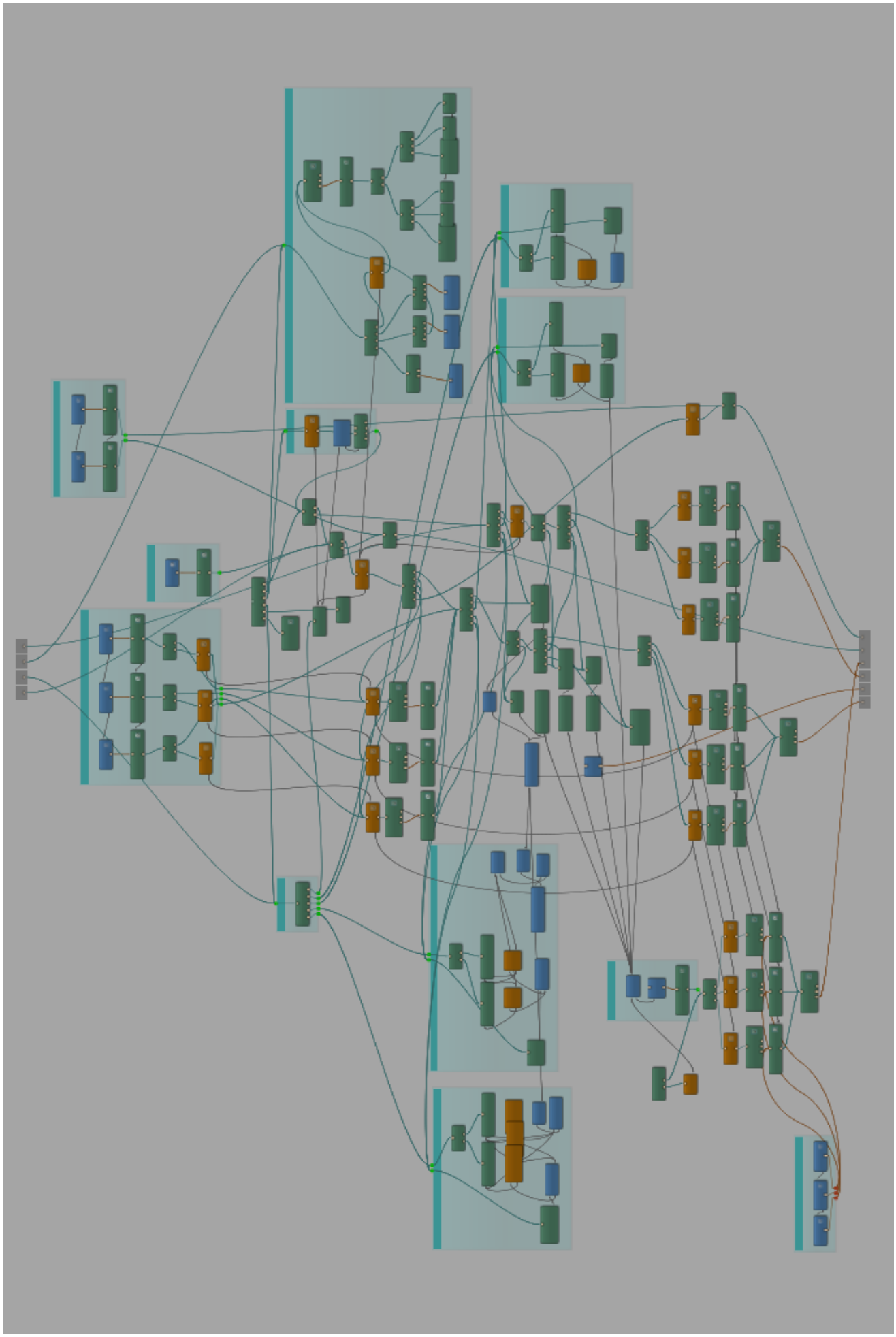
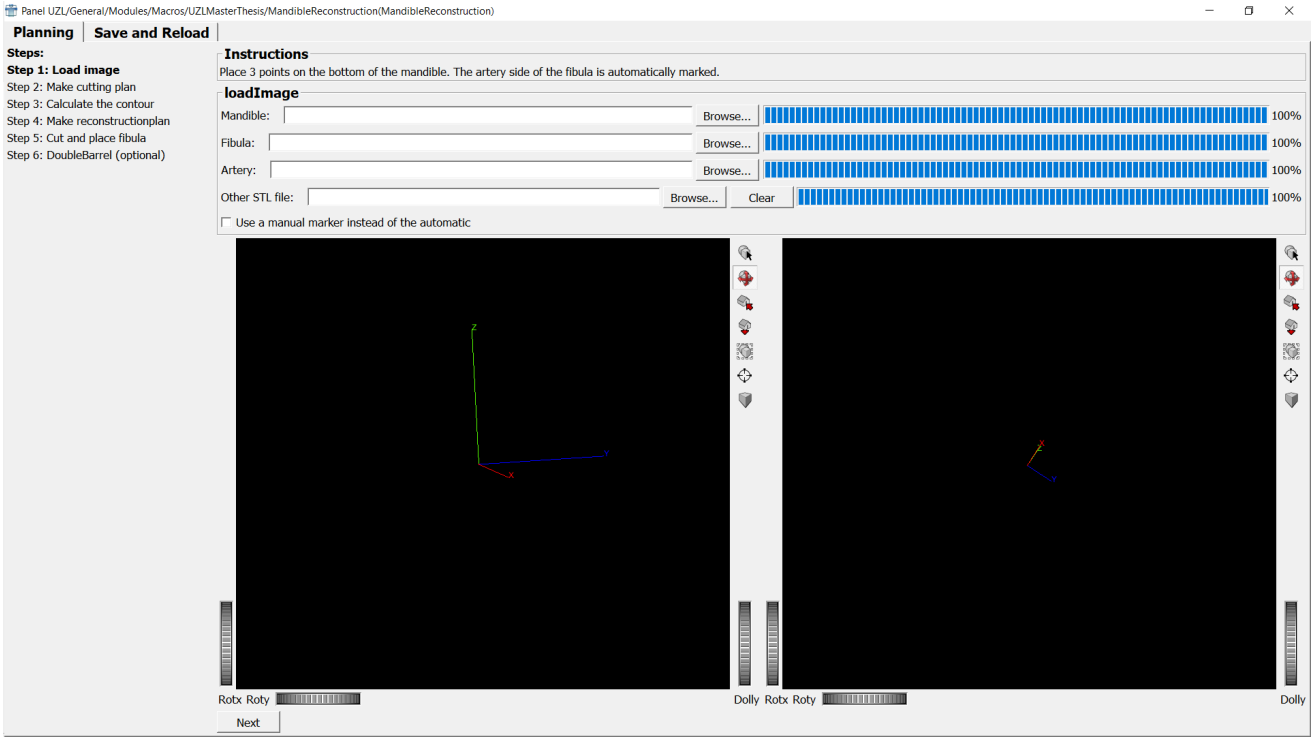


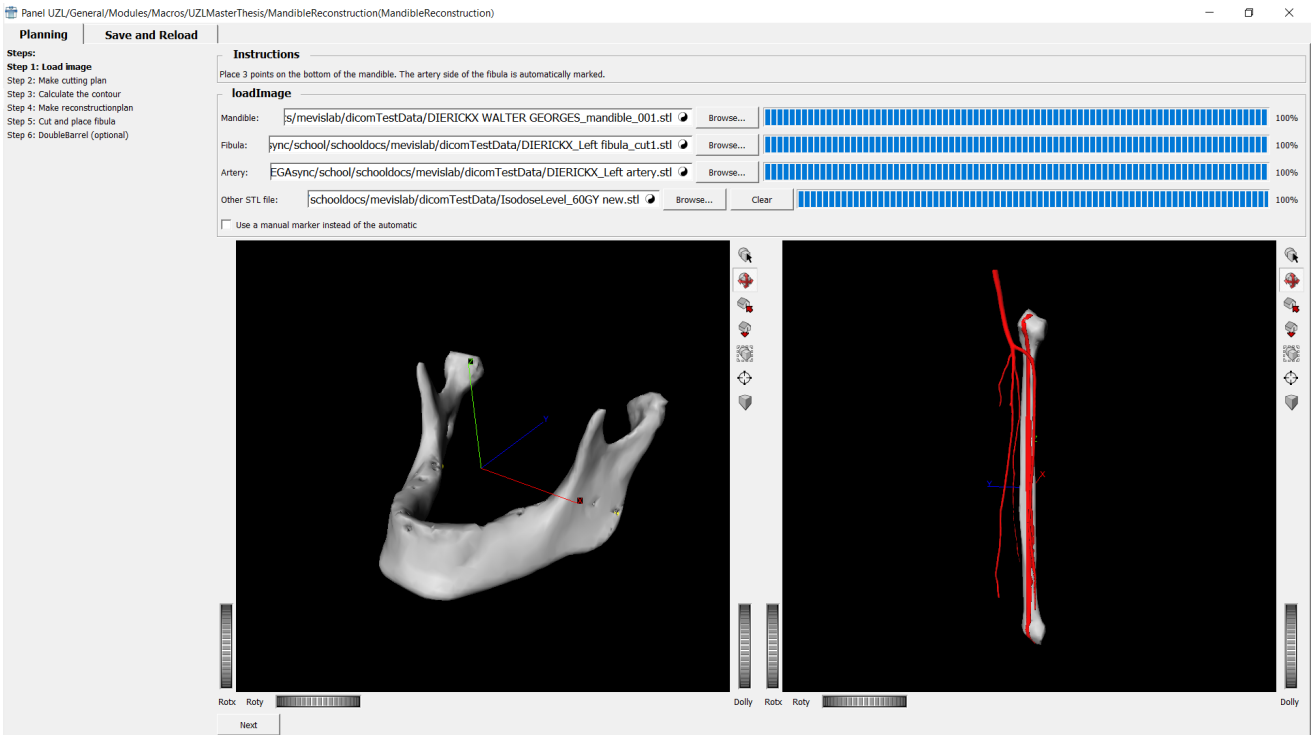
Figure A.11: The internal network of the PlaceFibulaInMandible macro

Appendix B

Appendix - Graphical user interface

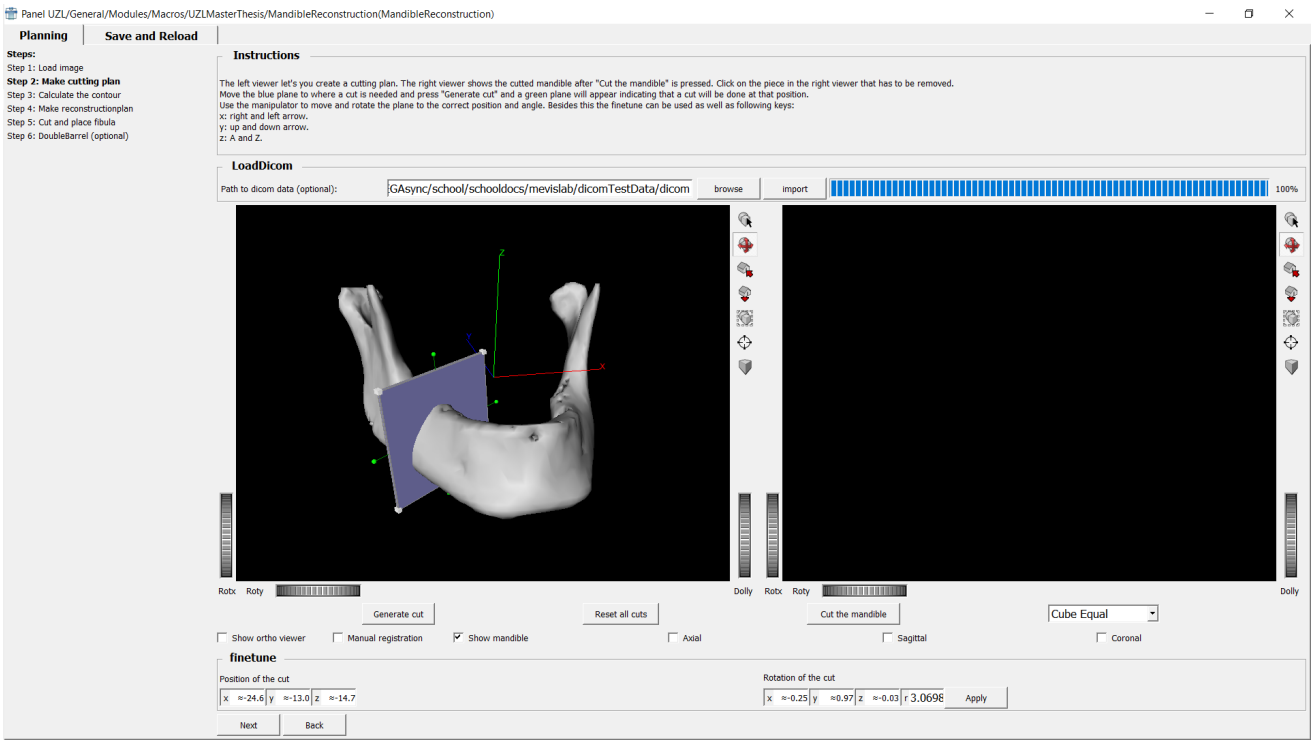


(a)

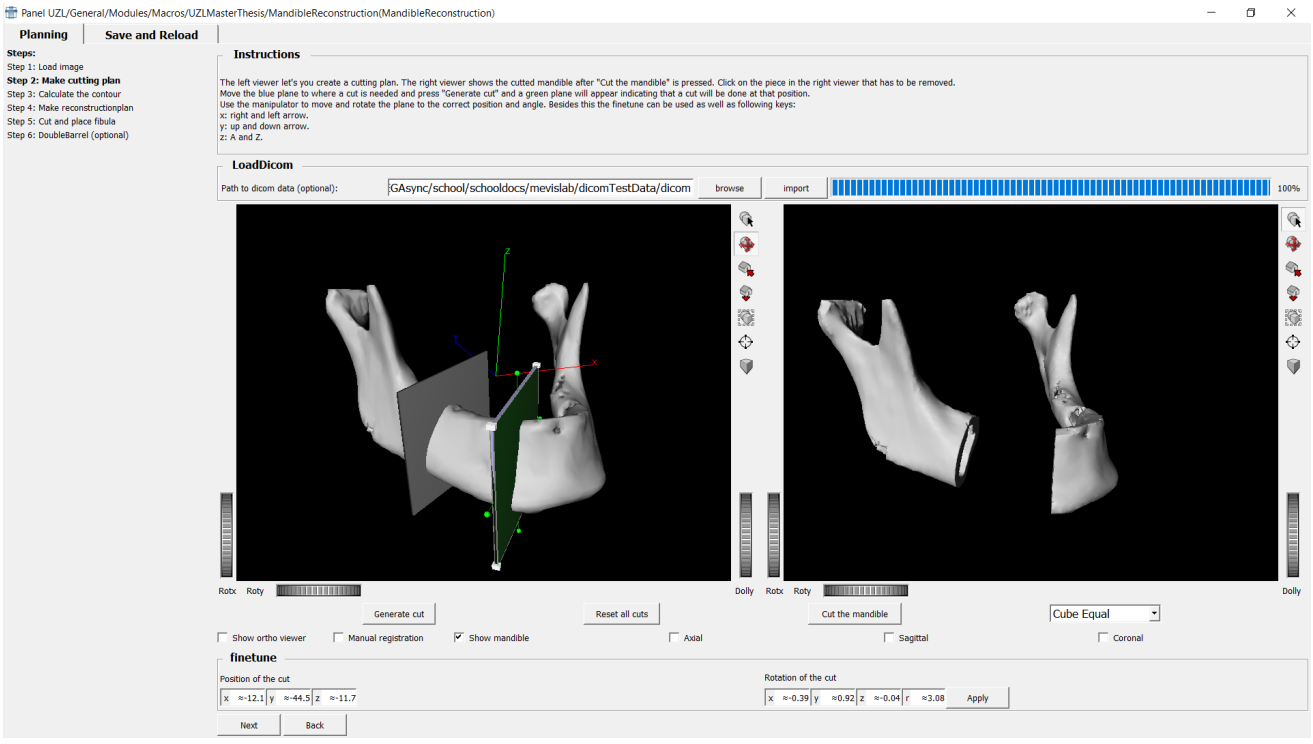


(b)

Figure B.1: The interface of step 1. (a) The initial empty interface. (b) Step 1 completed.

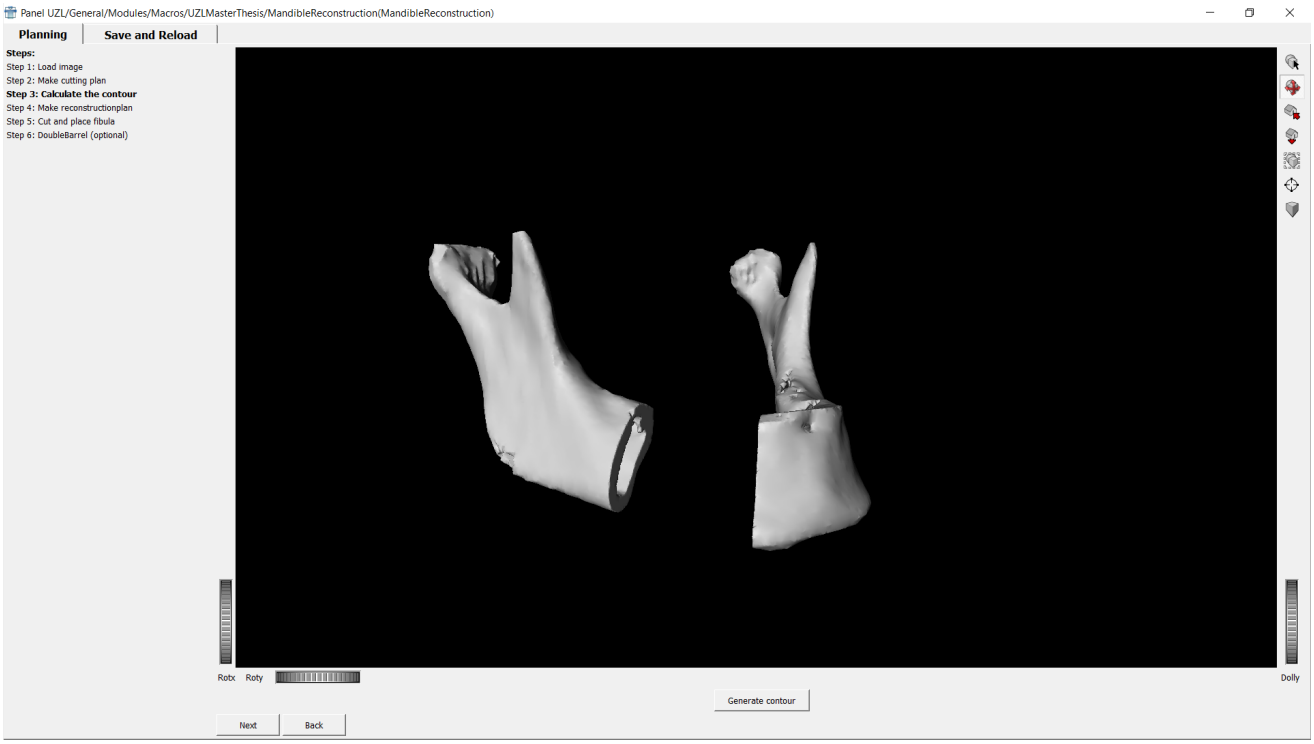


(a)



(b)

Figure B.2: The interface of step 2. (a) The initial view of step 2. (b) Step 2 completed.



(a)

(b)

Figure B.3: The interface of step 3. (a) The initial view of step 3. (b) Step 3 completed.

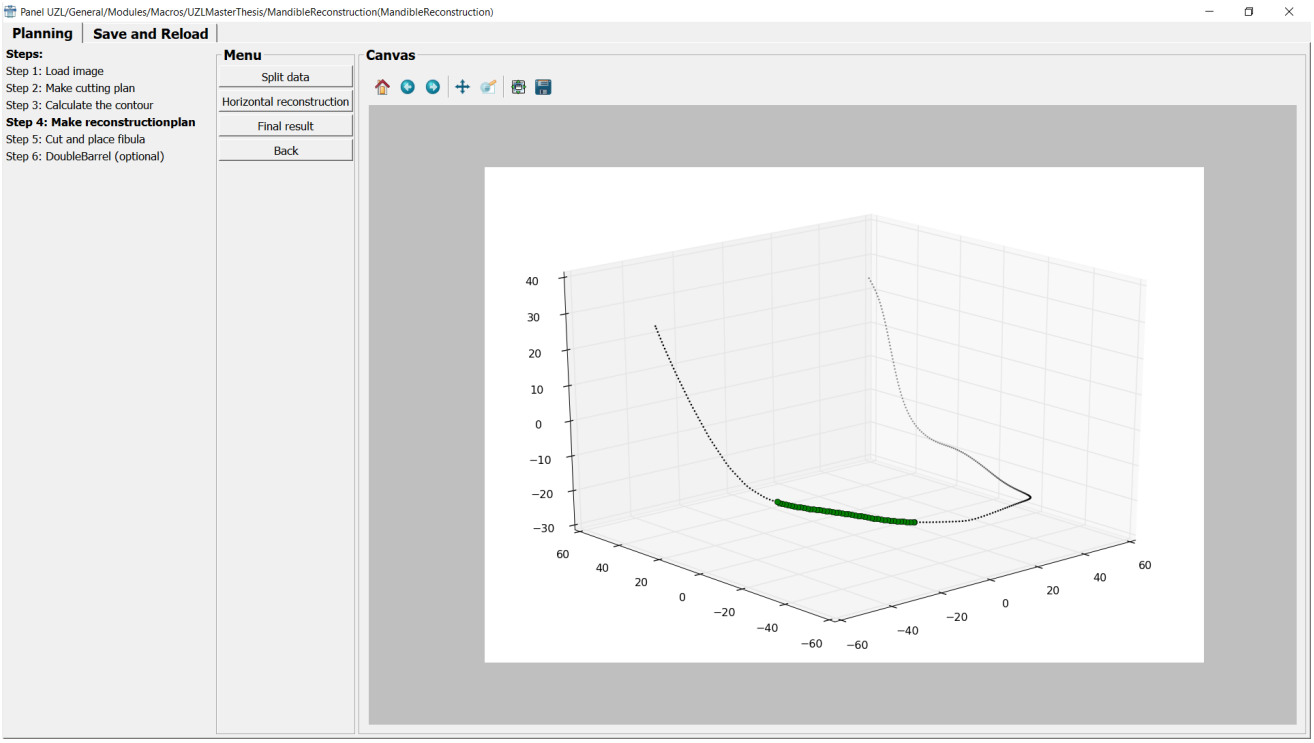


Figure B.4: The interface when starting the the planning with the contour line

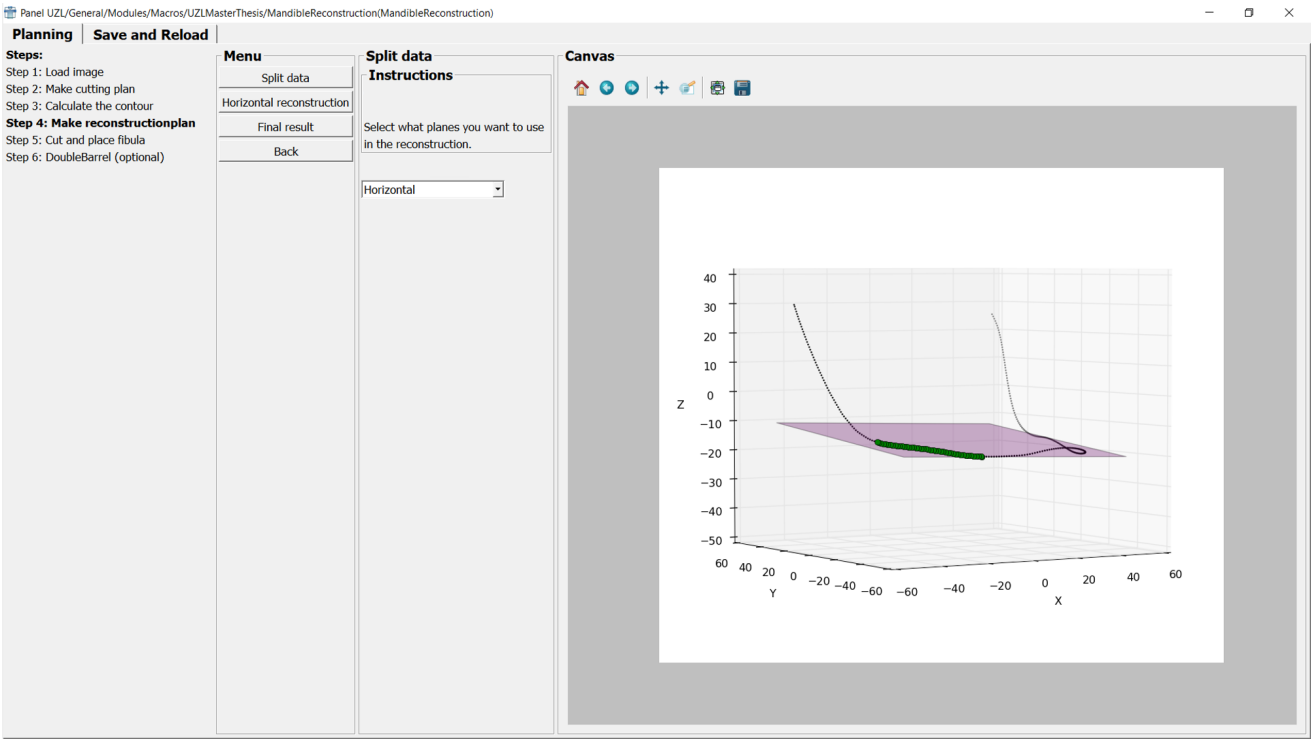


Figure B.5: The interface for splitting the data in vertical and horizontal data. Here, only the plane that fits trough the horizontal data is shown

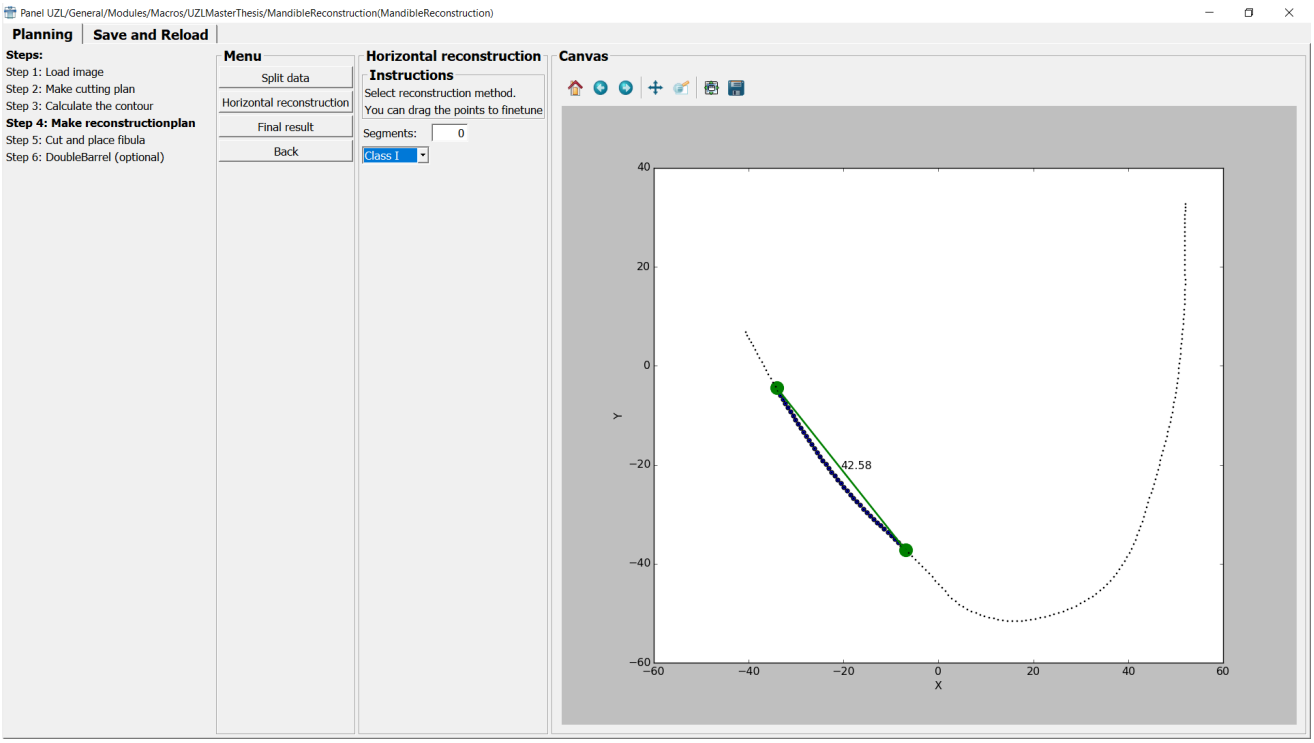


Figure B.6: The interface that allows the user to place 1-3 segments in the horizontal plane

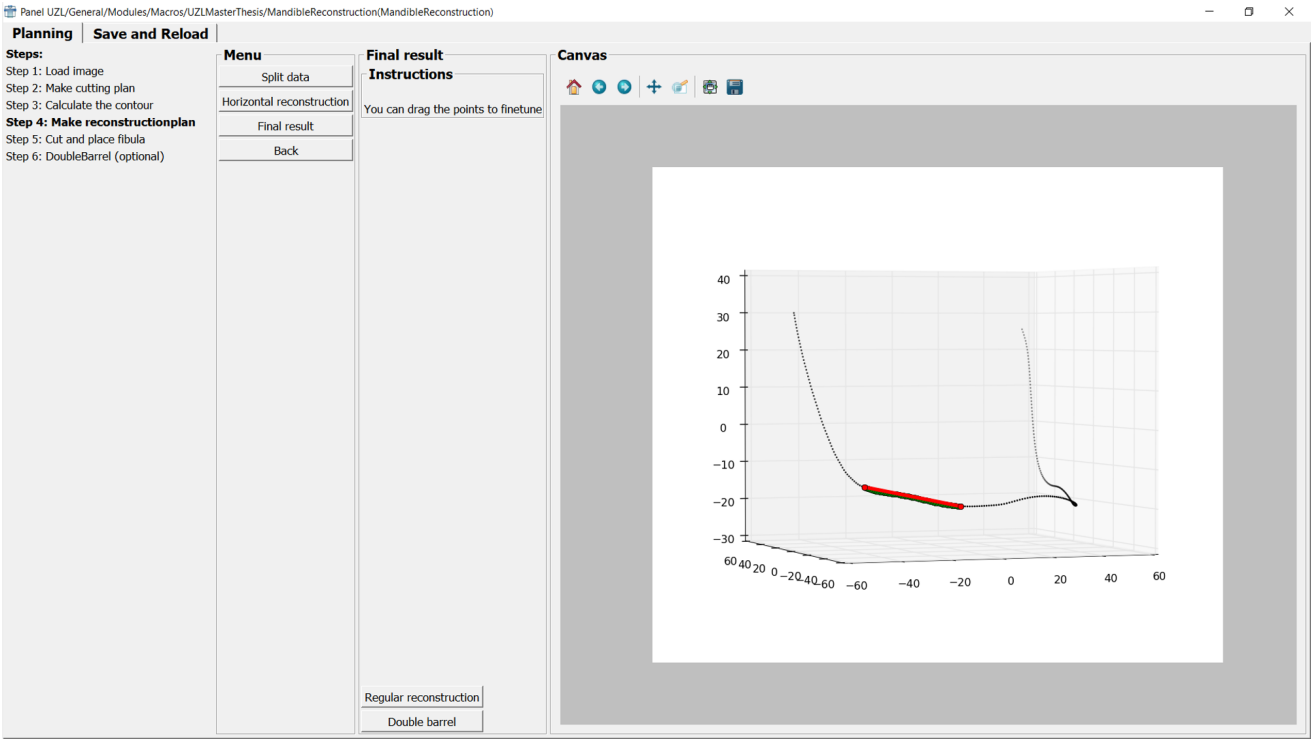


Figure B.7: The interface that shows the final result of the planning with the contour line

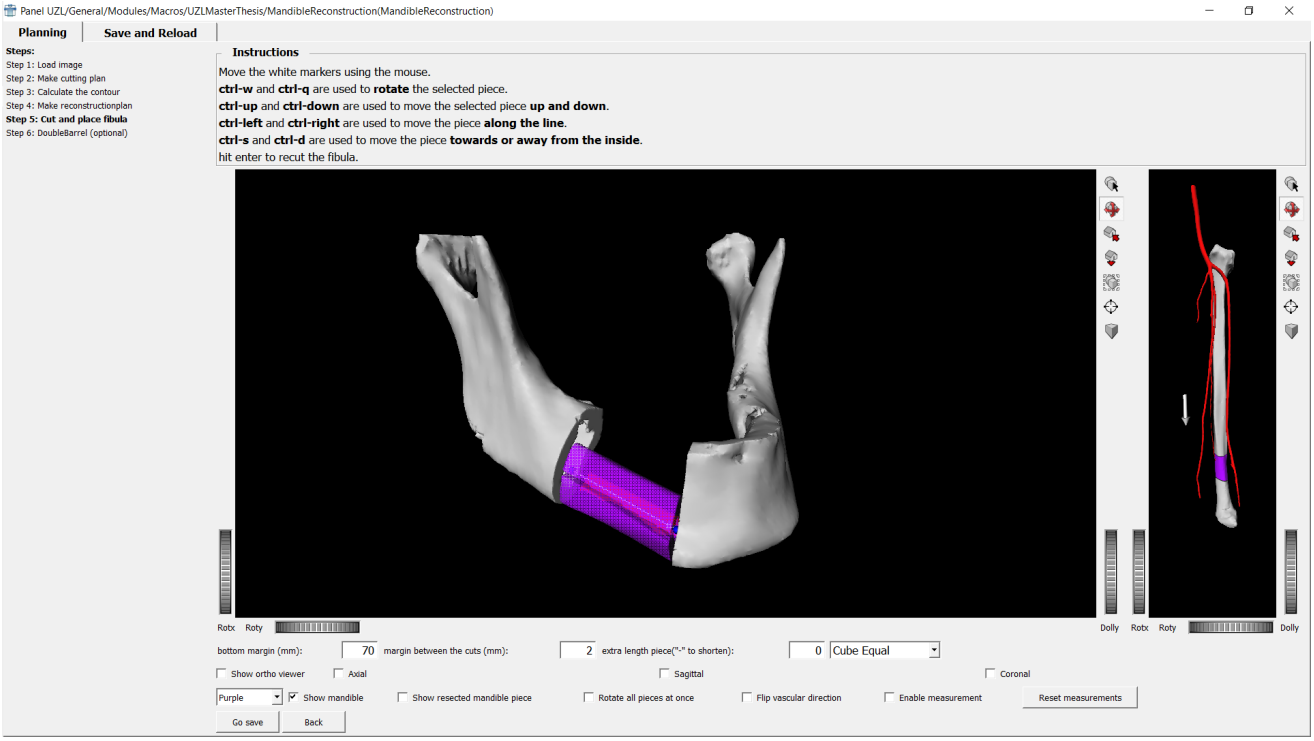


Figure B.8: The interface of step 5

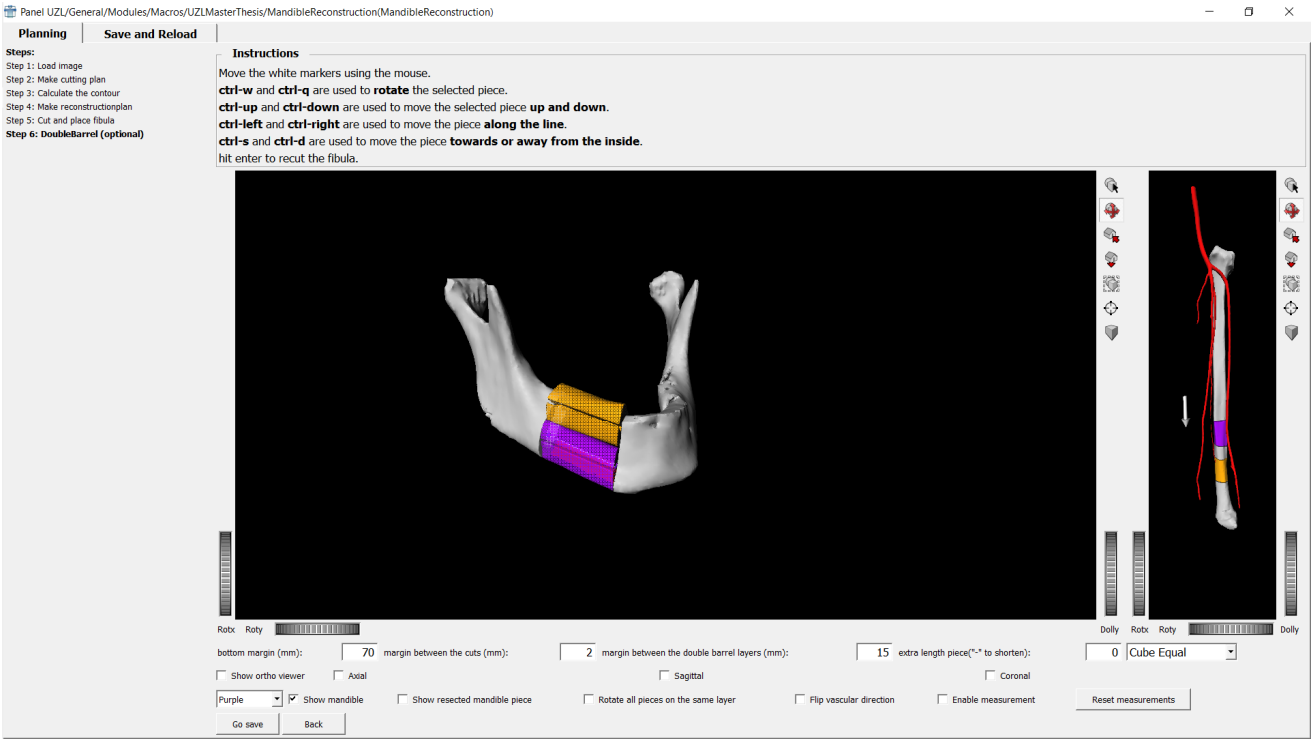


Figure B.9: The interface of step 6

Figure B.10: The interface for exporting the files and saving as well as reloading the project