Centre Line Computation of Blood Vessels in the Mesh Domain

Karim Fathy

Department of Radiology and Biomedical Imaging, Faculty of Medicine and

Health Science, University of Sherbrooke

Dr. Kevin Whittingstall

July 29, 2021

Acknowledgments

I would like to thank Felix Dumais for his continuous support and help throughout this internship. His aid has resolved numerous bugs and problems. Also, I would like to thank Reihanh Forouhandehpour for her transfer of knowledge.

Abstract

I have joined Dr. Whittingstall's lab to develop an application that automates the computation of the centreline of blood vessels in the Circle of Willis. The lab used to utilize VMTK, a tool that analyzes and models blood vessels; however, due to its need for user input, it was time-consuming to analyze tens or even hundreds of patients. Also, VMTK is no longer supported by its developers, and has reached what is known in the software development world as its 'End of Life.'

The aim of the project is to input a GT file, a nifti file that contains artery annotations. The program should output a centreline for the inputted arteries. That centreline would then be used to compute the diameter of the blood vessel and other measurements.

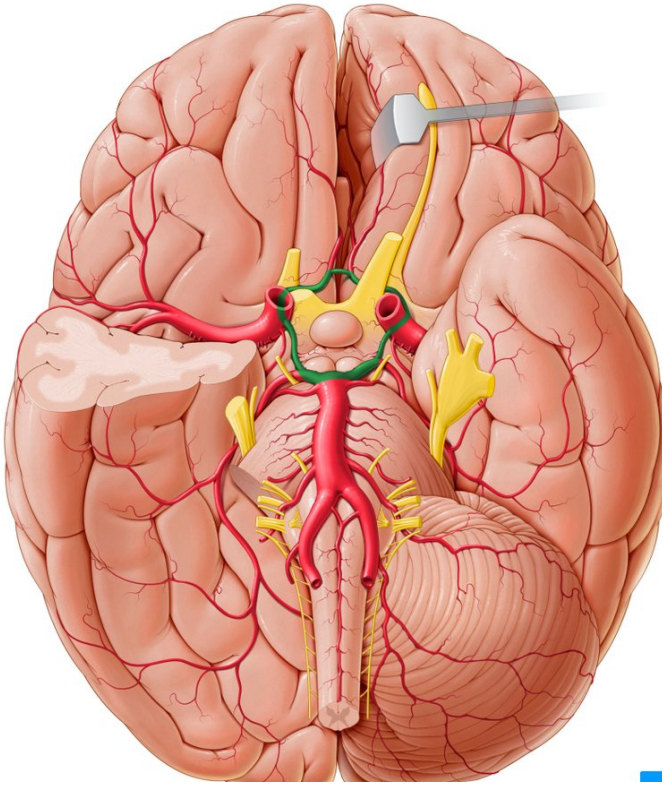

Figure 1, Illustration of the Circle of Willis, kenhub.com/en/library/anatomy/circle-of-willis

The project delivered can filter a GT image according to user input on which artery is needed. The pipeline then continues to construct a mesh of the filtered artery using marching cubes and smoothes the mesh using Laplacian Filtering. It continues by generating a skeleton of the mesh which is basically a centreline of the blood vessel; however, the priorly mentioned method has proven to show a high standard of deviation when computing the average diameter of the artery through the use of a mesh's centreline. The diameters extracted are inconsistent and inaccurate.

Introduction

The Circle of Willis is a group of arteries found in the base of the brain where they intersect and form a circle (Circle of willis: Anatomy, function, and what to know). Although he was not the first to discover it, it was named after Dr. Thomas Willis. He was the first to explain its function. Also, he provided a full and undisputed drawing of the Circle of Willis (Rengachary et al., The legendary contributions of Thomas willis (1621–1675): The arterial circle and beyond 2008). The circle consists of left and right internal carotid arteries, left and right anterior cerebral arteries, left and right posterior cerebral arteries, left and right posterior communicating arteries, anterior communicating artery, and the basilar artery.

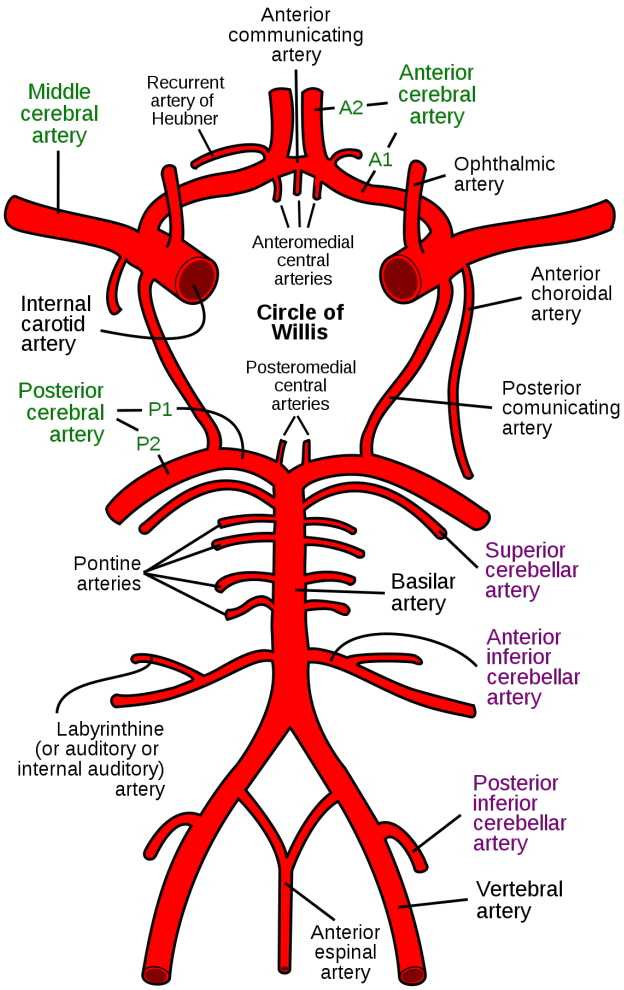

Figure 2,Illustration of the Circle of Willis, commons.wikimedia.org/wiki/File:Circle_of_Willis_en.svg

The circle provides blood flow to most of the brain and may prevent strokes by changing the direction of blood flowing through it, forward or backward, when a connected artery has a blockage or narrowing. Its shape allows it to provide bidirectional blood flow whenever needed (Circle of willis: Anatomy, function, and what to know).

The Pipeline/The Methodology

Artery Filtering

In the long term, the objective of the application is to analyze the whole Circle of Willis and compute its centreline; however, as a proof of concept, the project was focused on computing the centreline for a small group of blood vessels. The group consists of the left and right posterior cerebral arteries and the basilar artery. However, later on, we have decided to focus mostly on the basilar artery to generate the best results for one particular artery. The pipeline utilizes the Numpy library to extract arteries according to their colour code on the nifti file. The function “artery_extraction” uses the integer array “artery_filter” to extract specific arteries, selected by the user. By calling the function the floating data of the nifti image is extracted. Then the image’s matrix is then cloned and set to zeros. Then the voxels with the corresponding colour codes will be set back to one.

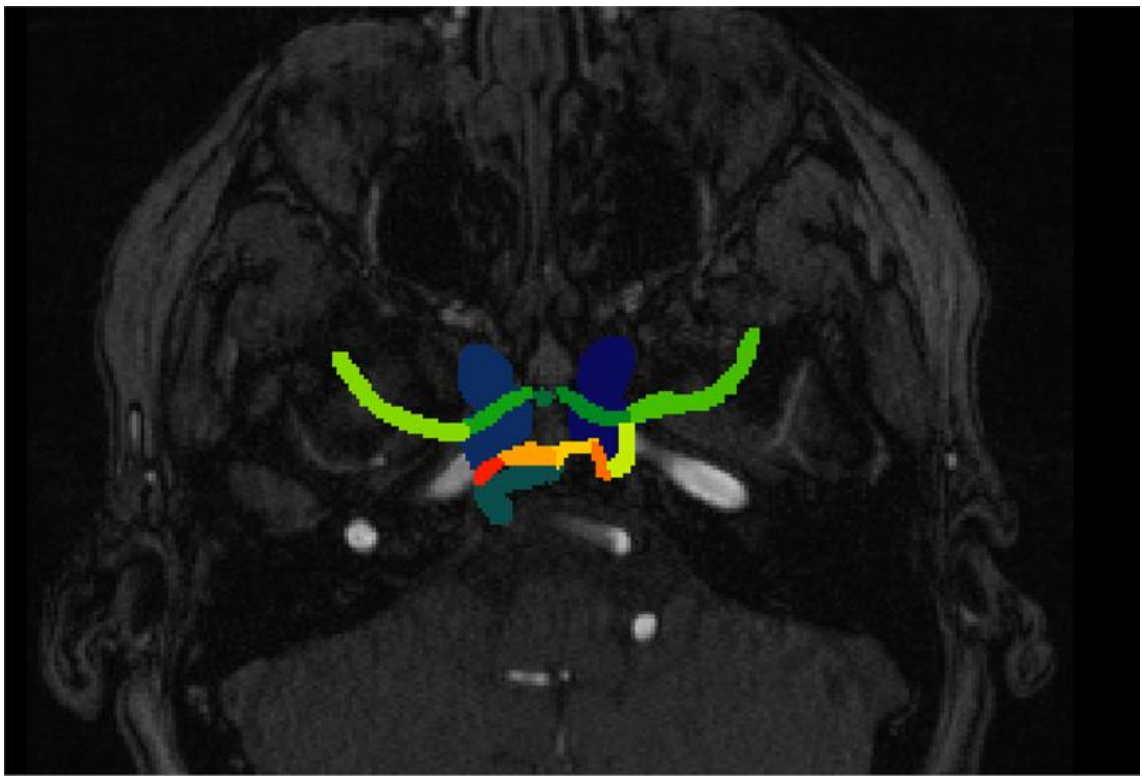

Figure 3, An annotated Circle of Willis

Meshing

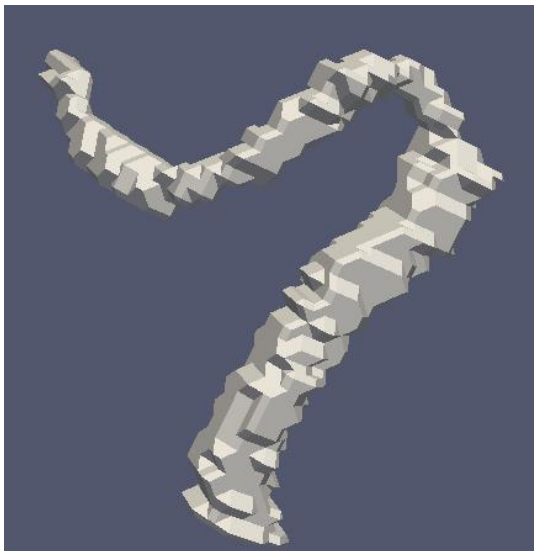

Figure 4, A mesh generated from the pipeline for the basilar, P1 and P2

The marching cube algorithm is used to generate a mesh of the selected arteries. It is an iterative algorithm that creates triangular meshes from voxels. The application relies on the Skimage library's implementation of the marching cube algorithm to turn the voxel image into a mesh. The pipeline utilizes the "marching_cubes_lewiner" function to generate the vertices and faces of the mesh. Optimal results were obtained when keeping the step size between 1 and 2. Also, a link with a detailed description will be left in the references section. Furthermore, the generated vertices and faces are then saved as an stl file using the "make_stl" function.

Smoothing

Figure 5, The mesh in Figure 4 after smoothing

To compute an ideal centreline, the mesh has to be smoothed. The pipeline uses Laplacian Filtering to symmetrically smooth the mesh. Trimesh's smoothing algorithms were a suitable choice. Trimesh is a python library that manipulates triangular meshes. After a number of trials, it seemed that the method that utilizes both Laplacian smoothing and Taubin filtering,"trimesh.smoothing.filter_taubin," generated the best results. Thus, it was set as the default method for smoothing; however, the user can choose his preferred smoothing algorithm through the parameter "-sa."

Skeletonizing

Furthermore, the smoothed mesh is then skeletonized using the Skeletor library. The Skeletor library is a python library that transforms a mesh into a skeleton by contracting the mesh into a skeleton through Laplacian smoothing. A skeleton object is similar to a tree with nodes that are connected through edges. Skeletor divides the skeletonizing process into three stages preprocessing, skeletonization, and post-processing; however, most of the preprocessing and post-processing functions have no observable or beneficial effects on the mesh or skeleton. The pipeline relies on the "skeletonize.by_wavefront" function which

was recommended by the developers of Skeletor to skeletonize tubular meshes such as blood vessels. In the example provided, the wavefront function extracts an ideal skeleton out of a neurone's mesh. I have experimented with all of Skeletor's skeletonize functions, and I have failed to generate acceptable results except with the wavefront function.

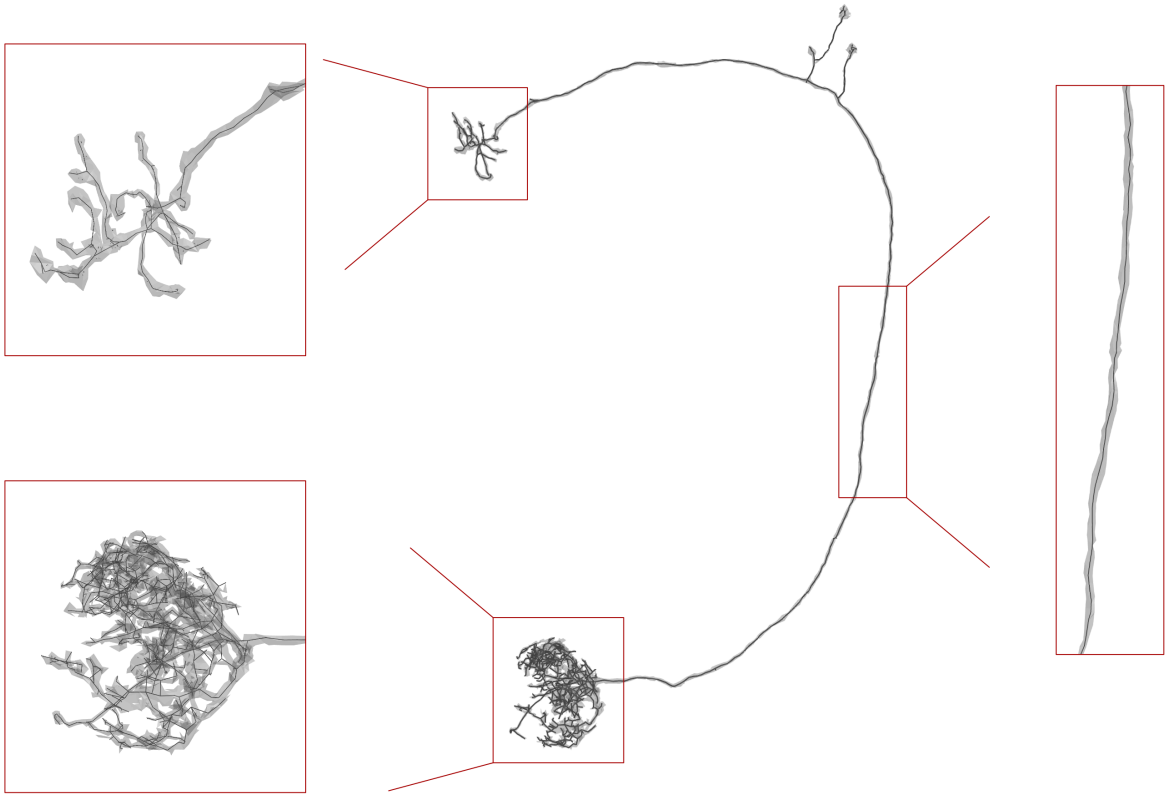

Figure 6, A skeleton of a neurone's mesh, schlegelp.github.io/skeletor/

Graph Manipulation, Centreline Extraction, and Diameter Computation

The final stage of the pipeline is graph manipulation where the skeleton object is loaded into a NetworkX graph to remove any unwanted branches that occur due to inconsistencies in the mesh surface or noise generated from the wavefront method. The class "GraphGenerator" is responsible for loading and manipulating the skeleton's structure. Also, it extracts the centreline and computes the average diameter of the blood vessel. The class requires two parameters, the skeleton and the artery numbers being represented by the skeleton. When an object of the class is being instantiated, the constructor is called, the class uses the skeleton's vertices and edges to construct a NetworkX graph, this is done fairly easily since the Skeletor library relies on NetworkX to construct skeletons. This is followed by the extraction of the endpoints of the graph by using a NetworkX node's degree attribute.

Furthermore, two in-house algorithms were developed one to handle branches in the p1, p2 and basilar, and the other is specific to the basilar. The first algorithm basically generates all the simple paths between all the endpoints in the graph. The equation for the number of simple paths between endpoints is in the figure below. The algorithm finds the longest path of those paths by counting the number of vertices on the path.

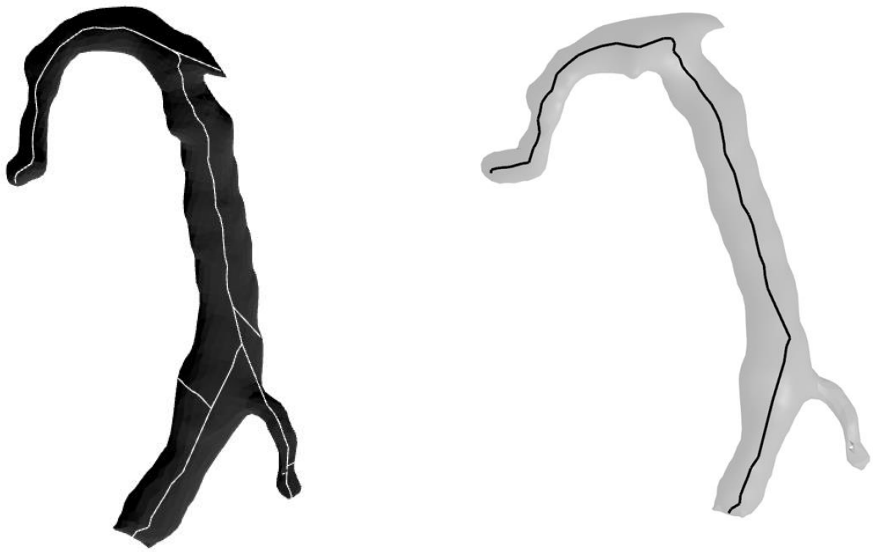

Figure 7, A skeleton of a basilar, P1, P2 before graph manipulation (left) and after graph manipulation(right)

Moreover, due to the inconsistency of results, we have chosen to focus all efforts on one artery which is the basilar, in hope that the Skeletor library would have a better time skeletonizing a simpler shape such as a basilar. Basically, the basilar's algorithm finds the highest and lowest endpoint on the z-axis and gets the simple path between them.

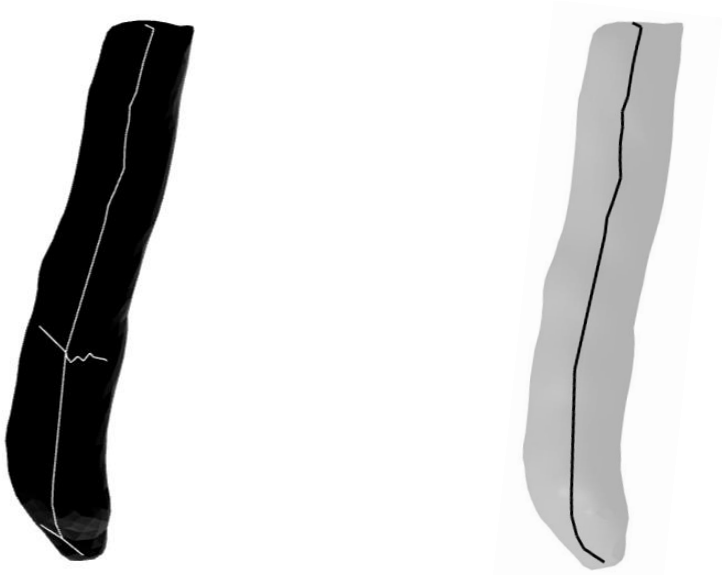

Figure 8, A skeleton of a basilar before graph manipulation (left) and after graph manipulation(right)

Additionally, vertices that are not on the longest path are removed from the skeleton. Also, the average diameter is then computed using Skeletor's radius attribute for nodes, and the centreline's vertices and edges are then extracted.

Reindexing the SWC Table and Plotting the Centreline

A skeleton object is built around an attribute called swc, a Pandas DataFrame that contains all vertices on the skeleton, their coordinates, their radius, and the child node that they are connected to. After the extraction of the centerline vertices, they are then loaded to Skeletor's "make_swc" function which will return a new swc table; however, the vertices need to be arranged in a manner that does not conflict with Skeletor's condition that each node must only have one parent node. Thus, the index function is called to rearrange the swc table.

	node_id	parent_id	x	y	z	radius
0	0	-1	79.755943	100.978722	34.361389	0.409274
1	1	0	80.038332	100.098990	34.072319	0.513647
2	2	1	80.407982	99.337925	33.645809	0.339549
3	3	1	80.054340	100.483073	34.449378	0.576247
4	4	3	80.188560	100.887939	35.052855	0.866129
5	5	4	80.435219	101.155129	35.675161	1.278588
6	6	5	81.222864	102.227116	40.338951	1.412396
7	7	6	80.211185	103.026106	41.165115	0.471593
8	8	6	81.385815	102.215768	40.127497	1.620725
9	9	6	82.642893	102.529577	45.430785	1.492681
10	10	8	81.616324	102.128070	39.938514	1.847675
11	11	9	82.704141	102.576668	46.135707	1.424000
12	12	10	81.949981	102.067391	40.181010	1.579314
13	13	12	82.312035	101.955907	40.014998	1.211416

Figure 9, An example of a swc table

Furthermore, the swc table is then used by a skeleton constructor to initialize a new skeleton object. The new skeleton could be plotted using “skeleton.show(mesh = True).” Setting the mesh parameter to true allows the user to see the centreline plotted inside of the original mesh.

Results

Trials on P1, P2, and Basilar

The main proposal at first was extracting the centreline for the p1, p2, and basilar arteries; however, numerous issues arose such as handling patients with no P1 artery. Although this exception case can be easily handled by computing the centreline for both P2 and basilar separately, two other problems made the proposal unfeasible. Firstly, is the high noise that occurs in some patients. The skeleton would sometimes be distorted and imitates a zigzag shape. The secondary issue is that the longest path between two endpoints could be between two endpoints on false branches or between the correct entry or exit point and a point on the false branch. Thus, we decided to focus on just the basilar as a proof concept.

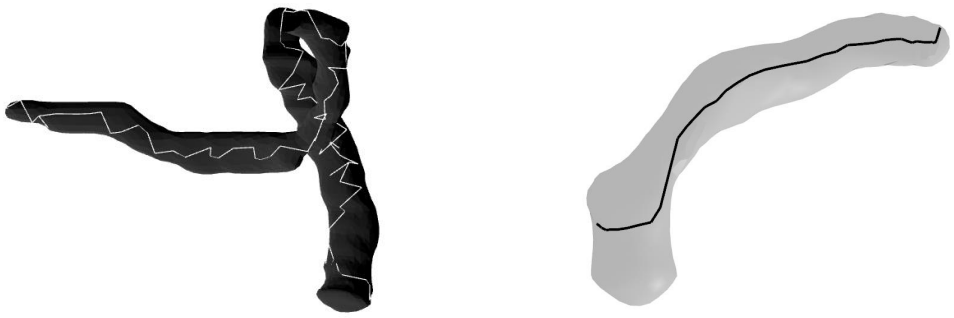

Figure 10, A skeleton with the high noise issue (left) and a skeleton with the false path issue (right)

Trials on the Basilar Artery

Trials on the basilar artery have shown more consistent results. The high noise issue is not present with the basilar artery, and the skeletons are plotted accurately. Also, no incorrect endpoints have been chosen off of branches, because the centreline is extracted by searching for the path between the highest and lowest endpoint on the z-axes. Additionally, no notable issues have been detected on the trials carried on the 10 patient sample used.

Furthermore, one of the uses of extracting the centreline is computing the average diameter of an artery. The method used to calculate the average diameter was using Skeletor's radius attribute as mentioned above. Unfortunately, the radii extracted from Skeletor were unreliable, and the average diameter computed over the sample showed a high standard deviation due to the fluctuation of values throughout the sample. One of the main reasons for this was that the inlet and outlet vertices, the endpoints, of the centreline tended to arch towards the mesh body. Thus, causing the first and last few vertices to have very small radii compared to its centre. In figure 11, the first few vertices have a lower value than the vertices closer to the midpoint of the centreline. Also, some vertices tend to have a radius value of zero as visible in figure 10.

Figure 11, An array representing the radii of all vertices from endpoint to endpoint

Comparing the diameters computed from this pipeline with the ones computed by the lab's voxel space pipeline, the voxel space pipeline shows far more consistent results than the mesh space pipeline. In figure 12, most voxel space diameters lie between 3.5 and 4 mm while most mesh space diameters are between 2 and 3 mm. The decrease in diameter value from the voxel to the mesh domain is understandable, due to the smoothing that contracts the mesh slightly.

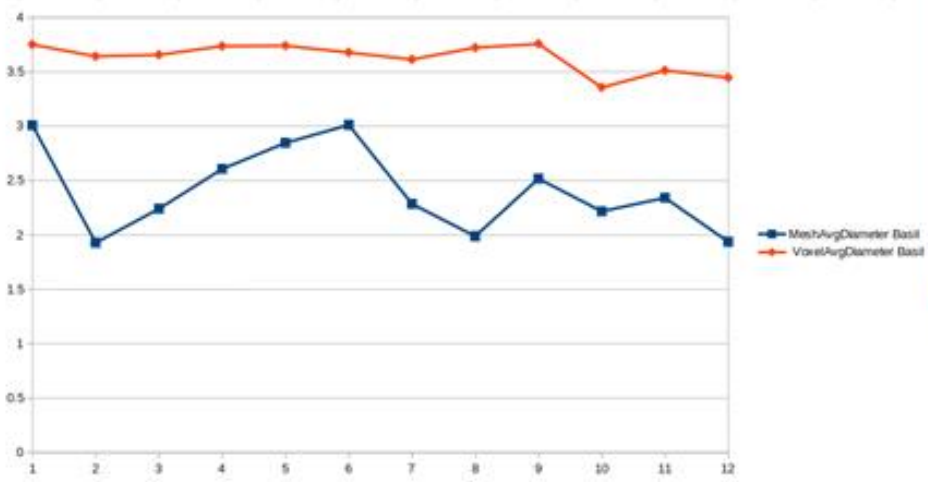

Figure 12, Comparison between the average diameters of patients' basilar artery in the voxel space (orange) and mesh space (blue)

Moreover, trying to avoid the inlet/outlet issue we compared the diameter of the midpoint vertices from the mesh space with the diameter of a voxel slice positioned around the centre of the basilar artery using FSL-eyes, a voxel space tool. The high standard of deviation still persisted. Thus, proving that Skeletor's radii are unreliable and inaccurate. Results of this method are plotted in figure 13 below.

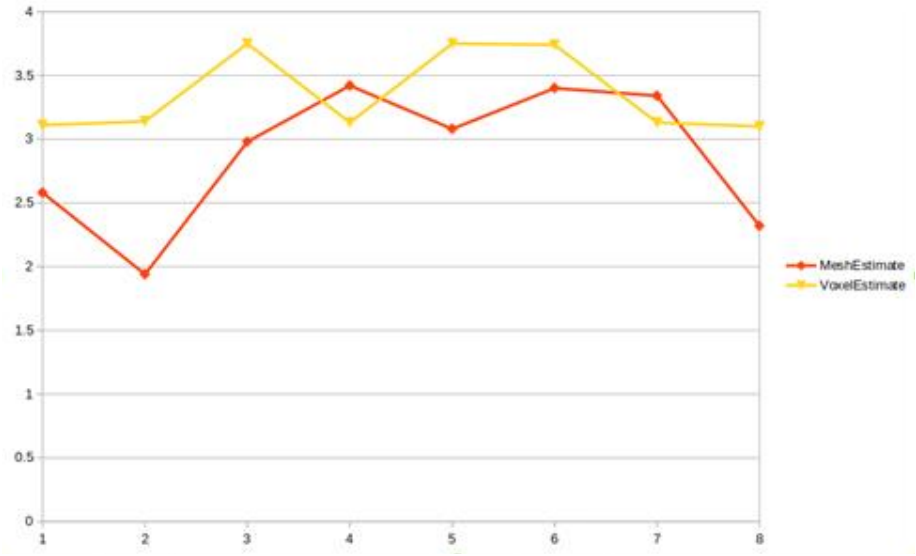

Figure 13, Comparison between the diameters around the midpoint of patients' basilar artery in the voxel space (yellow) and mesh space (red)

Conclusion

The pipeline is very reliable for filtering the Circle of Willis according to colour code, making a mesh of the filtered arteries, and manipulating an artery's skeleton. The pipeline is only consistent in extracting skeletons of one artery at a time, and faces problems when the blood vessel becomes more complex or longer. Skeletor's radii have proven to be inaccurate even if the centreline is plotted correctly.

I would have liked to compute the diameter of the blood vessel without relying on Skeletor's radii; however, the low resolution of the skeleton, due to its low number of vertices, made the task quite difficult. The neurone example used by Skeletor is far more complex than any blood vessel in the Circle of Willis; nevertheless, Skeletor was able to extract a perfect skeleton from the neurone mesh. After a deep analysis of the mesh of the neurone, it seems that the mesh is a little bit different than the meshes made by the pipeline. The neurone mesh seems to have fewer triangles than the mesh generated by the pipeline; however, I have failed to understand why Skeletor works best on a mesh with fewer triangles meaning lower resolution. It may be that the neurone mesh used

is an ideal input that Skeletor's developers configured the library for, or maybe that Skeletor only supports certain types of meshes. Although Skeletor was unreliable in extracting the centreline and computing the radii, it can still be utilized to plot the centreline in the original mesh.

References

- MediLexicon International. (n.d.). *Circle of willis: Anatomy, function, and what to know*. Medical News Today. https://www.medicalnewstoday.com/articles/circle-of-willis#function.

- Rengachary, S. S., Xavier, A., Manjila, S., Smerdon, U., Parker, B., Hadwan, S., & Guthikonda, M. (2008, October 1). *The legendary contributions of Thomas willis (1621–1675): The arterial circle and beyond*. jns. https://thejns.org/view/journals/j-neurosurg/109/4/article-p765.xml.

- Edwin Ocran MBChB, M. S. (2021, May 31). *Circle of willis*. Kenhub. https://www.kenhub.com/en/library/anatomy/circle-of-willis.

- *Module: Measure¶*. Module: measure - skimage v0.19.0.dev0 docs. (n.d.). https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.marching_cubes.

- *trimesh.smoothing¶*. trimesh.smoothing - trimesh 3.9.27 documentation. (n.d.). https://trimsh.org/trimesh.smoothing.html.

- *skeletor*. skeletor API documentation. (n.d.). https://schlegelp.github.io/skeletor/.