Segmentation of Blood Vessels from 3D Medical Image

Author: Vineet Jacob Kuruvilla, vineet.jk@gmail.com
Date: 06/06/2018

Objective

1. To segment the desired blood vessels from a 3D medical image (in vti format) using thresholding and to export it as STL or VTP file
2. Segment one of the two blood vessels by allowing the user to select one or more seed point. The segmentation should be done without any explicit input for the threshold value from the user

Summary

For *objective 1*, three different approaches were explored and implemented:
A) Numpy & Manual Thresholding- The vti file was read and image was converted to numpy dictionary and processed. The image threshold to extract the blood vessels were given as input by the user. The threshold value was decided based on values in the blood vessel region observed on multiple slices.
B) VTK & Manual Thresholding- In this method, the data was processed as a vtkImageData itself. There was no conversion to numpy or any outside format. The thresholding was also done in VTK using vtkImageThreshold. Here again, the threshold value was decided based on observations from image slices.
C) Numpy & Threshold value based on seed point- In this method, the image was processed as numpy array as described in method (a). But the threshold value was not set by the user. Here, the user is shown the first slice (or any slice) in the medical image and is asked to make a single click in the blood vessel region. The threshold value is computed based on the intensity values in a small ROI around that point and is then used to threshold the image. This seed point is used not just for discovering the threshold value but is also used in the next bonus task to segment one of the blood vessel.

In all three method, the segmented blood vessels are converted to a PolyData and then exported as STL file, which can be viewed using ParaView software. In all three method, some noise/extraneous elements are visible in the STL data. Least noise was observed in Method C. Based on performance and ease of use, Method C is

considered the best among the three methods. The bonus task was built upon the this method.
A fourth method was also looked into. It is a modification of Method C. It is mentioned in Additional Study section, point 5.

For *objective 2*, only one method was implemented. For this task one or more seed points can be given by user and this is then used to segment one of the two blood vessels present in the 3D medical data. I have used the vtkImageThresholdConnectivity for this task. This method flood fills an image based on the upper and lower threshold values using the seed points to stick to a continuous region. In my implementation, I have used 6 seed points but one well placed seed point (for example, somewhere in the midsection of the volume) can give the exact result. The program was able to accurate segment the blood vessel and discard all other parts in the 3D volume.

Dependencies

1. Python 2.7
2. Numpy 1.11
3. scikit-image 0.13.0
4. matplotlib 1.5.1
5. vmtk 1.4.0
6. vtk 8.5.1

Dependencies 1-4 was installed from Anaconda Distribution. The installation file and instruction can be found here.
Dependencies 5 -6 were also installed from Anaconda Distribution. The instructions can be obtained here.

Operating System used was Ubuntu 16.04
IDE used was PyCharm Community Edition 2018.1
The results were viewed using ParaView 64-bit version 5.0.1

Method & Result- Objective-1

Method A - Numpy & Manual Thresholding

Here are the steps I followed:

1. Read in img.vti file using vmtkImageReader. This allowed me to convert the imagedata to numpy array dict
2. Convert ImageData to numpy array dictionary using vmtkImageToNumpy

3. Display images slice-wise so as to understand the threshold value to be used for blood vessels. The threshold value used in the program was greater than or equal to 450. An upper limit of 1000 was also used though it didn't make any difference. Matplotlib library was used for displaying the slices.

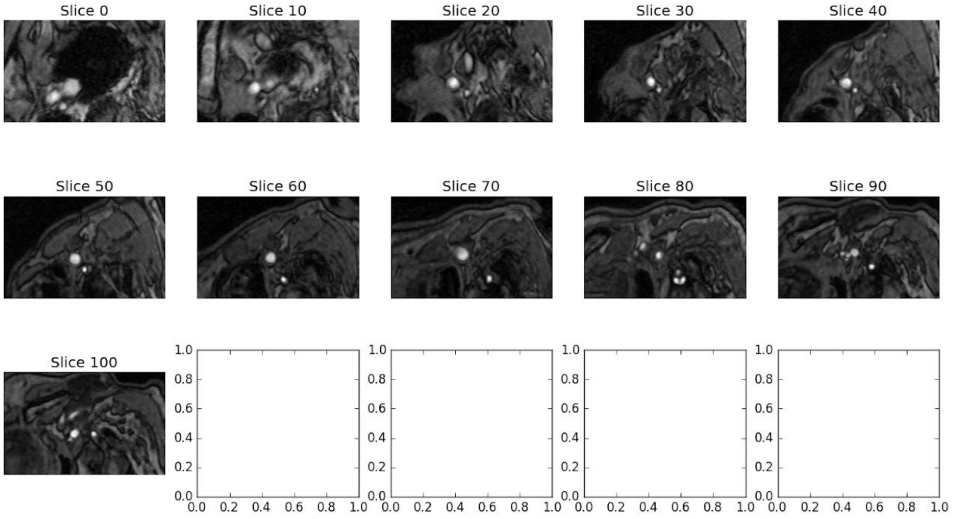

Figure 1 Image slices along the Z axis

4. Thresholding the image (numpy array) using the values from step 3 to obtain a binary image

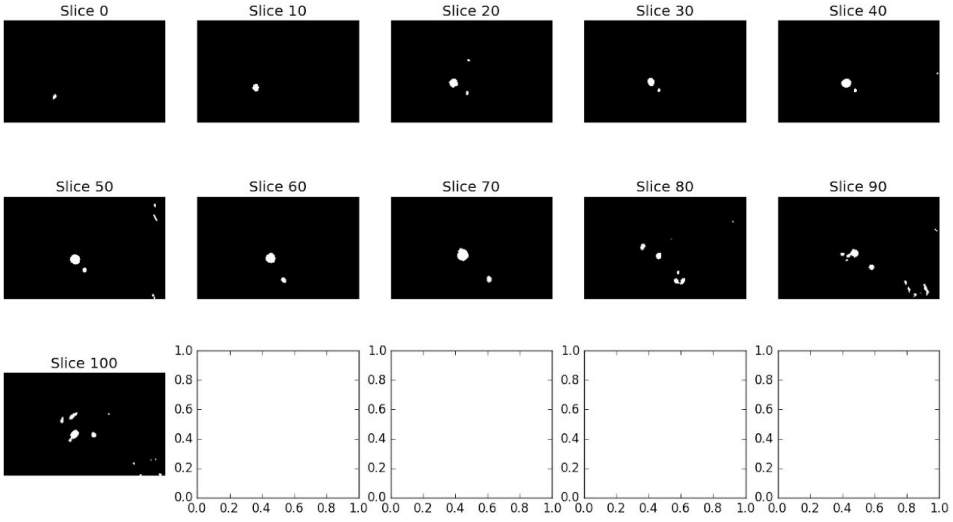

Figure 2 Image slices after thresholding

5. Erosion, followed by dilation so as to remove small specks in the image and also to recover lost boundary of the regions that remain, which is mostly that

of the blood vessel. Scikit-image library was used for this step. A kernel of size [3,3] was used for both erosion and dilation.

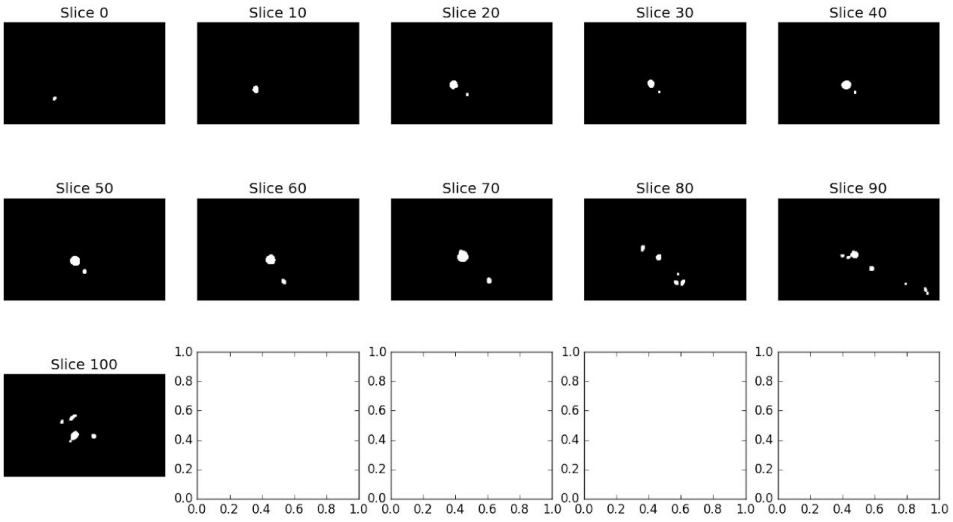

Figure 3 Image slices after erosion and dilation

6. This “cleaned” binary image is then written back to vtkImageData
7. The 3D surface is generated from the vtkImageData using the Marching Cube algorithm. The VTK implementation of marching cubes (vtkMarchingCubes) was used for this purpose.
8. The 3D volume is then exported to STL format using vtkSTLWriter

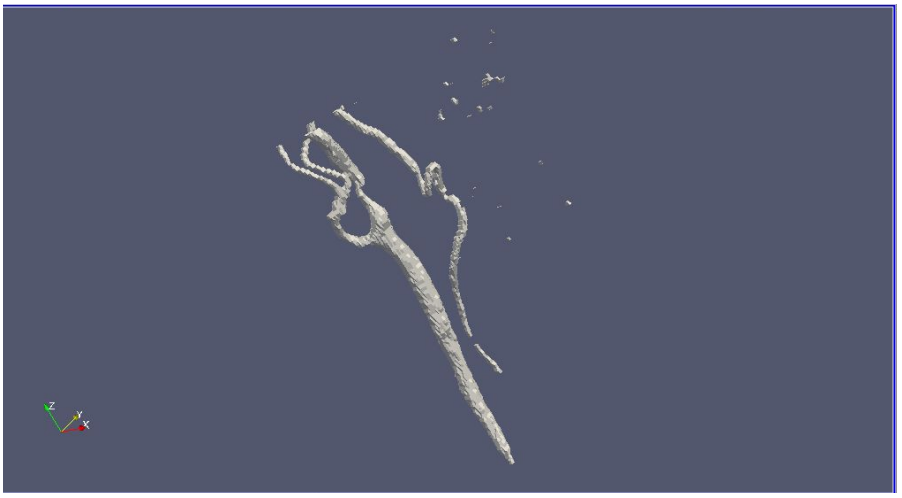

Figure 4 Segmented blood vessel. STL file was visualised in Paraview

Thoughts on the result

1. The threshold values were entered manually. This may not be an ideal solution in practice.

2. Inspite of the processing, there was still extraneous elements in the threshold image and hence, also in the final STL file as can be seen in the figure. The method in the bonus task can be a possible solution to this noise problem. Also additional filtering can be used to drastically improve the result.
3. Though the vessel are fully connected, some parts are missing especially in the vertebral artery. Better threshold values needs to be used.

Method B- VTK & Manual Thresholding

Method B was implemented only using vtk library. Reading and Thresholding and final import to STL file was done in VTK.
Here are the steps I followed:

1. Read in the img.vti file using vtkXMLImageReader (Same as Figure 1)
2. Threshold the image using vtkImageThreshold. The threshold values used were the same ones used in Method A. This method too requires user input for the threshold value

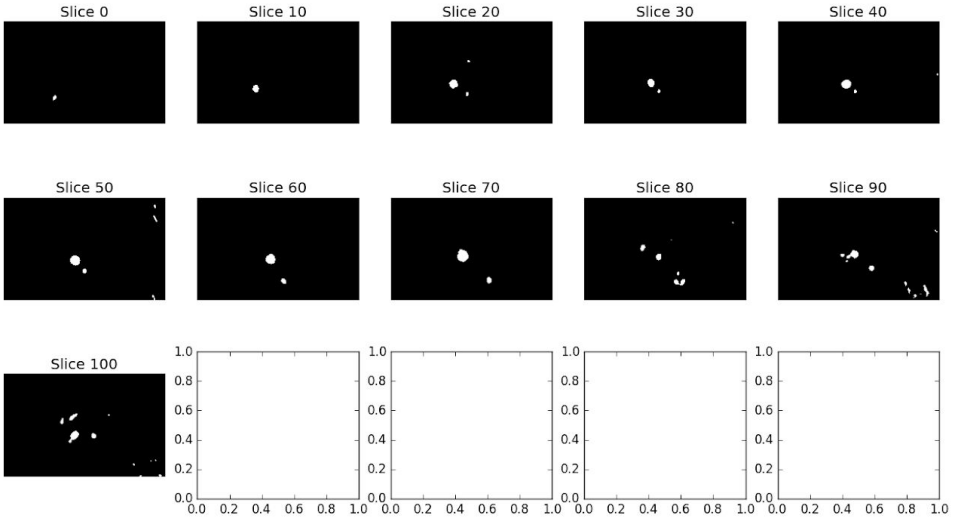

Figure 5 Image slices after thresholding

3. Generate the 3D volume from the threshold data using vtkMarchingCubes
4. Export the volume data as STL file

Figure 6 Segmented blood vessel. STL file was visualised in ParaView

Thoughts on the result

1. The program is much smaller and easier to follow as only one library (VTK) is used. Also since the image datatype/format are all in vtk standard there are no conversions required and errors are minimised
2. In this program, after thresholding, no filtering was done. Hence, the amount of noise in the extracted volume is much larger compared to Method A. This is one aspect I would improve upon.
3. Given more time, I would do the whole processing in VTK, including filtering and visualisation.

Method C- Numpy and Threshold from seed point

The processing in this method is very similar to that of Method A. The file img.vti is first read in and then converted to numpy array dictionary. The difference between Method A and C is in the way the threshold value is discovered. As explained previously, in Method A, the threshold value is found by the user by analysing different slices and the intensity levels in the blood vessel region.

In Method C, the user is showed one slice of the volume and asked to select the blood vessel region with a single mouse click. With this click, a small ROI ([6,6] around the clicked point) is extracted and the mean and standard deviation of the intensity levels in that ROI is computed. The threshold value is set as the upper limit of the intensity distribution of the ROI,

threshold value = intensity_mean + intensity_stdev.

The upper limit is used as the threshold value as the mean value tends to underestimate the intensity of the selected region due to presence of some surrounding tissues in the ROI.

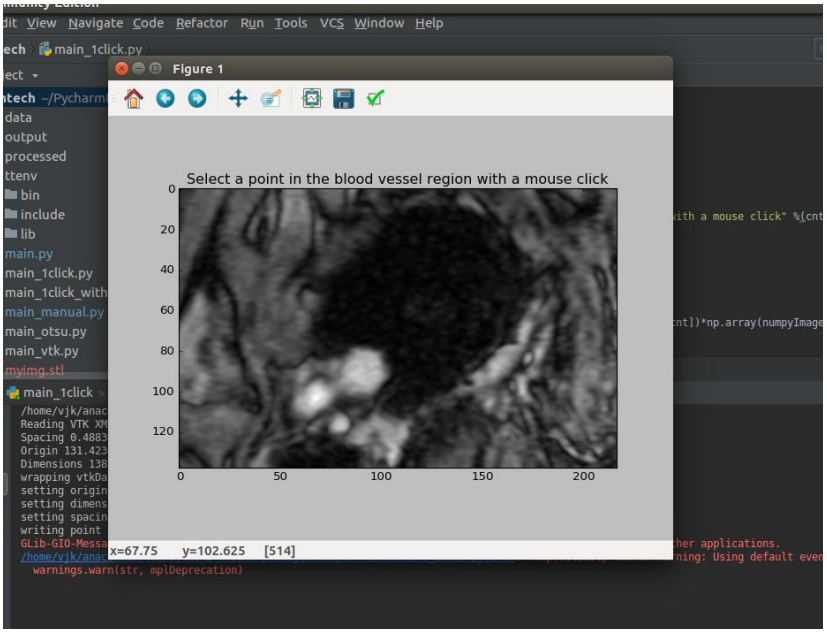

Figure 7 Slice (1st) display for user to select a point in the blood vessel

The seed point selected by the user is not just for threshold calculation but will act as the seed point (or one of the seed points) for Objective-2 as well. In the Figure 7, the slice showed is the 1st slice in the image data. But any slice, ideally from the middle section of the 3D image data, can be used to compute the threshold value and also used as the seed point.

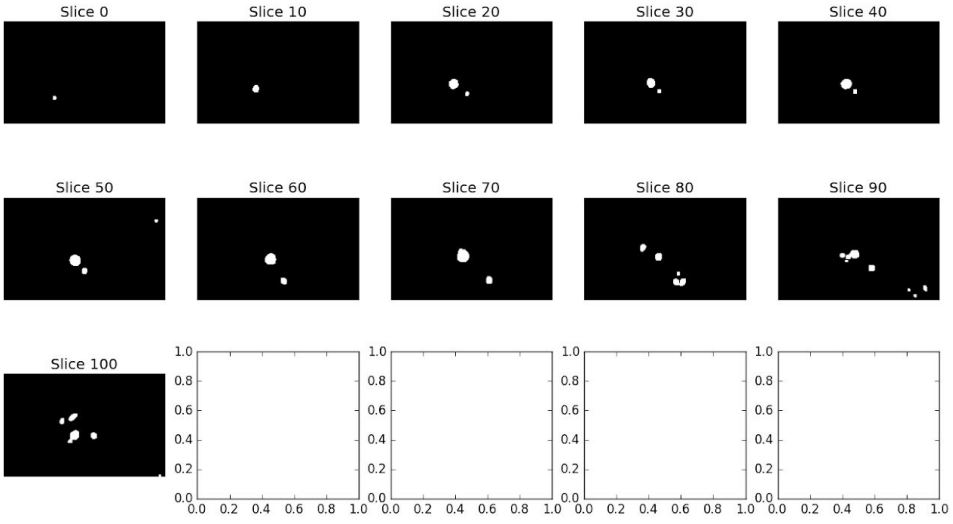

Figure 8 Image slices after thresholding using the threshold value from seed point and erosion & dilation

As in Method A, after the threshold value is computed, the program proceeds to threshold the 3D image data. Figure 8 shows the slices after thresholding, erosion and dilation.

The following steps are also same as that of Method A, the cleaned up binary numpy array is then written back to vtkImageData. Marching Cubes algorithm (vtk implementation) is then used to build 3D volume from the image and then finally exported as STL file. The final result is shown in Figure 9.

Figure 9 Segmented blood vessels. STL file was visualised in ParaView

Thoughts on the results

1. The result with this method is comparable to that of Method A. However, the ease of use and level of automation is higher in Method C.
2. Just as explained in Method A, the noise filtering can be improved upon. Also, a better and more automated solution to threshold value identification or feature identification using machine learning can be adopted with more data.
3. The key advantage of this method is that it can easily be extended to use region growing algorithm to identify any desired volume in the 3D image data
4. Improvements can be made in threshold value estimation method. Right now I use a very basic method to estimate the threshold value.

Method & Result- Objective-2

As mentioned in the Tech Task document, there are different ways to implement a region growing algorithm to identify the required blood vessel. In this bonus task, I have used the VTK library, vtkImageThresholdConnectivity to solve the task.
I have built this part as a continuation of the Method C implementation. In Method C, the threshold value is identified using the ROI around a seed point, supplied by the user with a single mouse click. In this objective 1, we get a 3D image data with

binary values where it is values are 1 in the blood vessel region and 0 in all other surrounding areas.

In the result shown here I have used 6 seed points to identify the volume to be extracted. However, even one seed in the middle section of the volume was able to extract the full volume of the required blood vessel. Just as in the case of seed point selection for threshold value, the user is shown different slices and has to select a single point in each slice to identify the blood vessel region.

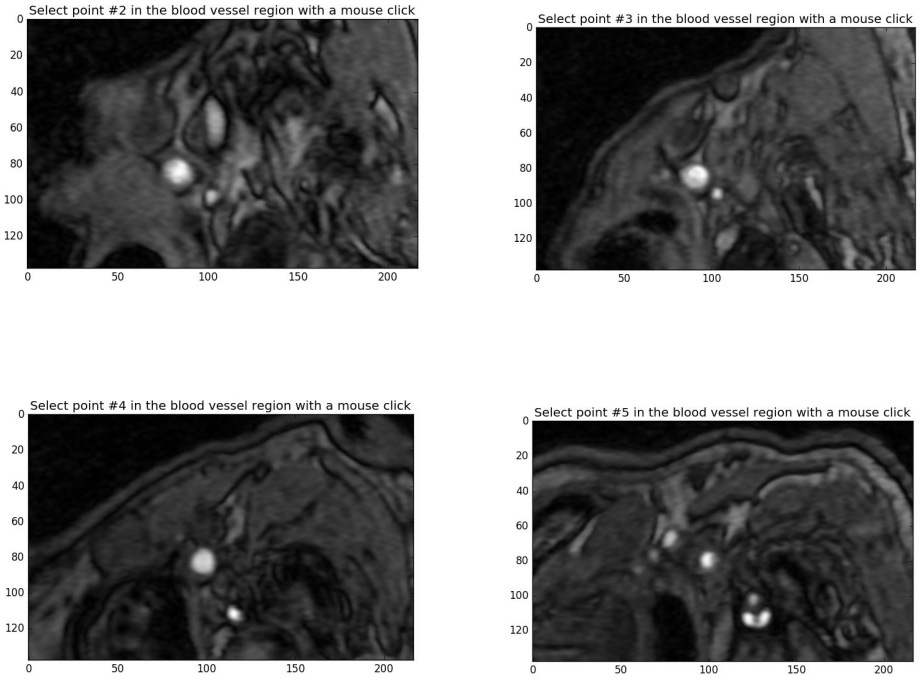

Figure 10 Image slices shown for the user to select the seed points

The user is asked to enter the number of seed points he/she requires.

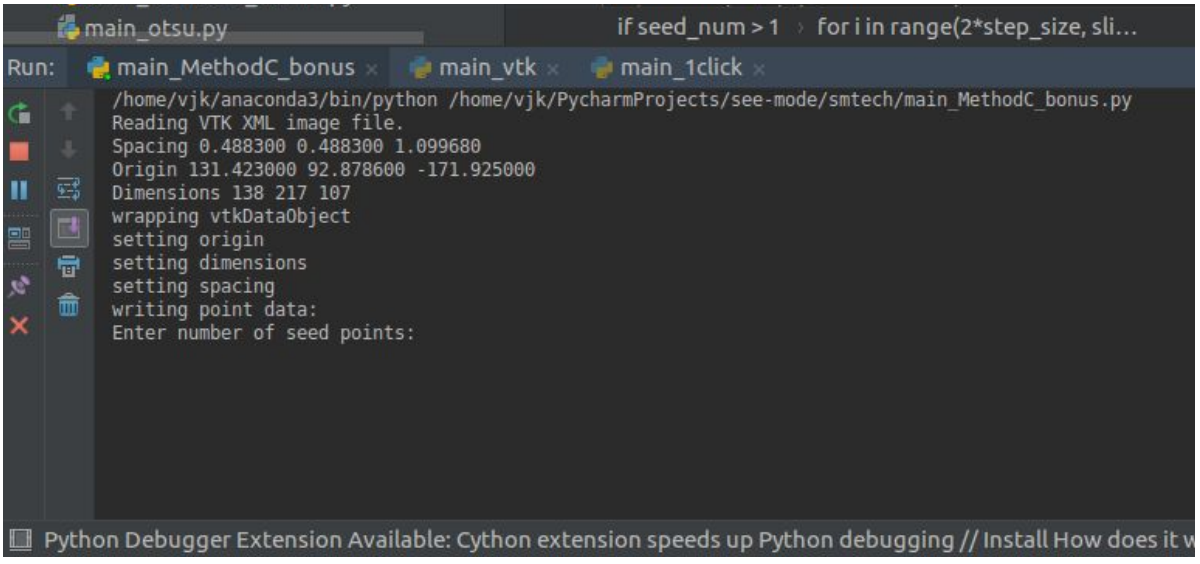

Figure 11 User has to enter the number of seed points

The position coordinate of the seed points were computed using the Origin and Spacing information in the vtkImageData.
*pos_coordinate = Origin + [seed_pt_idx] * Spacing*,
where, *seed_pt_idx* is the index of the seed point in numpy array

vtkImageThresholdConnectivity extracts all the points in the image data which has region with value 1 and is also part of a connected volume based on the seed point or points supplied by user. If the seed points lie in the region of the desired blood vessel then only the volume of that vessel is extracted. All other points are set to 0. This step is followed by conversion of image data to polydata using Marching Cube algorithm and then finally exported as an STL file.

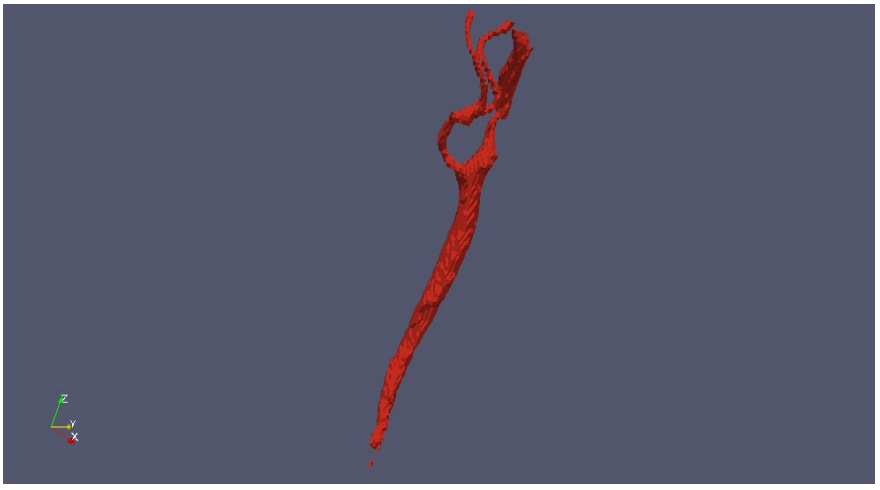

Figure 12 Extracted blood vessel using vtkImageThresholdConnectivity and supplied seed points

Below I show the result when using only one seed point. If the user wants to supply only one seed point, a slice from the middle of the 3D image data is shown to the user to select the seed point. However, the single seed point does not work well consistently. There is too much noise.

Figure 13 Extracted blood vessels using a single seed point (objective 1)

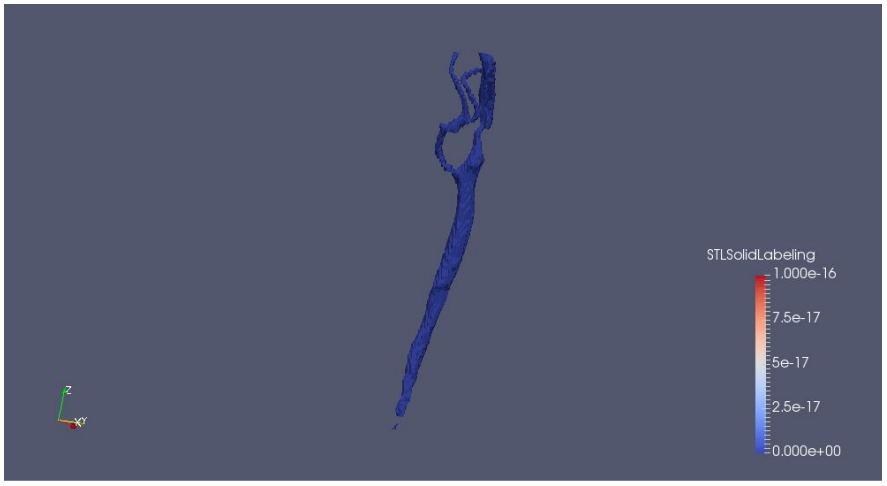

Figure 14 Carotid Artery extracted using the vtkImageThresholdConnectivity from a single seed point

Thoughts on the results

1. The implementation meets the requirement stated in the Tech Task document.
2. Using the vtkImageThresholdConnectivity algorithm VTK library, we are able to eliminate all the noise and get the blood vessel neatly
3. Threshold value estimation from all the seed points can be implemented and will help improve the overall segmentation quality.

Additional Study

1. In my current implementation of Method C, I use only one of the seed points (first one) to compute the threshold value. I tried to use all the seed points to estimate the threshold value by taking the mean of ROIs around the seed points. However, this method was not successful. The estimated threshold was too high that it missed all the blood vessel region
2. For the bonus task, I tried an method using vtkPolyDataConnectivityFilter and then extract the largest region. But this required no seed points as input but it result was not satisfactory. Only a partial portion of the carotid artery was extracted using this method.
3. Just as a trial, I tried to use the Otsu thresholding method to get the threshold value of the image. The method is not well suited for this tech task. But I experimented with it to see if I could use the Otsu threshold as a starting point and use it to estimate the optimal threshold value. But it did not work well.
4. I tested my code with img1.vti - img5.vti. My program worked well for img4.vti and img5.vti. But did not work well for the img1 - img3.vti. The main problem

was in the bonus task. But even the segmenting task was not very good compared to the results from the other images.

5. A **modified version of Method C was implemented** (all_slices folder) whereby the intensity threshold is computed using ROI from all the seed point neighbourhood. Also, the slightly bigger neighbourhood was used for this implementation of [12, 12]. The result were much better for img1.vti - img3.vti especially in the bonus task. I tested the result with 3 seeds and 6 seeds. The STL files are saved in all_slices folder.