Chapter 2

AortaGeomRecon Research and Development

This chapter will discuss the research and development of the 3D Slicer plugin AGR. ~~AortaGeomRecon~~ stands for Aorta Geometry Reconstruction. The main objective of ~~this software~~ is to semi-automatically build the 3D geometry of the aorta from the patient's chest CT scans. The existing methods often involved ~~a~~ extensive manual ~~steps~~ work interspersed with software assistance. An experienced user, who should be a medical domain expert, typically needs to do a minimum of 10 minutes of manual work. Current AGR allows users who have taken the university level anatomy introduction course to set the hyperparameters within half a minute, ~~following~~ by a maximum 2 minutes of execution time.

In this chapter, we first introduce the existing methods ~~with~~ different softwares ~~to perform the aorta segmentation, and~~ discuss them advantages and disadvantages. Then, we demonstrate our segmentaiton algorithm, with a detailed step-by-step explanation of what the algorithm is doing and the result it generates. Finally, we discuss

our experience using the platform 3D Slicer, and demonstrate our development result in 3D Slicer plugin.

2.1 Existing Methods

There are many segmentation software options available to users. We will discuss two popular ~~software~~ options: ITK-Snap and 3D Slicer.

2.1.1 ITK-Snap bubble method

ITK-Snap provides a segmentation method that first requires the user to select multiple voxels with a custom initial size and an expanding size within the volume. This method is referred to as the "bubble method" [6].

Through many iterations, the voxels expand to fill the entire volume of the vessel. As a final step, the user will need to cut off the extra parts of the volume. Figure 2.1 shows ITK-Snap UI executing a segmentation of an aorta.

The advantages of the bubble method is that it is guaranted to produce a correct segmentation ~~result~~ for a valid image. A medical domain expert can manually control the wanted area, and visually observe the segmentation result expanding, shrinking. Moreover, the user can erase the unwanted parts.

The disadvantages of this method is that the operations described above are complicated. They are easier to say then do. An operator who has previous experience building the geometry with this method still needs about 20 minutes of manual work to build a new aorta geometry. Plus, ITK-Snap software can only read VTK (The Visualization Toolkit) file; therefore, the chest CT scans that are usually Digital Imaging

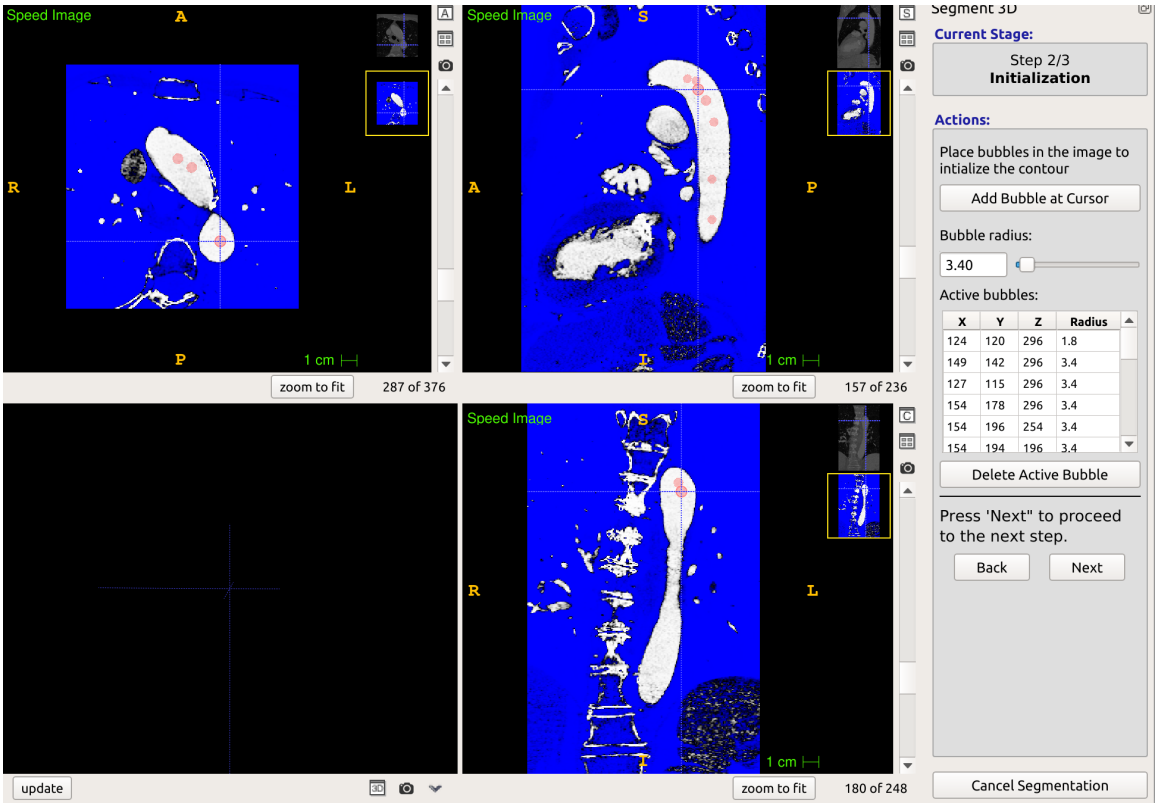


Figure 2.1: ITK-Snap's Bubble Segmentation UI [23]

and Communications in Medicine (DICOM) format [7], needed a manual conversion before using this software and its segmentation method.

2.1.2 3D Slicer threshold segmentation

3D Slicer is another well-known medical image processing software for academic uses. 3D Slicer provides multiple segmentation methods [1]. One of the quickest and easiest to use is the intensity based segmentation.

This method first lets users select a small area that belongs to the wanted area on a 2D plane (Axial, Sagittal, and Coronal). 3D Slicer read the pixels' intensity of the surrounding area, and segments based on the intensity threshold. Any pixel's

intensity that is within the range will be segmented as the segmentation result, as demonstrated in Figure 2.2.

Like the bubble method, this method often reads extra volume, and requires the user to cut the unwanted parts. A YouTube video shows an experienced user who gets the aorta 3D geometry with 8 minutes of manual works.

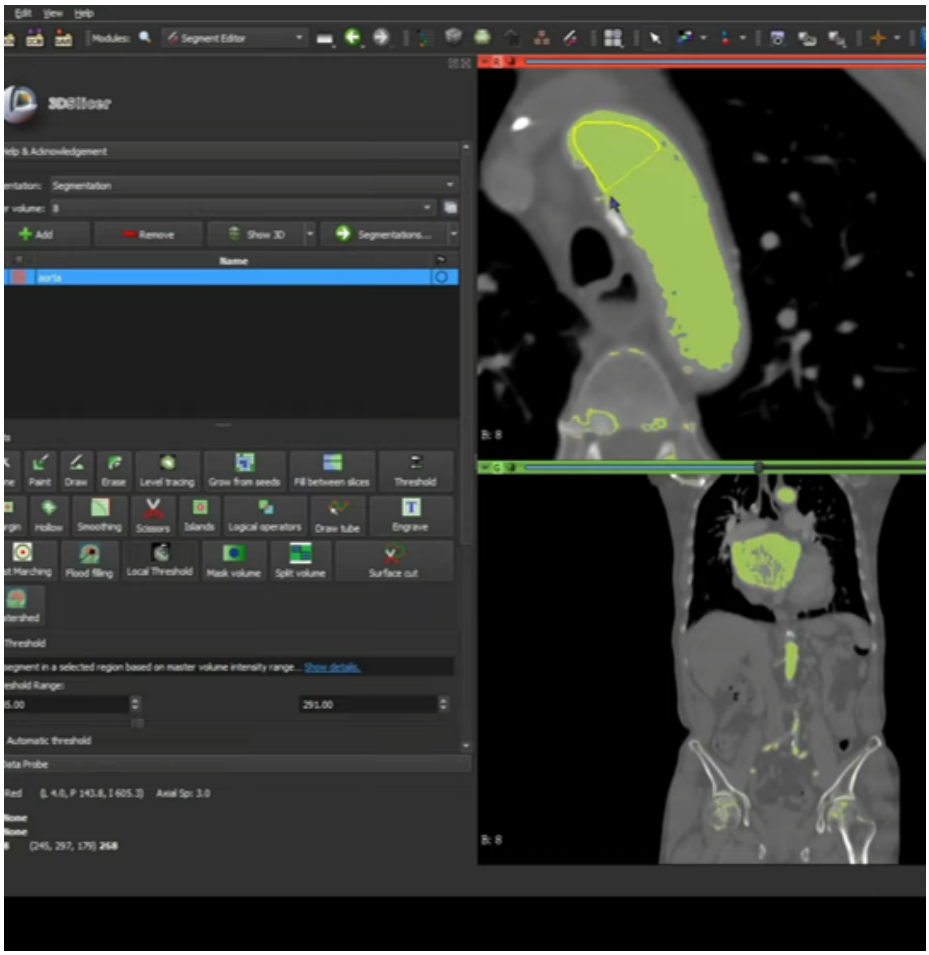


Figure 2.2: 3D Slicer Built-in Segmentation Method [11]

What are the disadvantages of this method? Besides the time, is there a danger of holes in the segmentation? gaps

2.2 Segmentation Algorithm

This section introduces the key concepts of the implementation on the segmentation algorithm. The algorithm is developed in Python with the external libraries SimpleITK and NumPy. The algorithm builds the 3D Aorta geometry by doing segmentation on each axial slice. The logic behind segmenting each slice from the axial view, is that there is one or two circles that is edged bounded in each axial plane view. Using this information, and given an initial aorta center coordinate, the algorithm continuously segments each axial slice circles closest to the previous aorta center coordinates. Finally, some hyperparameters tuning can let the algorithm pick up the pixels that were missing but belongs to the part of the aorta.

In this section, we first introduces the contexts for exertal libraries used in the implementation of the algorithm. We show the parameters and hyperparameters of the algorithm. Then we explain the algorithm workflow with a step-by-step demonstration.

2.2.1 Background

SimpleITK is an open-source multidimensional image analysis library developed by the Insight Toolkit community for the biomedical sciences and beyond [3][15]. NumPy is the fundamental package for scientific computing with Python, especially for the performance of multidimensional array processing [9]. The algorithm will use functions from these two libraries for image processing and multidimensional array processing. For example, the algorithm segments each slice with ThresholdSegmentationLevelSetImageFilter from SITK.

The algorithm works best with the chest volume cropped to a rectangular prism

that contains the aorta and parts of the other organs such as the backbone, blood vessels, and the heart. This can be done with 3D Slicer and its built-in modules, Volume rendering and Crop Volume, or researched by the user to find the starting point and the size to crop.

another means familiar to the user for finding

2.2.2 Parameters

At the beginning of the algorithm, the user inputs two integer coordinates indicating the position of the descending aorta and ascending aorta center on a single slice. The yellow dots in Figure 2.3 shows an example of the aorta seeds. These seeds will be updated by the algorithm after processing each axial plane.

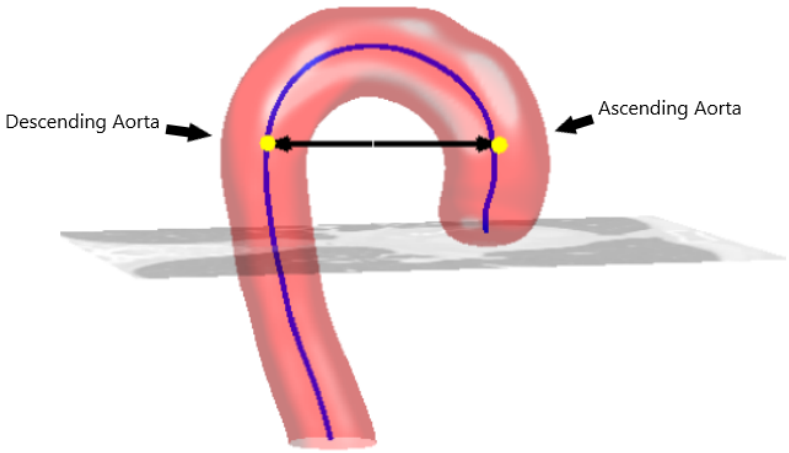


Figure 2.3: The aorta seeds [13]

The following list of hyperparameters that can be tuned to get the best segmentation result:

- The stop limit, which controls the stop condition

- The threshold coefficient, which controls the segmentation acceptable intensity range
- The kernel size, which controls the label image circle size
- The threshold Segmentation Level Sets Image Filter parameters, including:
  - The RMS error
  - The maximum iteration
  - The curvature scaling
  - The propagation scaling

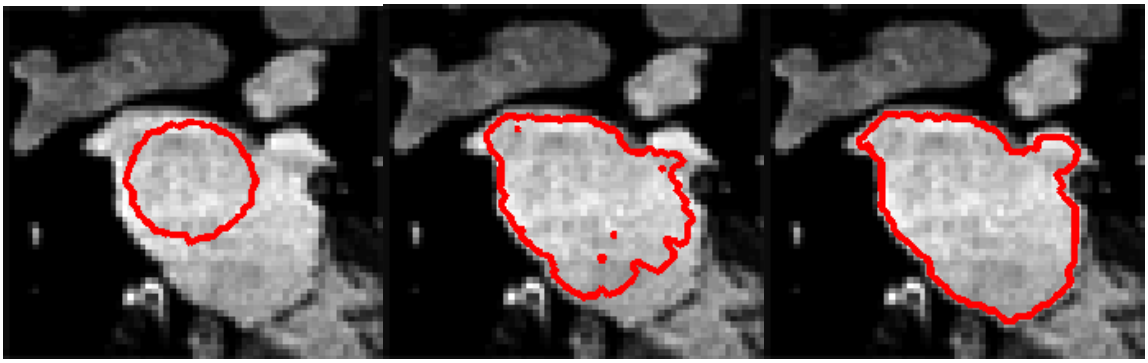
One of the most important parameters is the threshold coefficient. Since the algorithm segments based on the intensity of the gray scale pixels, decreasing the threshold coefficient decreases the acceptable range of the pixels, and vice-versa.

2.2.3 Algorithm Overview

When the user has selected the aorta seeds, the plane where the aorta seeds are located is the initial plane. From this plane towards the bottom (toward the feet) is the inferior direction. The upward direction (toward the head) is the superior direction. This algorithm segments each slice with SITK::ThresholdSegmentationLevelSetImageFilter. The principles of this image filter can be explained with two terms: Level sets segmentation method, and a threshold range that defines the intensity of the acceptable pixel.

Level Sets are an important category of modern image segmentation techniques

based on partial differential equations (PDE), i.e. progressive evaluation of the differences among neighboring pixels to find object boundaries. The pictures 2.4 demonstrate an example of how the Level Sets method work on finding the region of the heart. It starts with a seed contour that is within the region of interest, then by finding the gradient based on the contour line, the segmentation result will propagate towards the outside of the region, until the maximum difference between the neighboring pixels are reached.



(a) Init image (b) Expanding (c) Result

Figure 2.4: Level Sets Segmentation

2.2.4 The steps to segment a single slice

In the following section, we will present each step to segment a single slice. These steps are applied in segmentations in both the superior and inferior directions. There is a difference in the stopping condition, which we will elaborate in the following section 2.2.4.6.

2.2.4.1 Create A Label Map

The algorithm uses SITK::BinaryDilateImageFilter to perform binary dilation to generate a circle-like shape around the center coordinates (user input's for the first slice and calculated by the algorithm for the rest of the slices). Each pixel within this shape will be labeled as a white pixel (value of 1), and the rest of the pixels are labeled as black pixels (value of 0).

The generated result is the label map image, and we will use it in the next few steps. The size of the circle-like shape is determined by the kernel size (user's input). The Figure 2.5 shows an example of generated label map image (the green parts) overlay over the original slice.

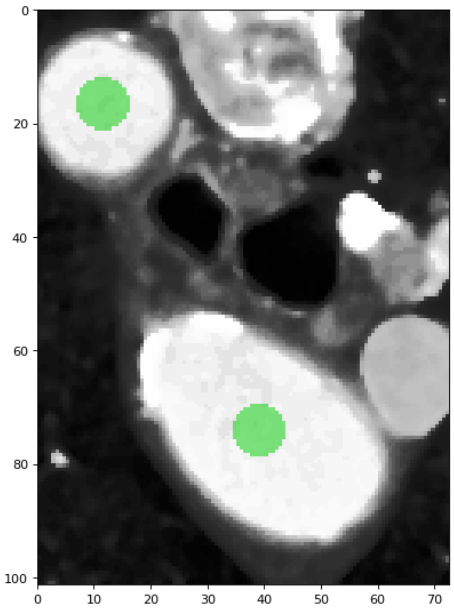


Figure 2.5: A Label Map

2.2.4.2 Create A Distance Map

With SITK::SignedMaurerDistanceMapImageFilter, the algorithm creates another image, the Euclidean distance transform of the label image from previous step. This is used as a contour line that helps build the gradient mentioned in Level sets. The Figure 2.6 shows an example of distance map.

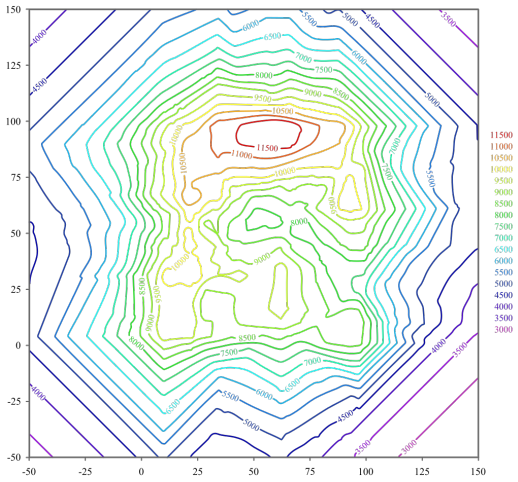


Figure 2.6: A Distance Map

2.2.4.3 Calculate A Threshold Range

By using SITK::LabelStatisticsImageFilter, the algorithm gets the mean and the standard deviation of the intensity values of the pixels that were labeled as the white pixel in the label map. The algorithm uses the threshold coefficient to calculate the lower and upper threshold to be used in the next step.

```
intensity_mean = self._stats_filter.GetMean(
    PixelValue.white_pixel.value)
std = self._stats_filter.GetSigma(PixelValue.white_pixel.value)
lower_threshold = (intensity_mean - self._threshold_coef*std)
upper_threshold = (intensity_mean + self._threshold_coef*std)
self._segment_filter.SetLowerThreshold(lower_threshold)
self._segment_filter.SetUpperThreshold(upper_threshold)
```

Figure 2.7: Code That Shows How To Calculate The Threshold Range

2.2.4.4 Segment A Single Slice

With SITK::ThresholdSegmentationLevelSetImageFilter, the seed image calculated in step 2.2.4.2, and the lower and upper threshold value calculated in step 2.2.4.3, the algorithm performs segmentation and generates a segmented slice. The Figure 2.8 shows an example of generated segmented slice (the green parts) overlay over the original slice.

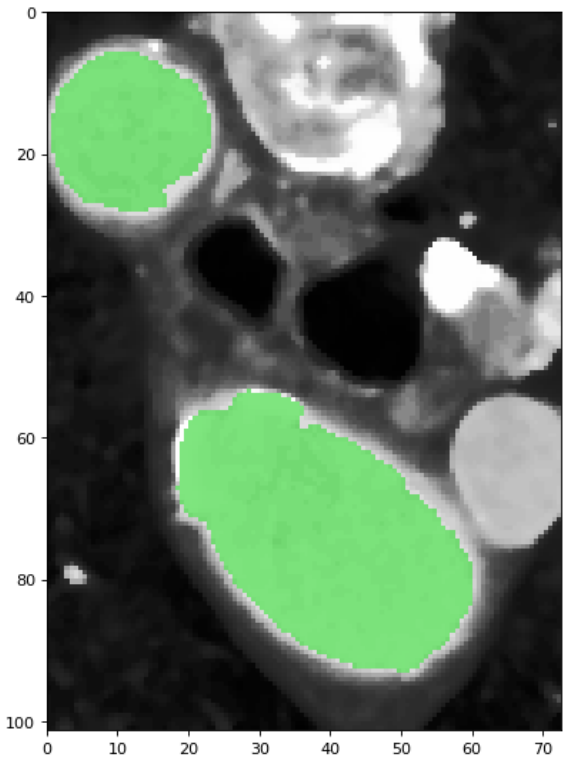


Figure 2.8: A Segmented Image On Top Of The Original Slice

2.2.4.5 Calculate New Centroids

By comparing each pixel segmented as aorta to the previous descending centroid and the previous ascending centroid, the algorithm use the positions of the points closer

to the previous descending centroid to calculate the new descending aorta centroid, and vice-versa for the ascending aorta centroid. However, at certain points during the segmentation in the inferior direction, the slice might reach the end of the ascending aorta, the aortic root, the algorithm will stop using and calculate ascending aorta centroid and only computes descending aorta centroid for the slices afterward.

2.2.4.6 Verify Segmentation Result

There are two main stopping conditions for verifying segmentation result, one condition for the segmentation in the inferior direction and the other one for the segmentation in the superior direction. Stopping limit is a user defined parameter to control the algorithm, to calculate the condition with the centroids position and the standard deviation.

In the inferior direction, if the new ascending aorta centroid that is closest to the previous ascending aorta center is reaching the distance limit, then the algorithm will stop and consider taking the new centroid closer to the ascending aorta. In other words, only 1 centroid will be used for descending aorta segmentation.

In the superior direction, if the standard deviation of the initial label map and the segmentation result label map have larger differences than the limit, the algorithm will stop processing segmentation for the rest of the slices. For example, assume that the standard deviation of the initial label image is 20, and the standard deviation of the segmentation label image is 40, with stop limit of 10, the program will halt immediately.

2.2.5 Algorithm Summary

Given two integer coordinates, ascending aorta centroid and descending aorta centroid, the algorithm set the inputted plane as the initial plane. From the initial plane to the bottom (toward the feet) plane, the algorithm calculates a label map with two centroid coordinates and kernel size, calculates a distance map with the label map, calculates a threshold range with the label map's selected pixels, performs segmentation, calculates new centroid coordinates, and verifies the segmentation result in case that it reaches the stop condition. Once the algorithm finishes the segmentation in the inferior direction, the algorithm works from the initial plane to the top (toward the head) plane, repeating the similar steps. Each segmentation result slice is stored in a SITK's image, which supports the conversion to VTK file or DICOM file. Figure 2.9 demonstrates a segmentation result, rendering in 3D Slicer.

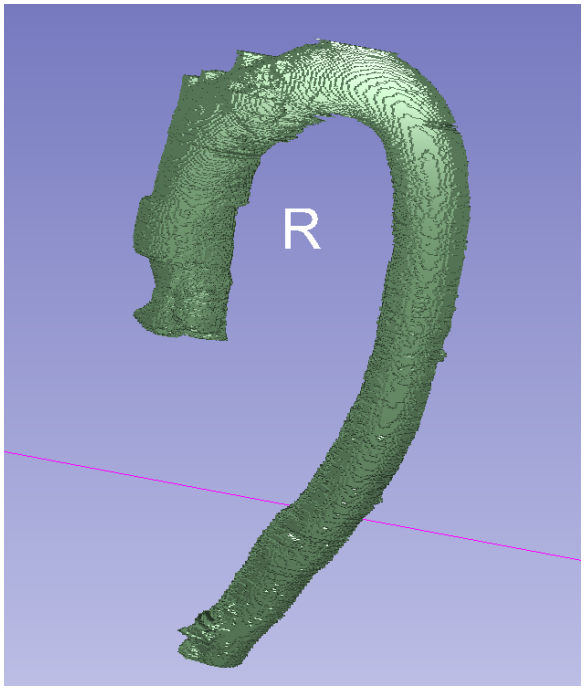


Figure 2.9: Segmentation Result

2.3 3D Slicer Extension Development

The project has started with a simple segmentation program build in Jupyter Notebook [12]. When getting a new patient's data, the user will need to investigate the chest CT scans using another software (3D Slicer, ITK-Snap), to get the readings such as coordinates and size to crop (the coordinates of the yellow dots shown in Figure 2.3).

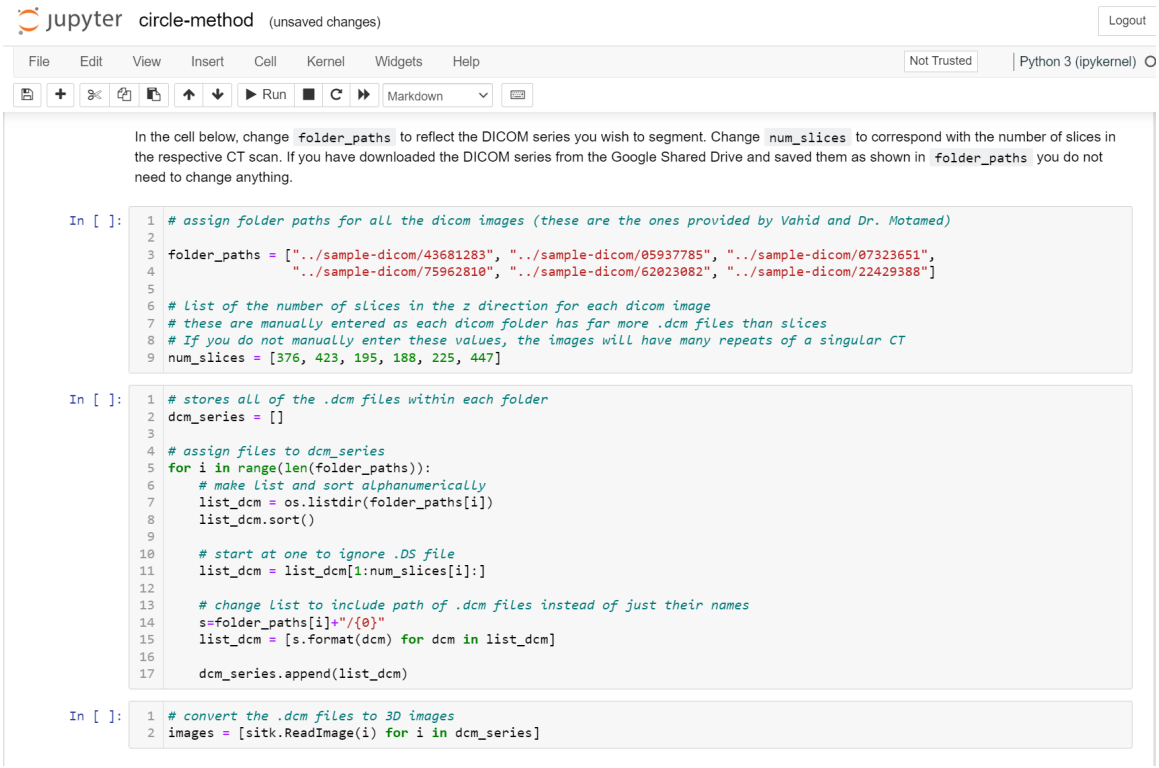


Figure 2.10: Jupyter Notebook Researched by Kailin Chu

Originally, the parameters entered by the users, and many other values are hard-coded in the Jupyter Notebook. To improve the usability of the AortaGeomRecon (reduce the amount of time for user inputs and execution), we implemented an extension module on 3D Slicer.

3D Slicer is an open-sourced medical image processing software for research. 3D Slicer provides useful modules such as Crop Volume module and Volume Rendering module that easily crop any volume. 3D Slicer is highly modulable with Python scripting to control the extension module sequence, and QT designer to generate Graphical User Interfaces.

3D Slicer supports modularization with an extension. An extension can compose multiple modules, where each module is dedicated to solve a sub-problem.

2.3.1 3D Slicer's data structure

3D Slicer's Data Structure can be divided into two categories. The Node data structure store large data such as DICOM with a Volume Node, Volume rendering Region of Interest Node, Label Map Volume Node. The parameters are stored as string from the UI component of the module. Every data stored in 3D Slicer can be accessed by the 3D Slicer's Widget Class and Logic Class for further processing.

3D Slicer stores all the above data in a scene object, which is also referred as MRMLscene file, on the higher level data format. 3D Slicer can load any MRMLscene file, this allowed user to retrieve all the data nodes and parameters. On the other hand, 3D Slicer has a special input module, the DICOM database allowed user to store DICOM metadata in 3D Slicer.

2.3.2 3D Slicer's scripted module

Every ScriptLoadableModule in 3D Slicer have a Widget Class and a Logic Class. The Widget Class is used to initialize the extension module's UI component, and the parameters tied to the UI component. The module's Logic Class is used to perform

the processing of the data. In the Logic Class, we initialize an AortaGeomRecon Segmenter object with the attributes set to the parameters reading from UI component, which are inputs by the user. After completing the segmentation with Segmenter object, we convert the SimpleITK image object to a volume node corresponding in 3D Slicer, which allow the user to visualize the segmentation result.

2.3.3 AortaGeomReconDisplayModule

In this section, we demonstrate the implementation details of the 3D Slicer plugin AGR. We first demonstrate the module's Graphical User Interface, then discuss on the module's logic and workflow.

2.3.3.1 Graphical User Interface

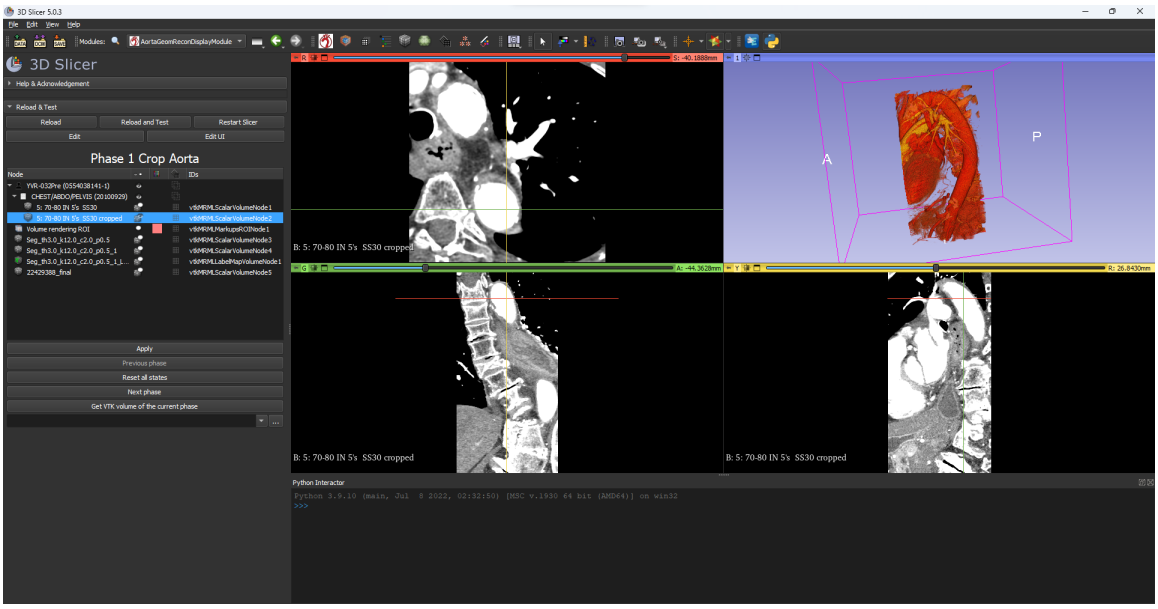


Figure 2.11: 3D Slicer UI

3D Slicer separate the UI into two parts. From the Figure 2.11, we can see that the four windows on the right side of the UI are used to visualize a volume. The left side shows the SubjectHierarchyTreeView which the module already stored many data nodes. The first node is the DICOM patient data with the chest CT scans stored as a volume, and the cropped volume is generated with Crop Volume Module. There is a Volume rendering ROI node and several ScalarVolumeNode which are the generated segmentation volume with different parameters.

The left side is the module UI. This part is designed and implemented differently based on the requirements of the modules. The Figure 2.12 shows the module UI that the AGR implemented, where each parameter stored to be passing to the algorithm class.

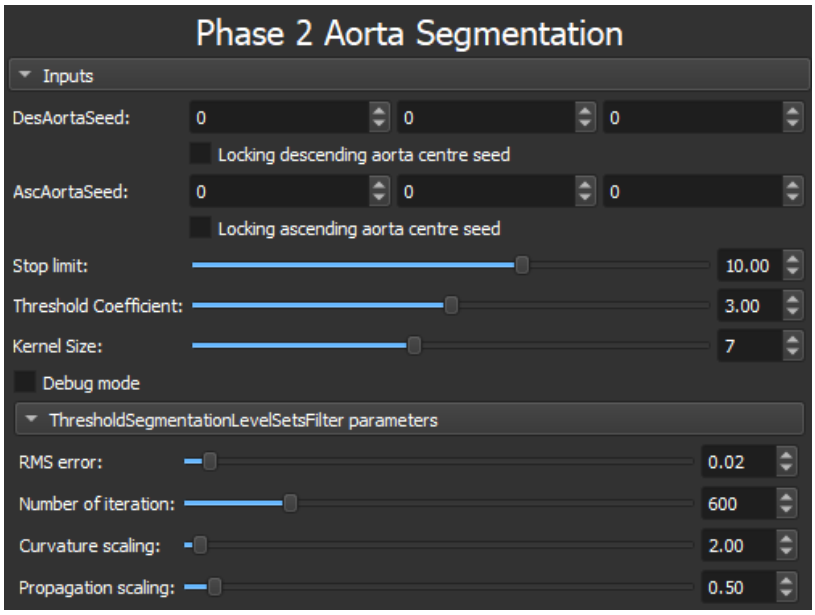


Figure 2.12: AortaGeomRecon Module UI

2.3.3.2 Module's Workflow

When the user first starting 3D Slicer and click on the AGR module, this warning message and tips appears in the module UI. The user must click on the confirm button below to proceed into the next steps. This warning message is an evidence for the assurance case, which I will explain in the next chapter.

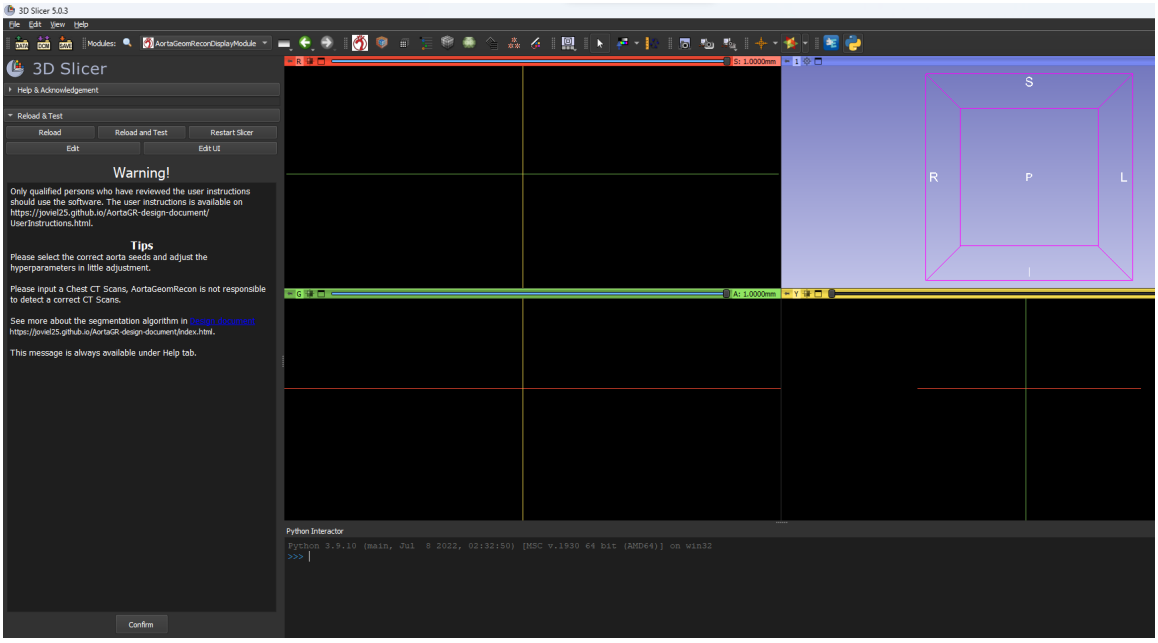


Figure 2.13: AortaGeomRecon Warning message

In the next step, assuming that the user has already read a DICOM image of the patient's chest, the user is asked to generate a cropped volume using the 3D Slicer's Volume Rendering module and the Crop Volume module. In this phase, the module UI displays only a SubjectHierarchyTreeView where the large data node are shown in this view. After generated a cropped volume, the apply button is enabled and the user can proceed to the next step.

In phase 2 aorta segmentation, the user is asked to input the parameters to perform

the segmentation. The module UI is same as the Figure 2.12. The necessary inputs are the two aorta seeds. Without any value for these two inputs, the module will not allow the user to generate a segmentation result. One of the advantages of using 3D Slicer is the interactive UI that supports reading coordinates on the volume interactively. On the right side of the Figure 2.11, we see a crossed intersection pointing to parts of the aorta, this intersection point allows the developer to read the coordinates. Moreover, we were able to automatically pick up the coordinates in real-time in the coordinate widget. As the user moving the intersection, we get the coordinate readings. The Figure 2.12 also demonstrates that the user can lock a seed, so the program stops picking up newest coordinate from the intersection point.