BUILDING AN ASSURANCE CASE FOR AORTAGEOMRECON SOFTWARE

BUILDING AN ASSURANCE CASE FOR AORTA GEOMETRY RECONSTRUCTION SOFTWARE

BY

JINGYI LIN, M.Eng.

A Report

submitted to the Department of Computing and Software

and the School of Graduate Studies

of McMaster University

in partial fulfilment of the requirements

for the degree of

Masters of Engineering

Masters of Engineering (2023) McMaster University

(Department of Computing and Software) Hamilton, Ontario, Canada

TITLE:	Building an Assurance Case for Aorta Geometry Reconstruction Software
AUTHOR:	Jingyi Lin M.Eng. (Computing and Software CRP), McMaster University, Hamilton, Canada
SUPERVISOR:	Smith Spencer
NUMBER OF PAGES:	xii, 25

Abstract

Assurance cases has been proven to be effective developing a real-time system software.

Another domain that requires the high standard correctness, completeness is medical software.

Throughout the development of the Aorta Geometry Reconstruction software, we implicitly listed the evidences that are essential to build our confidence in the software for assurance cases, build the artifact and the evidences simultaneously.

Finally, we present this software with the list of the evidences built for assurance cases, to show that the assurance cases can apply well on the medical software

Your Dedication

Optional second line

Acknowledgements

Acknowledgements go here.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

$A \leq B$	A is less than or equal to B

Definitions

Challenge	With respect to video games, a challenge is a set of goals presented to the player that they are tasks with completing; challenges can test a variety of player skills, including accuracy, logical reasoning, and creative problem solving

Abbreviations

AortaGeomRecon	3D Slicer's extension module, Aorta Geometry Reconstruction
SRS	Software Requirements Specification

MG	Module Guide
SITK	SimpleITK

Declaration of Academic Achievement

The student will declare his/her research contribution and, as appropriate, those of colleagues or other contributors to the contents of the thesis.

Chapter 1

Introduction

This chapter includes an introduction for AortaGeomRecon assurance cases study. In section 1.1, we are dicussing the objective of this case study, and this documentation. In section 1.2, we are explaning some terms and definitions used throughout this document. Section 1.3 discussed about the problem statement, and finally section 1.4 gives the brief outline for this document.

1.1 Objective

In this study, we present the result of applying of assurance cases in a developing medical software to build the stakeholders' confidence in this software.

1.2 Background

Aorta is the largest artery that carries blood from the heart to the circulatory system. It has a cane-liked shape with Ascending aorta, Aortic arch and Descending aorta.

Unlike the real cane stick, the descending aorta might not be straight for each patient.

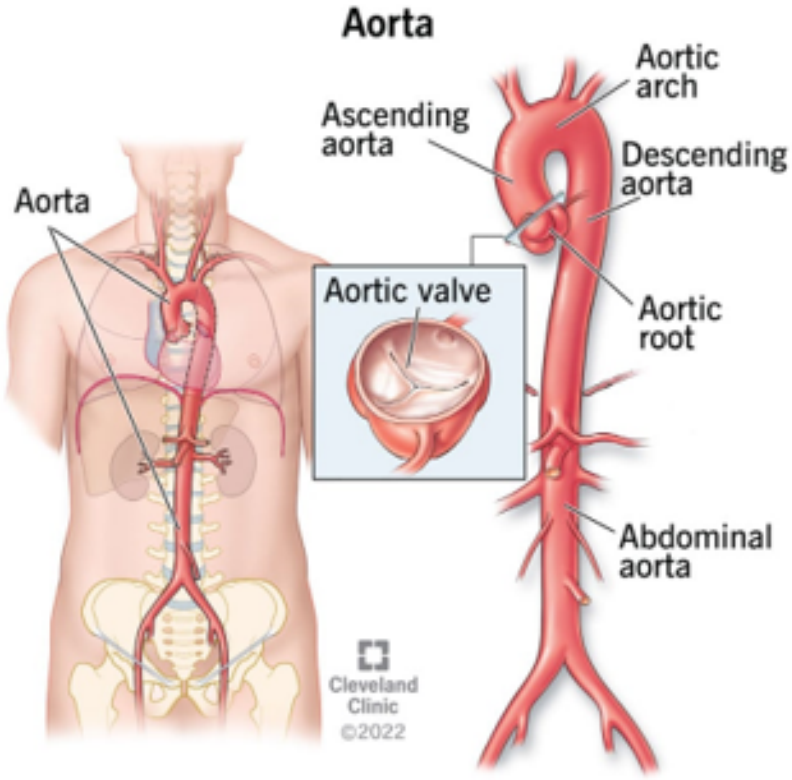

Figure 1.1: Aorta

Aorta segmentation in CT scans is important for:

- Coarctation of the aorta
- Aortic calcification quantification
- To guide the segmentation of other central vessels.

Assurance cases

3D Slicer

Image Processing

1.3 Problem Statement

The main goals of this project is building a software that can quickly build the 3D geometry of the Aorta from CT Chest scans, while applying the assurance cases for this academic medical software. This project shows the assurance cases can indeed help build up our confidence in the medial software in general, because medical software like real-time system software, needed completeness and correctness.

Build Software and Assurance cases for this software. Start with a list of Functional and Non-Functional requirements.

1.4 Thesis Outline

Chapter 2

AortaGeomRecon Research and Development

This chapter will discuss about the research and development of the AortaGeomRecon.

AortaGeomRecon stands for Aorta Geometry Reconstruction. The main objective of this software is to semi-automatically build 3D geometry of the Aorta from the patient's chest ct scans. The existing methods are often involved of extensive manual works by using a software with many steps. An experienced user, who might be a medical domain expert, needs to do a minimum of 10 minutes of manual works.

The implementation till the date of this report can let the users who have the user characteristics described in SRS (Lin, 2023) get the Aorta 3D geometry with only a few hyperparameters which can be set within half a minute, and the result requires maximum 2 minutes of execution time.

2.1 Existing Methods

There are many segmentation software available to the users, we will discuss the two builtIn methods in two softwares, ITK-nap and 3D Slicer.

2.1.1 ITK-Snap bubble method

ITK-Snap provides a segmentation method that first let user to select multiple voxels with a custom intial size and expanding size within any volume. We refer this method as "bubble method".

Through many iterations, the voxels expand to fill the entire volume, finally user will need to cut the extra part of the volume. This Figure 2.1 shows the ITK-Snap UI executing the segmentation.

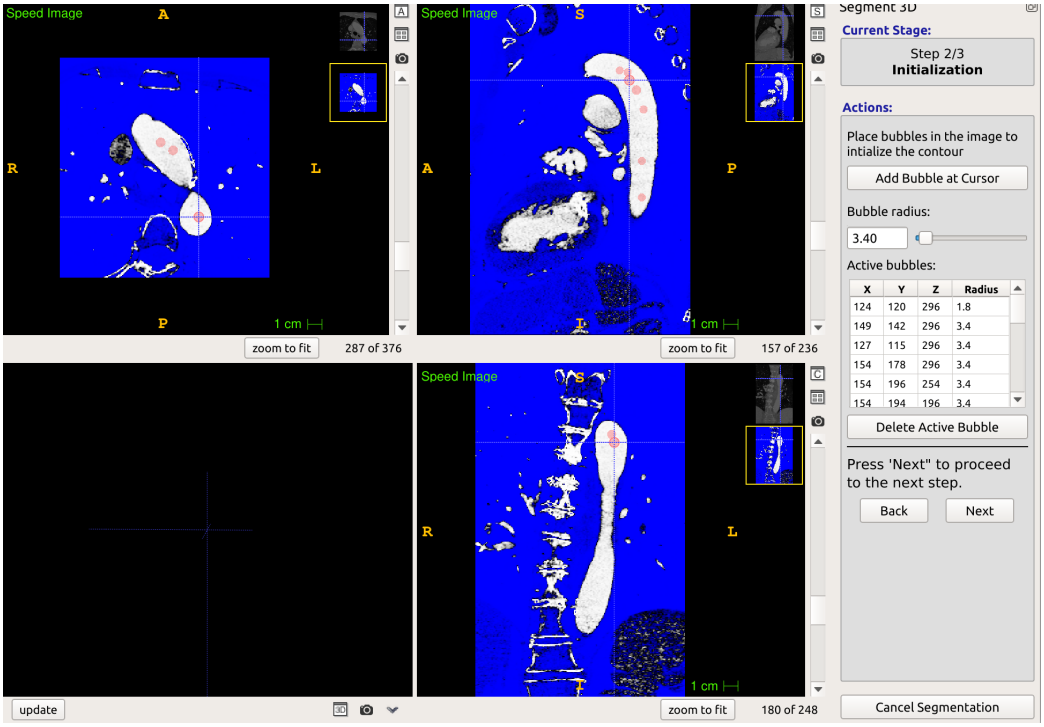

Figure 2.1: ITK-Snap's Bubble segmentation method

The advantages of the bubble method is that it guarantes to produce a correct the segmentation result. A medical domain expert can manually control the wanted area, and visually observing the segmentation result expanding, shrinking and the user can erase the unwanted part.

The disadvantages of this method is that the operations described above are complicated. Eeasier to say then do, an opeartor who has previous experience building the geometry with this method still needed 20 minutes of manual work building a new aorta geometry. Plus, ITK-Snap software can only read VTK file, therefore the chest CT scans that are usually DICOM, 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. 3D Slicer provides multiple segmentation methods, and one of the quickiest and easiest to use is the intensity based segmentation.

This method first let user 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 segment based on the intensity threshold. Any pixels's intensity that is within the range will be segmented as the segmentation result.

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

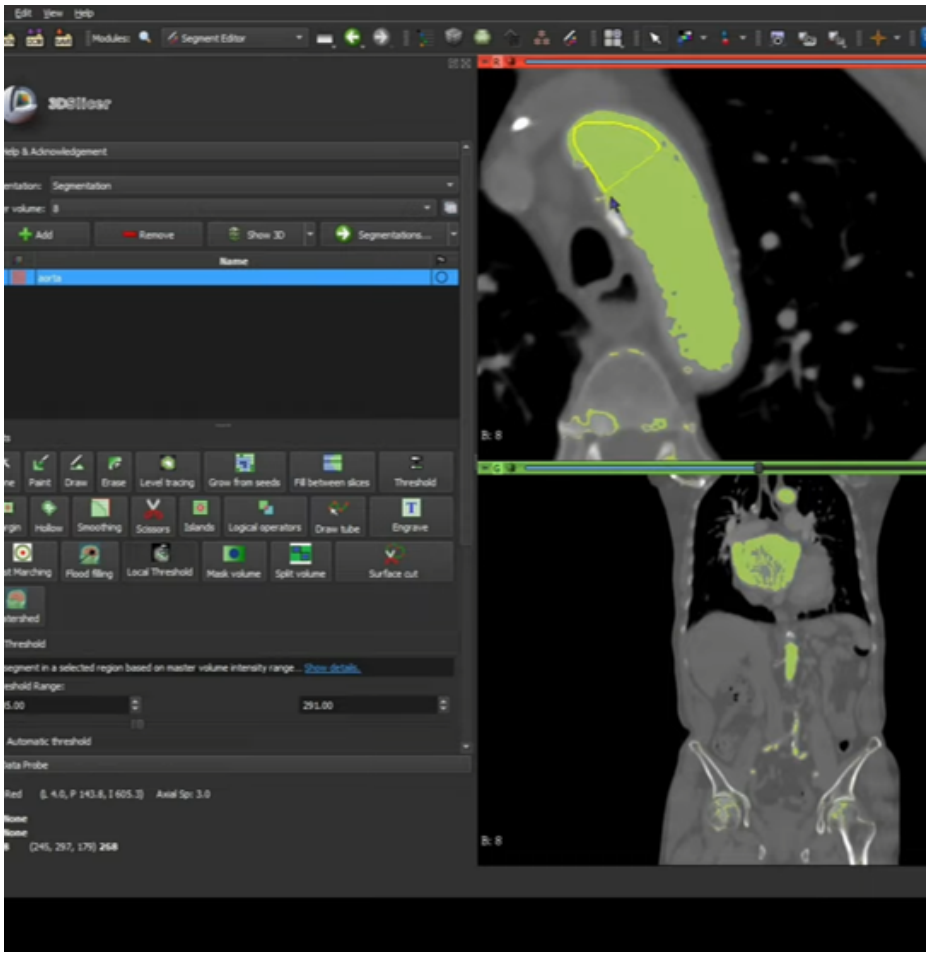

Figure 2.2: 3D Slicer BuiltIn Segmentation Method

2.2 Segmentation Algorithm

This section introduces the key concepts of the implementation on the segmentation algorithm. The algorithm is developed in Python, by using external libraries including SimpleITK and Numpy.

2.2.1 Background

SimpleITK is an open-source multi-dimensional image analysis library Developed by the Insight Toolkit community for the biomedical sciences and beyond. NumPy is the fundamental package for scientific computing with Python, especially for the performance on multi-dimensional array processing. The algorithm will use functions from these two libraries for image processing and multi-dimensional 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 builtIn modules, Volume rendering and Crop Volume, or researched by the user to find the starting point and the size to crop.

At the beginning of the algorithm, the user inputs two integer coordinates indicating the position of the descending aorta and ascending aorta centre on a single slice. The yellow dots in Figure 2.3 shows an example of the aorta seeds.

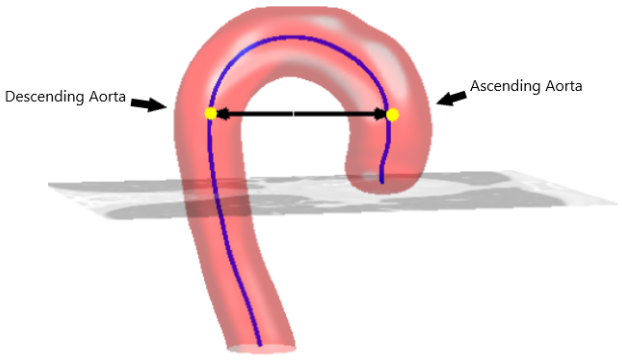

Figure 2.3: The aorta seeds (Kurugol et al., 2012)

2.2.2 Parameters

The main parameters inputs from the users are:

- The centre coordinates of Descending Aorta and Ascending Aorta locate on the same axial plane
- The stop limit which controls the stop condition
- The threshold coefficient which controls the segmentation acceptable intensity range
- The kernel size which controls the init size
- The threshold Segmentation Level Sets Image Filter parameters, including:
  - The rms error
  - The maximum iteration
  - The curvature scaling
  - The propagation scaling

2.2.3 Algorithm Overview

When the user has selected the aorta seeds, the plane where the aorta seeds located is the initial plane. From this plane towards the bottom (the feet) is inferior direction. On the other hand, it is superior direction. Instead of processing the entire volume, the algorithm will process each axial plane

2.2.4 The steps repeat for each axial plane

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 centre 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 2.4 shows an example of generated label map image (the green parts) overlay over the original slice.

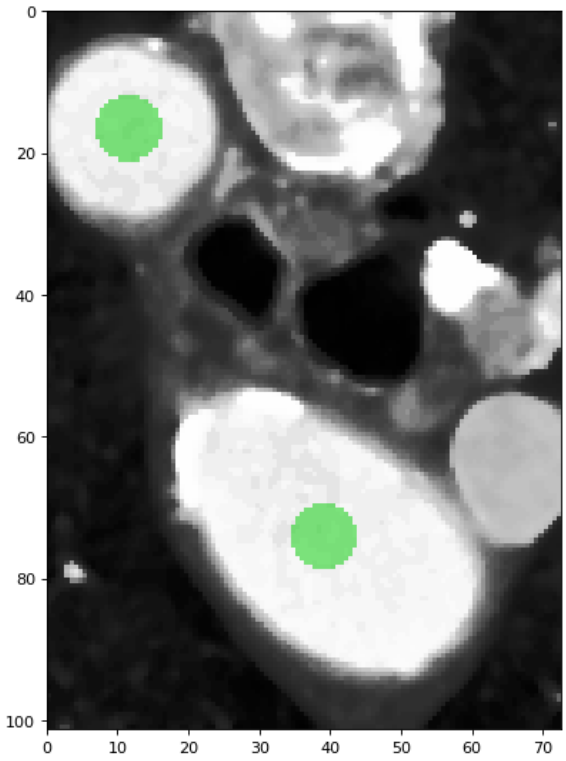

Figure 2.4: A label map

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 2.5 shows an example of distance map.

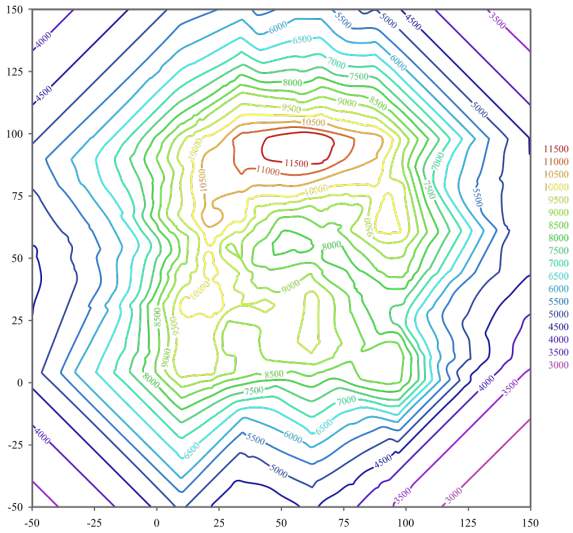

Figure 2.5: A distance map

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 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 2.6: Code snippet shows the intensity threshold calculation

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 generated a segmented slice. The Figure 2.7 shows an example of generated segmented slice (the green parts) overlay over the original slice.

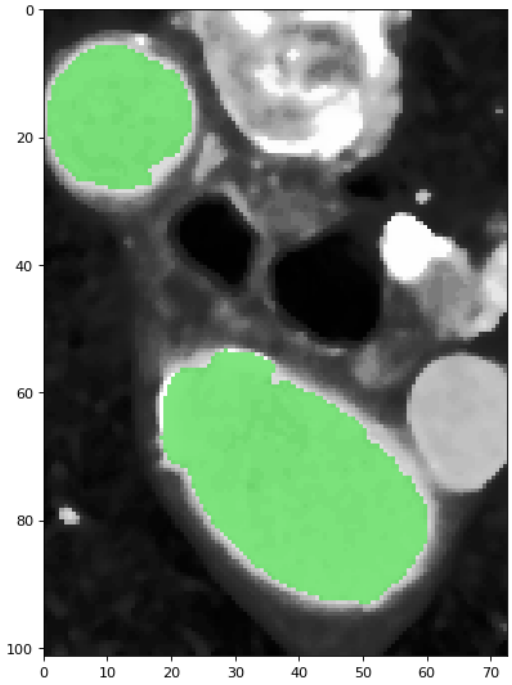

Figure 2.7: A segmented image on top of the original slice

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 new descending aorta centroid, and vice-versa for the ascending aorta centroid. However, at certain point during the segmentation in inferior direciton, the slice might reaches the end of the ascending aorta, where the voxels belong to the part of the heart. The algorithm will stop using ascending aorta centroid and only computes descending aorta centroid for the slices

afterward.

2.2.4.6 Verify segmentation result

There are two main stop conditions for verifying segmentation result, one condition for the segmentation in inferior direction and the other one for the segmentation in superior direction. Stop limit is an user defined parameter to control the algorithm.

2.2.5 Algorithm Summary

After repeating the steps explained above,

2.3 3D Slicer Extension Development

The project has started with the a simple segmentation algoritm build on the jupyter notebook. 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 of the starting voxel and the size crop the volume, and get the index of the aorta seed. This Figure 2.3 shows an example of the aorta seeds.

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 a open-sourced medical image processing software for research. 3D Slicer provides useful modules such as Crop Volume module and Volume Rendering module, and it's highly modulizable with Python Scripting to control the extension module sequence, and QT designer to generate Graphical User Interface.

3D Slicer supports modulization with an extension. An extension is a software

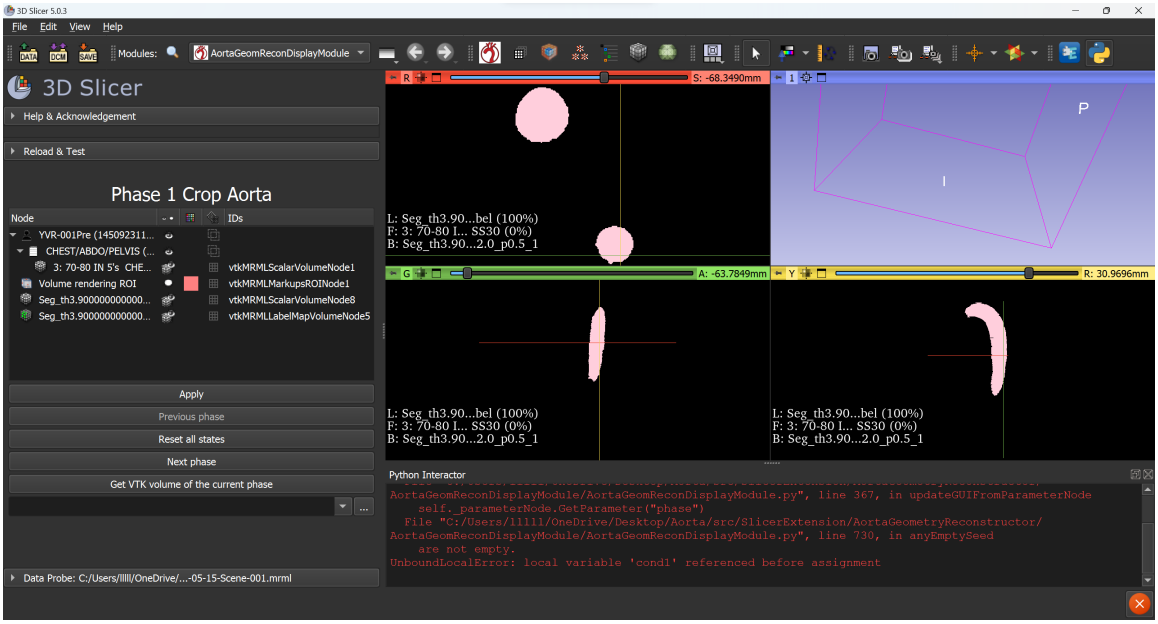

Figure 2.8: AortaGeomRecon UI phase 1

bundle that includes every modules to solve a specific medical image problem. An extension can compose multiple module, 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: Node for storing large data such as DICOM with a Volume Node, Volume rendering Region of Interest Node, Label Map Volume Node, and parameters are read from extension's module UI and are stored as string. Every data stored in 3D Slicer can be accessed by the 3D Slicer's Widget Class and Logic Class for further processing.

On a higher level, 3D Slicer stores all the above data in a scene object. 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

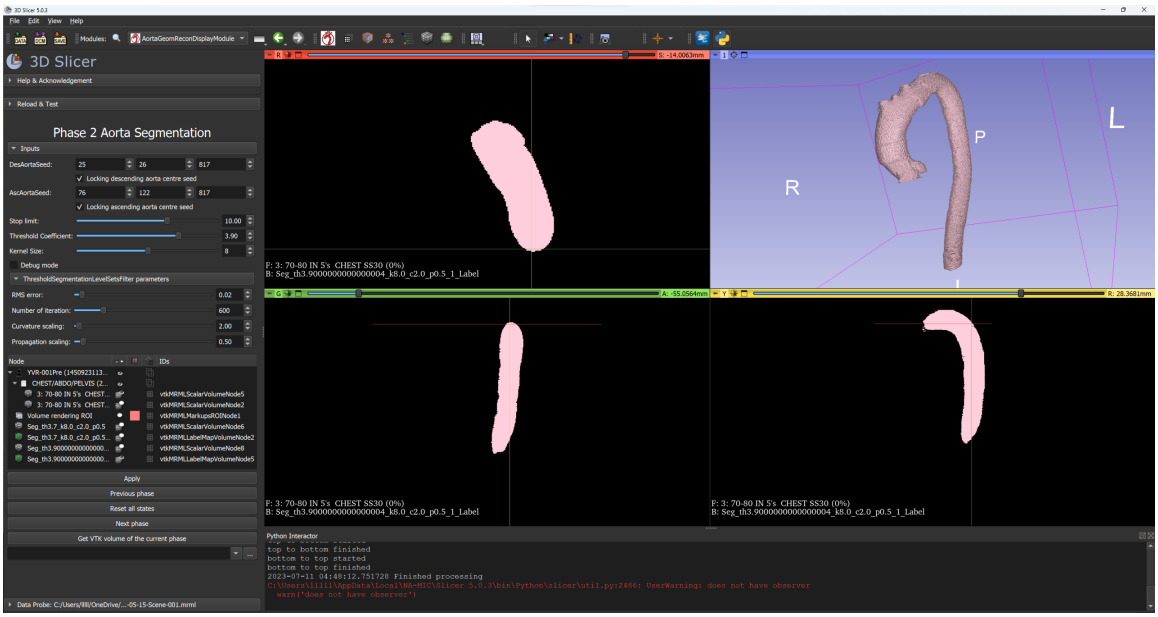

Figure 2.9: AortaGeomRecon UI phase 2

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 compoenent. 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.

2.4 GitHub and Workflows

This project uses GitHub for version control.

GitHub issues tracker use to keep track the items to work on throughout the development of the project

GitHub Project is used for dividing a large issue into smaller tasks, expected date of the completion. This is useful for project management.

GitHub Workflow is a great tool for Continuous Integration tests.

We uses GitHub Workflow for Linter and Continuous Integration tests.

Chapter 3

Assurance Cases and Selected Evidence for AortaGeomRecon

3.1 Assurance Case Development

3.2 Assurance Case for Software Specification Requirements

- (Lin, 2023)
- Goals
- Assumptions
- Data Definitions
- Instance Model

- Functional Requirements
- Non-Functional Requirements
- Traceability Matrix Between Different Sections
- Traceability Matrix Between Requirements and Other Sections

3.3 Assurance Case for Implementation

- Design Document
  - Sphinx - Python Documentation Generator
  - Comments in the source code
  - Algorithm Overview - Explanation and Program Workflow
  - Glossary
- Module Guide
  - Module Decomposition
  - Traceability Matrix between Modules and Requirements
  - Traceability Matrix between Modules and Source Code
- Test Case
  - GitHub Workflow
  - Continuous Integration tests
    - build “Ground Truth Data”

    * Steps
    * Coverage

3.4 Algorithm Review

3.4.1 Algorithm Review with Kailin Chu

3.4.2 Algorithm Review with Dr. Dean Inglis

- Python Spyder IDE - Data Visualization and Debugging tool
-
-

Chapter 4

Conclusion and Future Works

4.1 Thesis Summary

4.2 Future Works

(Kurugol et al., 2012) discussed about a segmentation workflow that required less hyperparameters.

Appendix A

Your Appendix

Your appendix goes here.

Appendix B

Long Tables

This appendix demonstrates the use of a long table that spans multiple pages.

Col A	**Col B**	**Col C**	**Col D**
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D

Continued on the next page

Continued from previous page

Col A	**Col B**	**Col C**	**Col D**
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D

Bibliography

Sila Kurugol, Raul San Jose Estepar, James Ross, and George R. Washko. 2012. Aorta segmentation with a 3D level set approach and quantification of aortic calcifications in non-contrast chest CT. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2343–2346. https://doi.org/10.1109/EMBC.2012.6346433

Jingyi Lin. 2023. System Requirements Specification. https://github.com/smiths/aorta/blob/main/docs/SRS/SRS.pdf.