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:	xvi, 54

Abstract

Assurance cases have 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

Aorta	The aorta is the largest artery of the body and carries blood from the heart to the circulatory system. It has several sections: The Aortic Root is the transition point where blood first exits the heart. It functions as the water main of the body. The Aortic arch, is the curved segment that gives the aorta its cane-like shape. It bridges the ascending and descending aorta. Throughout the documentation, Aorta would only include Ascending aorta, Aortic arch and Descending aorta. Abdominal aorta is not considered as the interested part.

Ascending Aorta

The ascending aorta is the first part of the aorta, which is the largest blood vessel in the body. It comes out of your heart and pumps blood through the aortic arch and into the descending aorta.

Descending Aorta

The descending aorta is the longest part of your aorta (the largest artery in your body). It begins after your left subclavian artery branches from your aortic arch, and it extends down into your belly. The descending aorta runs from your chest (thoracic aorta) to your abdominal area (abdominal aorta).

Organ Segmentation

The definition of the organ boundary or organ segmentation is helpful for the orientation and identification of the regions of interest inside the organ during the diagnostic or treatment procedure. Further, it allows the volume estimation of the organ, such as the aorta.

Inferior Inferior is the direction away from the head; the lower (e.g., the foot is part of the inferior extremity).

Superior Superior is the direction toward the head end of the body; the upper (e.g., the hand is part of the superior extremity).

Slice A 2-dimensional image is retrieved from a 3-dimensional volume.

Kernel Size The size of the kernel for binary dilation.

Label Map A labeled map or a label image is an image that labels each pixel of a source image.

Binary Dilation

Binary dilation is a mathematical morphology operation that uses a structuring element (kernel) for expanding the shapes in an image.

rms_error Value of RMS change below which the filter should stop. This is a convergence criterion.

Maximum iteration

Number of iterations to run

Curvature scaling

Weight of the curvature contribution to the speed term.

Propagation scaling

Weight of the propagation contribution to the speed term.

Segmented slice

A 2-dimensional image retrieved by applying SITK's ThresholdSegmentationLevelSetImageFilter with the euclidean distance transform image, the original image, and the threshold value calculated with the mean and the standard deviation of the intensity values that were labeled as the white pixel.

Contour Line A contour line (also isoline, isopleth, or isarithm) of a function of two variables is a curve along which the function has a constant value so that the curve joins points of equal value.

Level Sets 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 below demonstrate an example of how 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 outside of the region until the maximum difference between the neighboring pixels are reached.

Threshold Coefficient

This coefficient is used to compute the lower and upper threshold passing through the segmentation filter SITK's ThresholdSegmentationLevelSetImageFilter. The algorithm first uses SITK's LabelStatisticsImageFilter to get the mean and the standard deviation of the intensity values of the pixels that are labeled as the white pixel. Larger values with this coefficient imply a larger range of thresholds when performing the segmentation, which leads to a larger segmented region.

Stop Limit This limit is used to stop the segmentation algorithm. It is used differently in segmentation in inferior direction and segmentation in superior direction.

Euclidean distance transform

The euclidean distance transform is the map labeling each pixel of

the image with the distance to the nearest obstacle pixel (black pixel for this project).

DICOM Digital Imaging and Communications in Medicine (DICOM) is the standard for the communication and management of medical imaging information and related data.

Abbreviations

AGR AortaGeomRecon

AortaGeomRecon

3D Slicer's extension module, Aorta Geometry Reconstruction

DICOM Design Document

DICOM Digital Imaging and Communications in Medicine

MG Module Guide

SITK SimpleITK

SRS Software Requirements Specification

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

The trustworthiness and assurance that a system will perform as anticipated needs heavy testing. When a software carries the critical responsibility of examining the human body, administering medication to patients, saving millions of lifes, and conversely, the tiniest bug or error in the implementation process could have derive into seriour consequences for individuals' well-being. This happens when there are problems with how the system works, whether we expected them or not. It's also influenced by things like the environment it's in, the risks it might face, and people who might want to cause harm. To feel sure about the system, we need to pay attention to its main qualities and collect good evidence that it's doing what we want it to do.

Medical software is tricky to make sure it's reliable. The way it's built is very sensitive, so it might not be possible for the users to check how it works. But without the implementation details, it's hard to trust it, especially for something as important as medical software.

Under this context, we want to explore the feasibility of applying assurance cases

over a medical software from the beginning of the development. With selected arguments and evidences, we want to show to the domain experts that the software delivers correct outputs when used for its intented use/purpose in its intended environment, and within its assumed operating assumptions.

1.1 Objective

In this study, we present the results of applying assurance cases during the development of medical software to enhance stakeholders' confidence in this software. The software, AortaGeomRecon, is a 3D Slicer extension module that aims to semi-automatically construct a 3D model of the Aorta from a patient's chest CT scans. Assurance cases serve as a method for providing assurance for a system by presenting arguments to justify claims about the system, based on evidence regarding its design, development, and tested behavior.

This case study first introduces the challenge of Organ/Aorta Segmentation and explores existing solutions, which might require time and effort from domain experts. Then, we outline the workflow and logic of our algorithm, as well as the working environment for using this module within 3D Slicer. Finally, we discuss our assurance cases, including selected arguments and evidence. This discussion aims to explain how these selected elements enabled us to cultivate confidence in the reliability of the medical 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.

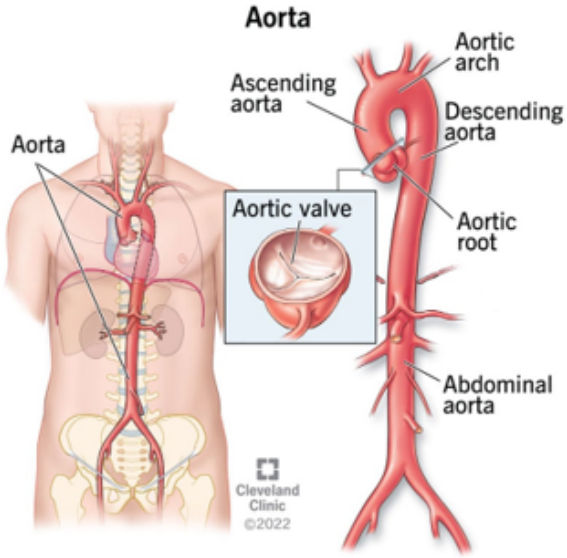


Figure 1.1: Aorta

Aorta segmentation in CT scans is important for:

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

1.3 Thesis Outline

The thesis is organized into three broad parts. In chapter 2, we introduce our program AortaGeomRecon by mentioning the existing methods, the AortaGeomRecon's algorithm overview and step by step workflow. We explain necessary terms and information to understand how the software functionsfinally, and the 3D Slicer extension module that the user interacts with to get the segmentation result with our algorithm. In chapter 3, we present our assurance case, some sections of our SRS, Design Documents, Module Guide, Algorithm Review, and a test case we developed for verifying and validating the correctness of program AortaGeomRecon. Finally, future work is proposed and conclusions are drawn based on the developed assurance case, SRS, segmentation algorithm and 3D Slicer module extension.

Chapter 2

AortaGeomRecon Research and Development

This chapter will discuss 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 (4) get the Aorta 3D geometry with only a few hyper-parameters which can be set within half a minute, and the result requires maximum 2 minutes of execution time.

2.1 Existing Methods

There are much segmentation software available to the users, we will discuss the two built-in methods in two software programs, ITK-nap and 3D Slicer.

2.1.1 ITK-Snap bubble method

ITK-Snap provides a segmentation method that first let user select multiple voxels with a custom initial size and expanding size within the 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.

The advantages of the bubble method is that it guarantee 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. Easier to say then do, an operator 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.

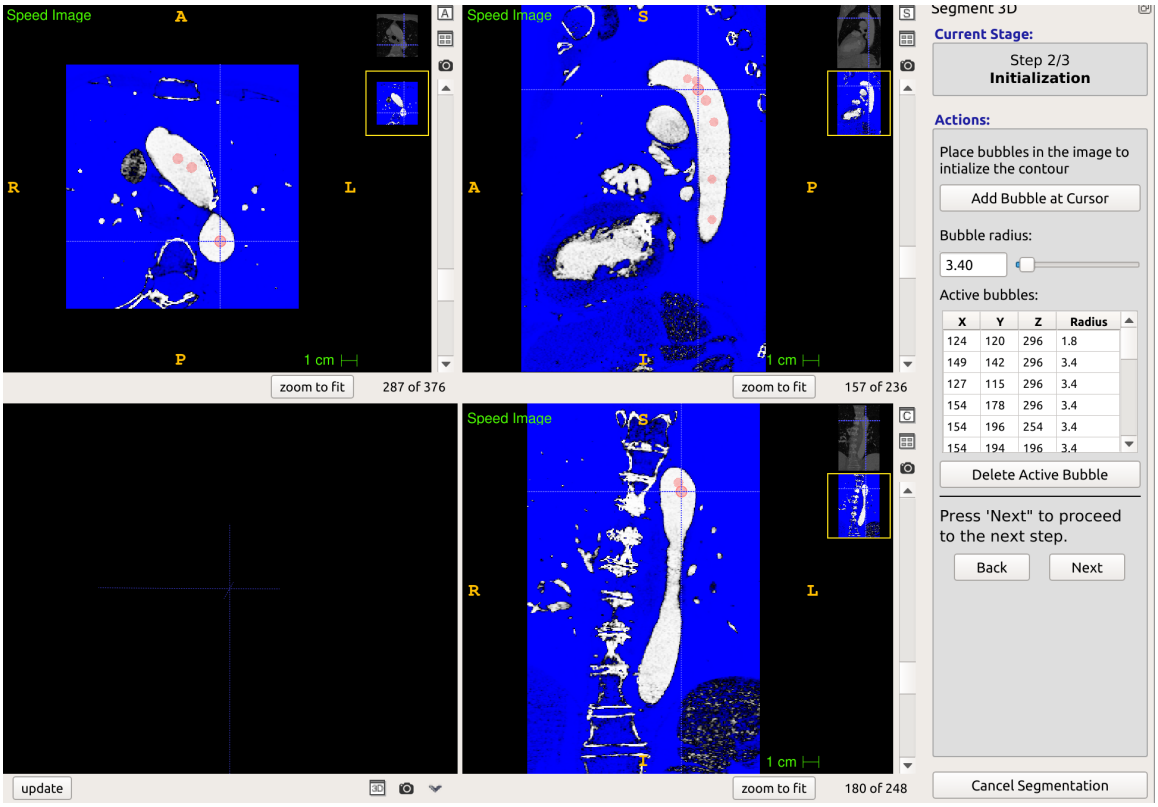


Figure 2.1: ITK-Snap's Bubble segmentation method

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 quickest 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 pixel'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.

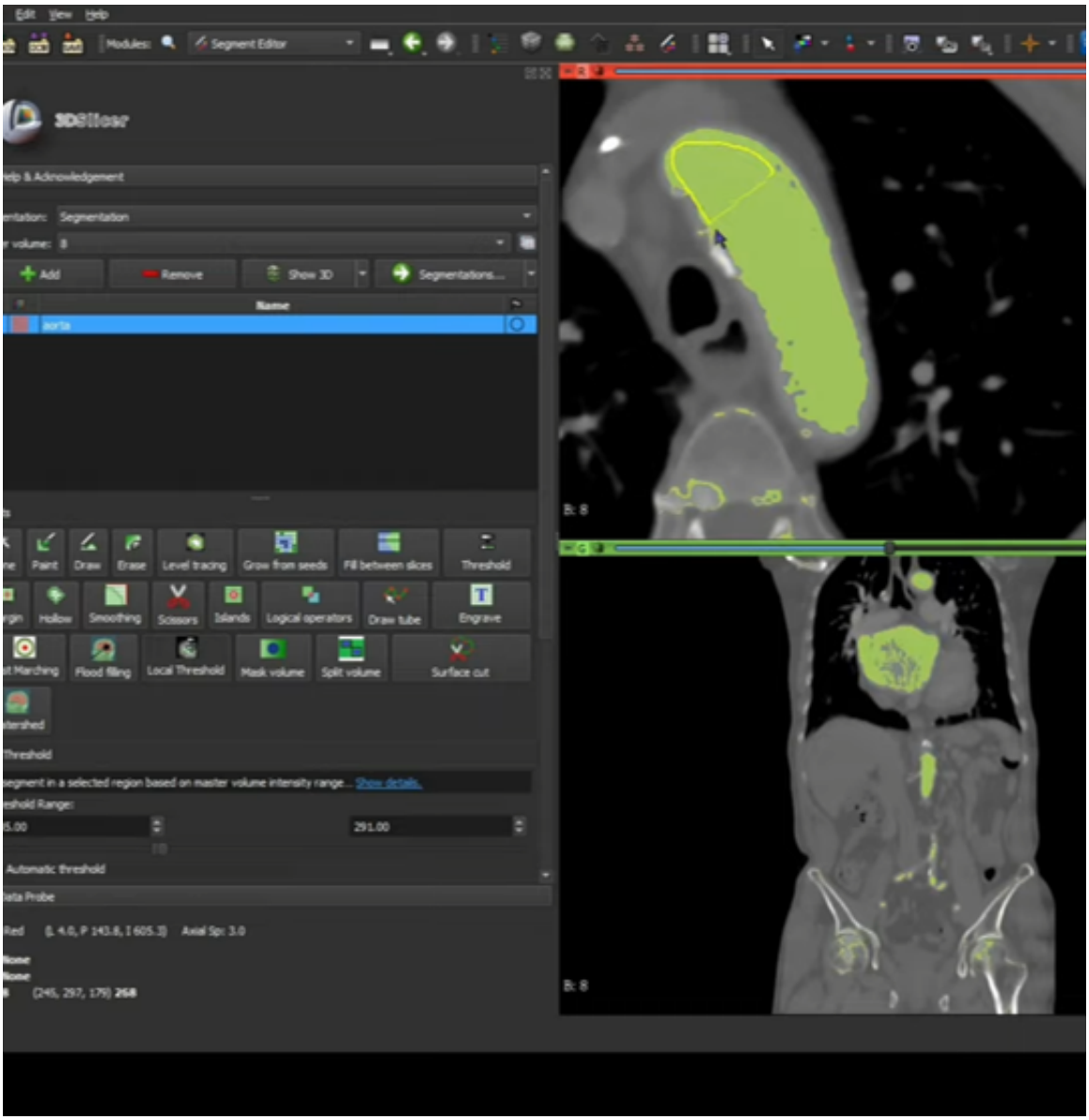


Figure 2.2: 3D Slicer Built-in Segmentation Method

2.2 Segmentation Algorithm

This section introduces the key concepts of the implementation on the segmentation algorithm. The algorithm is developed in Python 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 coordination, the algorithm continuously segments each axial's slice circle closest to the previous Aorta center coordinate. Finally, some hyperparameters tuning can let the algorithm pick up the pixels that were missing but belongs to the part of the Aorta.

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. NumPy is the fundamental package for scientific computing with Python, especially for the performance on multidimensional array processing. 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.

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.

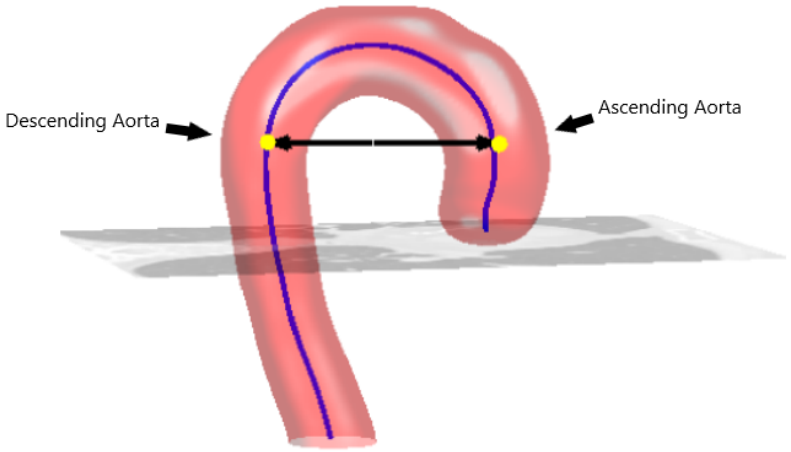


Figure 2.3: The aorta seeds (2)

On the other hand, this 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:

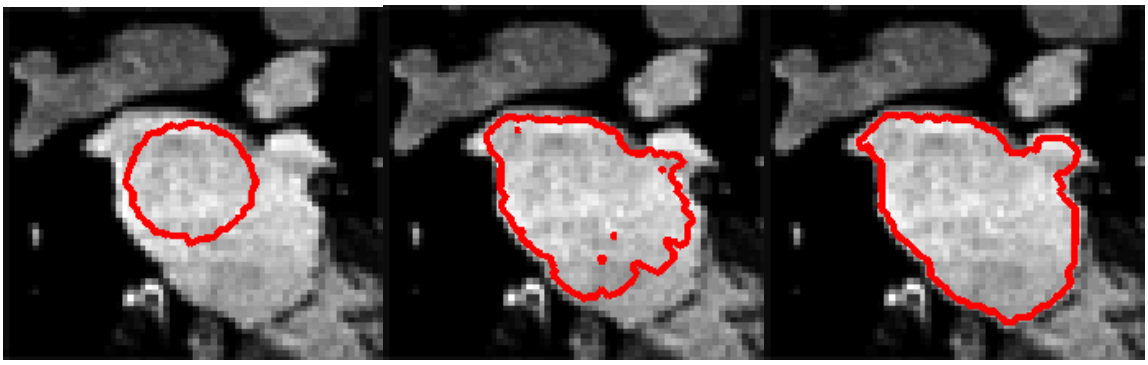
- The RMS error
- The maximum iteration
- The curvature scaling
- The propagation scaling

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 would decrease 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 located is the initial plane. From this plane towards the bottom (the feet) is inferior direction. On the other hand, it is 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 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 outside 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 applied in both segmentation in superior and inferior direction, there is a difference in the stop condition, which we will elaborate in the following section.

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.

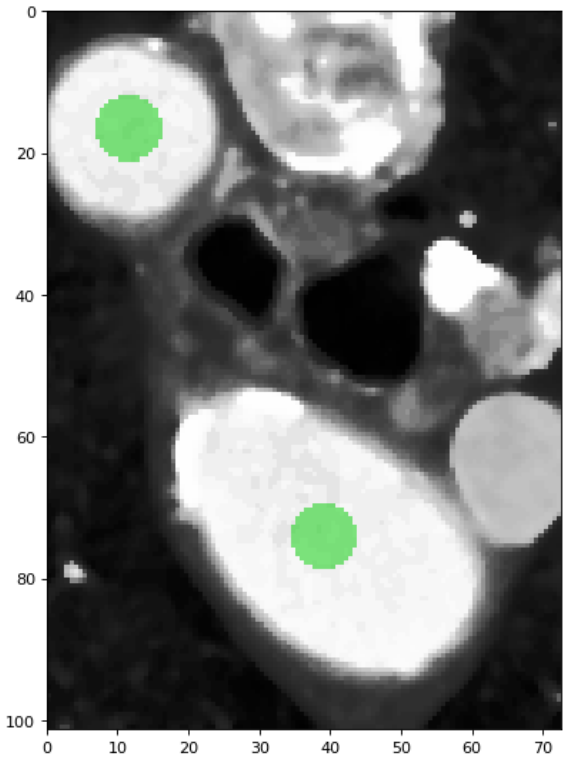


Figure 2.5: A label map

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.

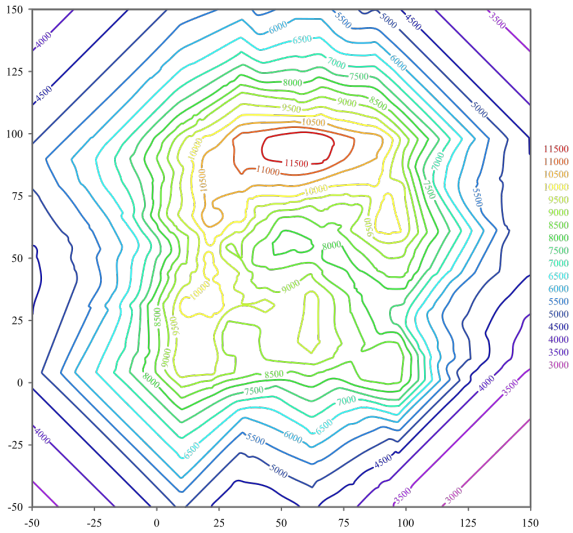


Figure 2.6: A distance map

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 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 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 2.8 shows an example of generated segmented slice (the green parts) overlay over the original slice.

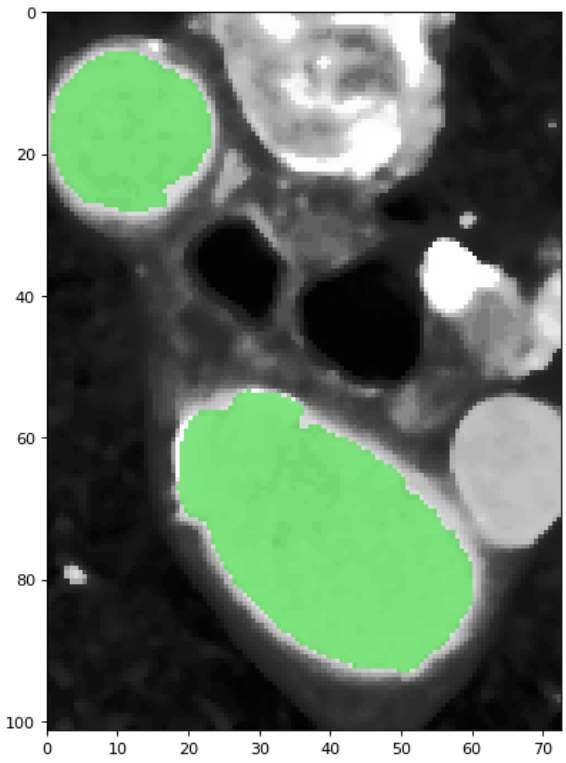


Figure 2.8: A segmented image on top of the original slice

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 new descending aorta centroid, and vice-versa for the ascending aorta centroid. However, at certain point during the segmentation in 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 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 a user defined parameter to control the algorithm, to calculate the condition with the centroids position and the standard deviation.

In 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 consider taking the new centroid closer to the ascending aorta. In other words, only 1 centroid will be used for descending aorta segmentation.

In superior direction, if the standard deviation of the initial label map and the segmentation result label map has larger difference 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 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 (the feet) plane, the algorithm calculate a label map with two centroid coordinates and kernel size, calculate a distance map with the label map, calculate a threshold range with the label map's selected pixels, perform segmentation, calculate new centroid coordinates, and verify the segmentation result in case that it reaches the stop condition. Once the algorithm finished the segmentation in inferior direction, the algorithm works from the initial plane to the top (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.

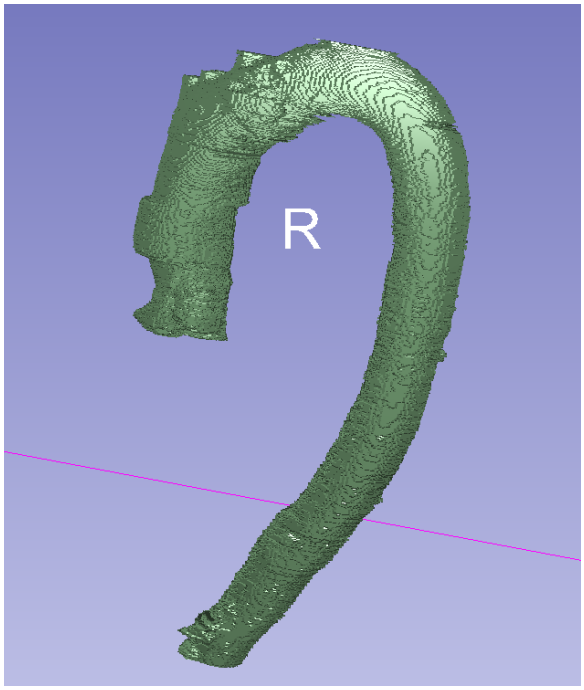


Figure 2.9: Segmentation Result

2.3 3D Slicer Extension Development

The project has started with a simple segmentation algorithm 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 such as coordinates and size to crop (the coordinates of the yellow dots shown in Figure 2.3).

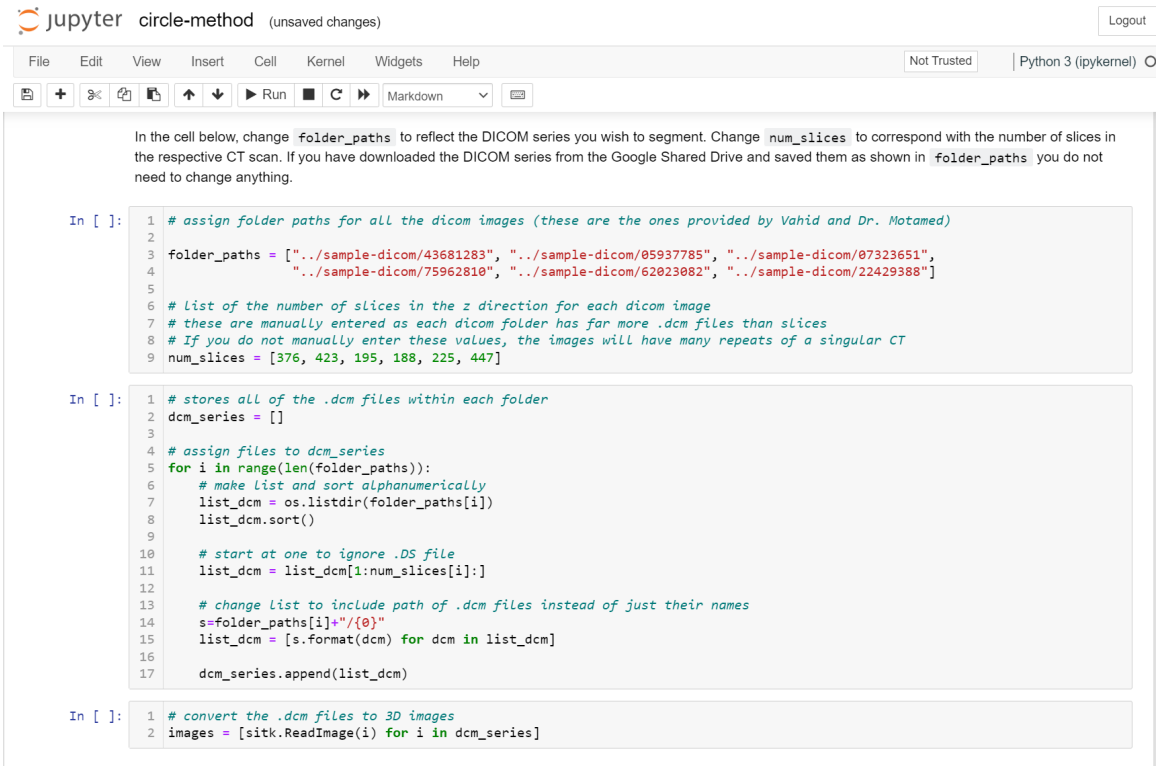


Figure 2.10: Jupyter Notebook Researched by Kailin Chu

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 Interface.

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

2.3.3.1 Graphical User Interface

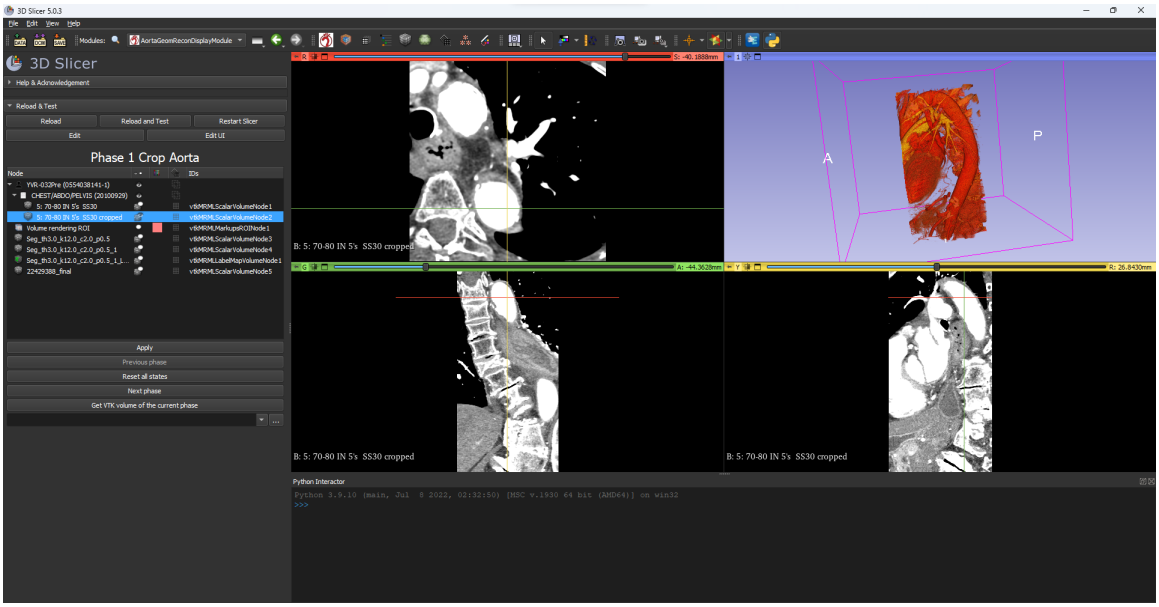


Figure 2.11: 3D Slicer UI

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 I 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.

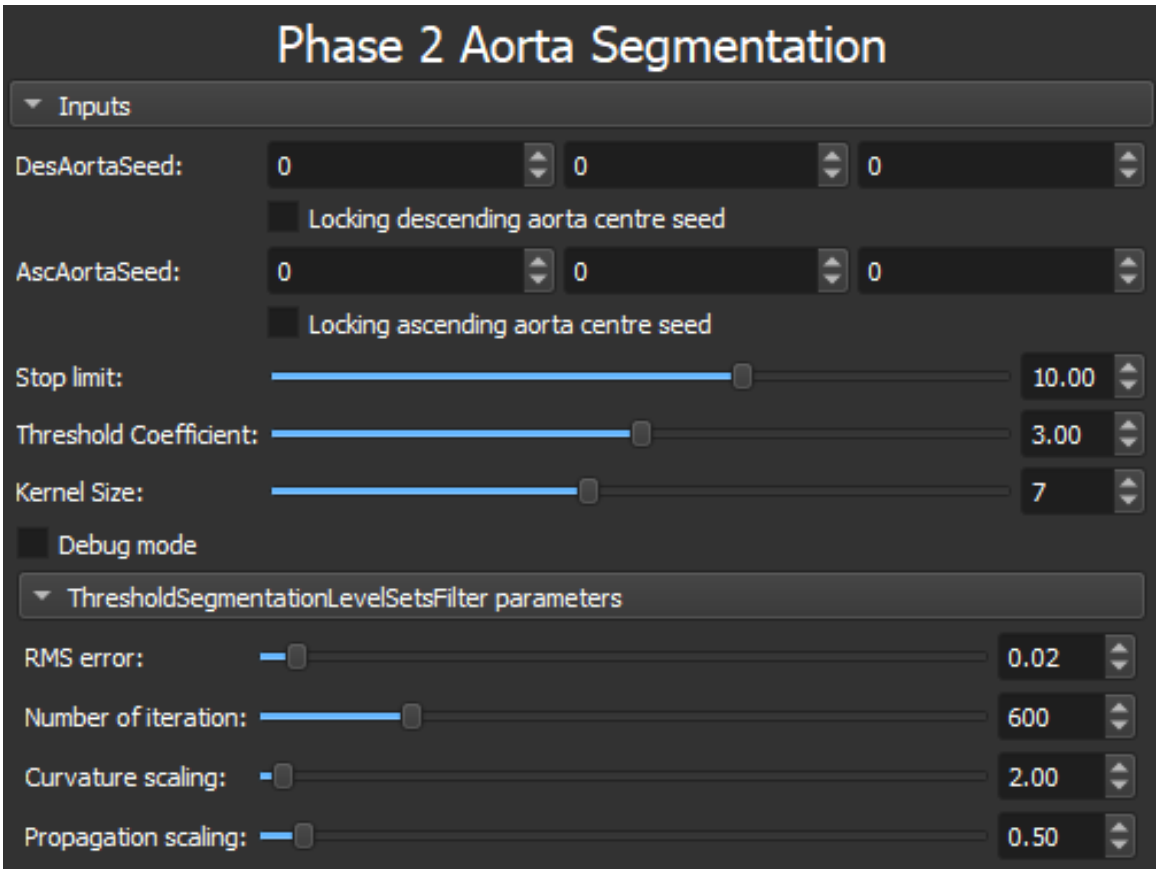


Figure 2.12: AortaGeomRecon Module UI

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.

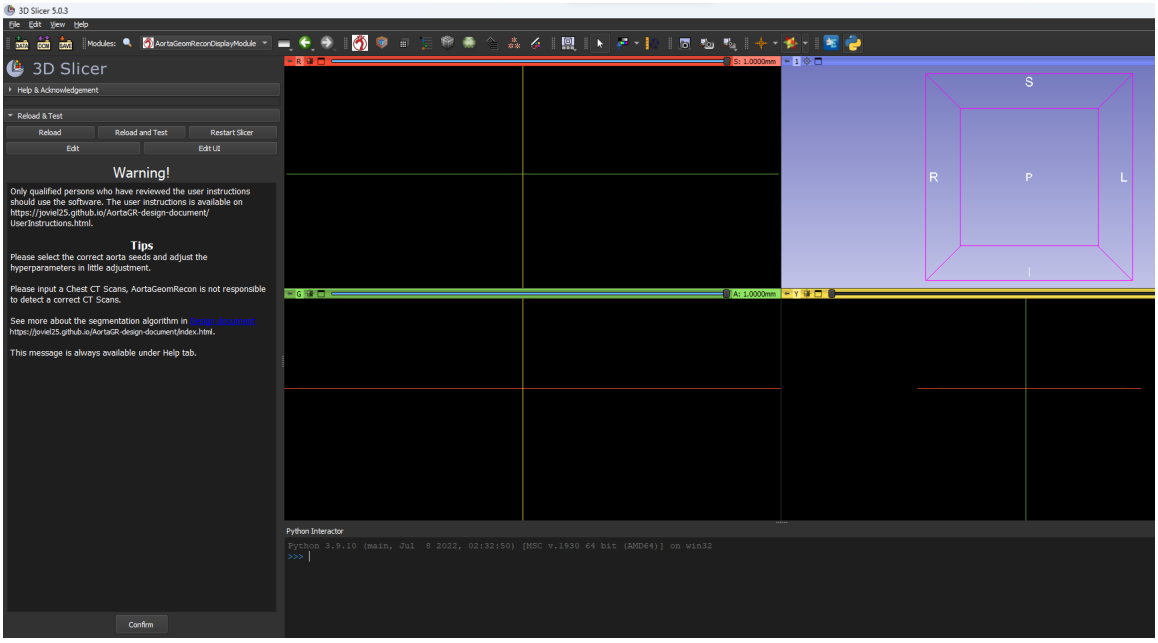


Figure 2.13: AortaGeomRecon Warning message

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 user to generate segmentation result.

Chapter 3

Assurance Cases and Selected Evidence for AortaGeomRecon

3.1 Assurance Case Development

Assurance Case shows the statements and the evidences to ensure the goals are fulfilled. By using Astah System Safety presenting in Goal Structuring Notation (GSN) arguments, we want to show that our software delivers correct outputs when used for its intended use/purpose in its intended environment, and within its assumed operating assumptions. The Figure 3.1 shows the top level of the assurance cases. Having the goal of delivering correct outputs, we are decomposing the goal into 4 sub goals, GR, GI, GA and GBA.

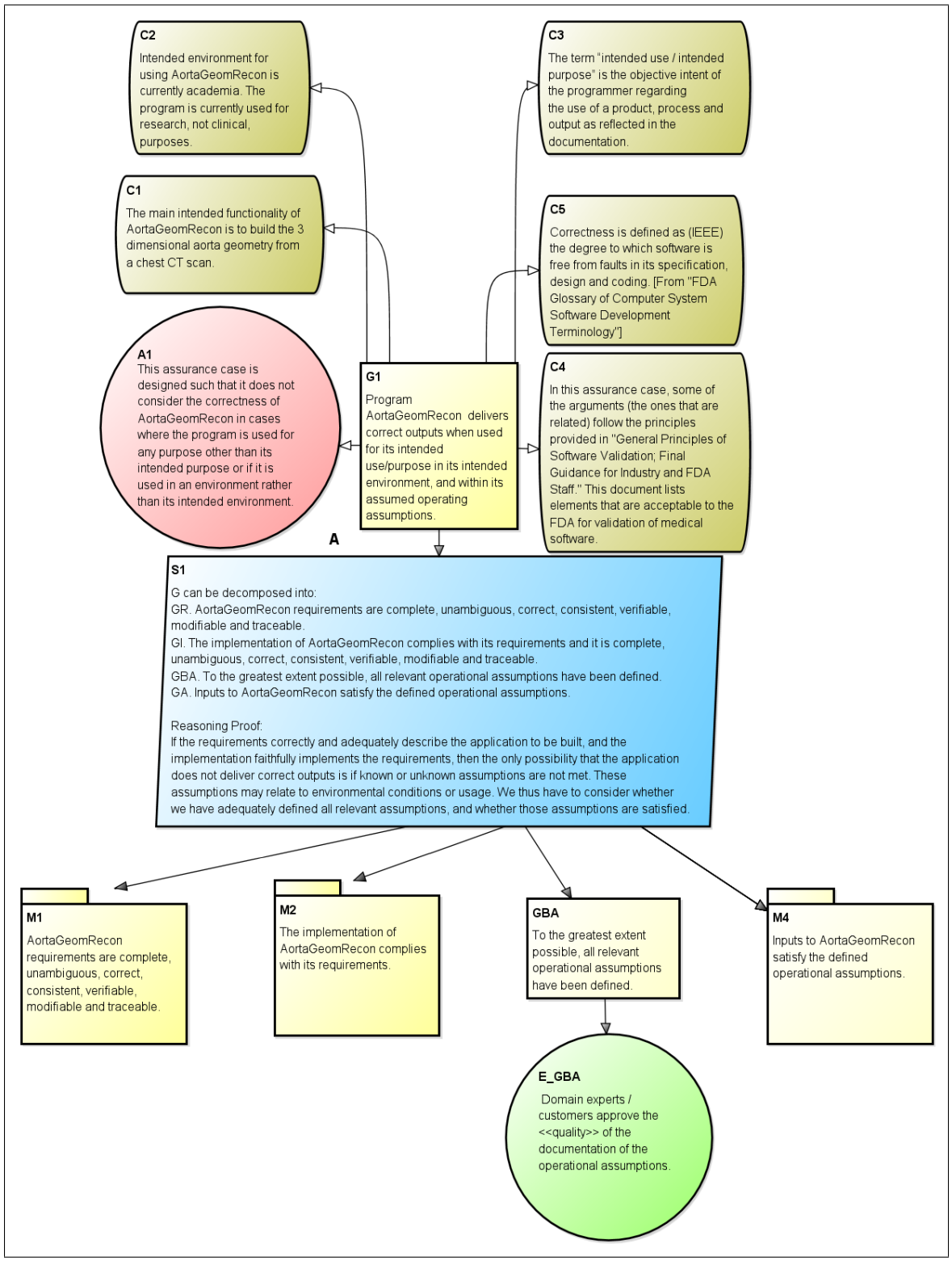


Figure 3.1: AortaGeomRecon Assurance Cases Top Level

3.2 Assurance Case for Software Specification Requirements

The first goal of getting a trusted software is having a complete, unambiguous, correct, consistent, verifiable, modifiable and traceable Software Specification Requirements (4) which shows the complete breakdown of the requirements with mathematical notation, data models and instance models. It's the foundation of the software development, and the design and the implementation will be based on the requirement document.

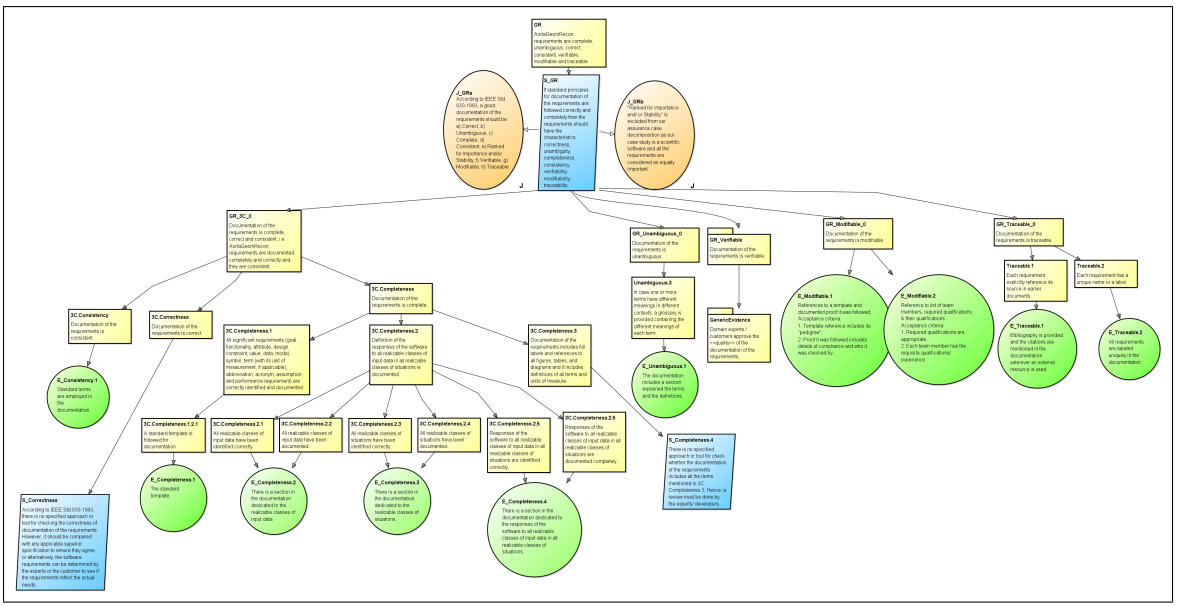


Figure 3.2: AortaGeomRecon Assurance Cases GR

As explains in the assurance case GR, one of most important statement of SRS having these characteristics is using a standard template. This template is provided by the domain expert, and my professor, Dr. Spencer Smith, who has researched on Application of Software Engineering Principles, Methods, Techniques and Tools to

Improve the Sustainability of Research Software.

The chapters in SRS and some of the most important sections in the chapter are explained below:

- Reference Material

  In this section, a Table of Symbols and an Abbreviations and Acronyms table are used to explain every symbol and Abbreviations used in the SRS document. These tables ensure the consistency and the unambiguous characteristics of the document. They are located at the very beginning of the document, so the reader will firstly look at these tables before reading the entire document.

- Introduction

  In the introduction section, we introduced the problems and the scope of the document to the user by explaining the purpose of document, abstracting the scope of requirements and defining the characteristics of intended reader.

- General System Description

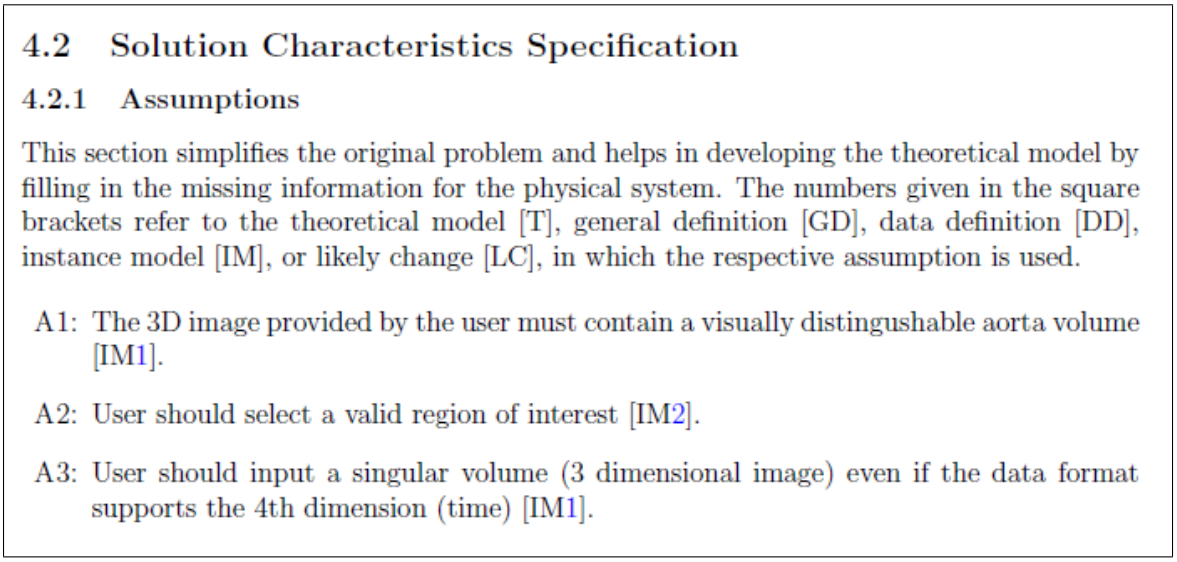
  The general system description includes a system context diagram which explains the relationship between the users, the inputs given by the user and the outputs of the AortaGeomRecon program. User responsibility and AortaGeomRecon responsibility are defined such that the reader knows what to do to successfully generate the desired result.

- Specific System Description

  In this section, we are presenting more details about the problem and the specific system to solve the problem. The first subsection Problem Description discussed

on the definition of Organ Segmentation, Coordinate Systems used in Medical Image, Physical System Description which is not available, and Goal Statements which is extract the three-dimensional segmentation of the aorta.

In the next subsection, Solution Characteristics Specification, we started with Assumptions to simplify the problem and helps in developing the theoretical model by filling in the missing information for the physical systems. In the subsection Data Definitions, we defined Voxel, Image/Slice, and Volume with mathematical notation so that the developer can easily interpret. Next, in the subsection Instance Model, we showed the mathematical meaning of Region of Interest, and Segmentation, which are the two essential terms that the developer must know in order to develop the solution.



4.2 Solution Characteristics Specification

4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [T], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: The 3D image provided by the user must contain a visually distingushable aorta volume [IM1].

A2: User should select a valid region of interest [IM2].

A3: User should input a singular volume (3 dimensional image) even if the data format supports the 4th dimension (time) [IM1].

Figure 3.3: AortaGeomRecon SRS Assumptions

- Requirements

  With all the information in the document, we can now present the Functional

Requirements and the Non-Functional Requirements for the program AortaGeomRecon. The Functional Requirements are defined By using the terms we presented in Data Definitions, Instance Model, and based on the other Functional Requirements. The Non-Functional Requirements usually have a measurement such as execution time, the effort of manual works, etc.

5.1 Functional Requirements

R1: Input the following functions, data and parameters:

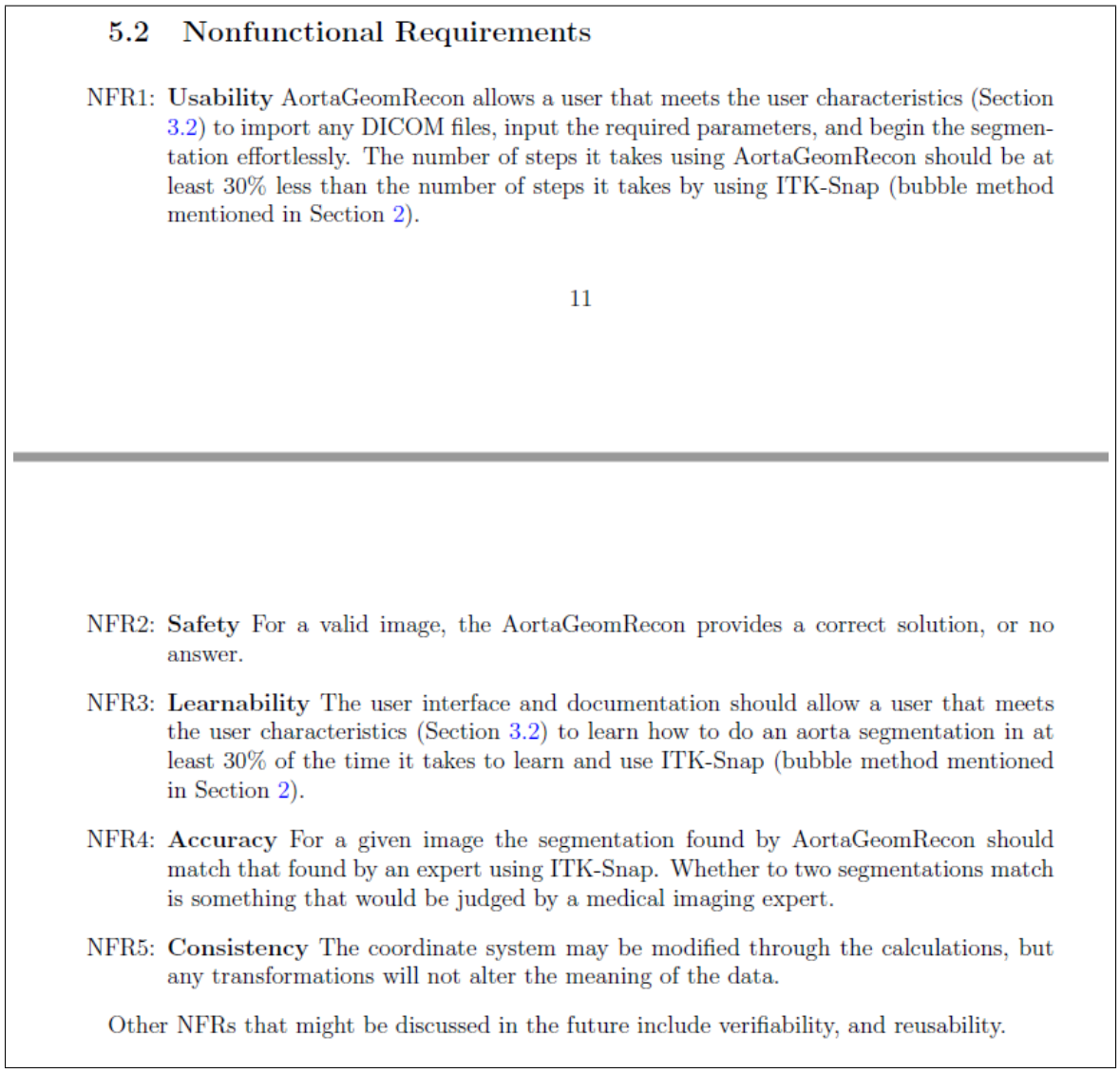
symbol	description
V	CT Scans volume (DD3)
$Seed\_a$	The seed of ascending aorta centre coordinate (IM2)
$Seed\_d$	The seed of descending aorta centre coordinate (IM2)

R2: Use the volume in R1 to create a second volume, the region of interest (IM1) that contains all voxels of the aorta.

R3: Perform segmentation (IM2) on the volume created in R2.

R4: Visualize a volume (DD3).

Figure 3.4: AortaGeomRecon Functional Requirements



5.2 Nonfunctional Requirements

NFR1: **Usability** AortaGeomRecon allows a user that meets the user characteristics (Section 3.2) to import any DICOM files, input the required parameters, and begin the segmentation effortlessly. The number of steps it takes using AortaGeomRecon should be at least 30% less than the number of steps it takes by using ITK-Snap (bubble method mentioned in Section 2).

11

NFR2: **Safety** For a valid image, the AortaGeomRecon provides a correct solution, or no answer.

NFR3: **Learnability** The user interface and documentation should allow a user that meets the user characteristics (Section 3.2) to learn how to do an aorta segmentation in at least 30% of the time it takes to learn and use ITK-Snap (bubble method mentioned in Section 2).

NFR4: **Accuracy** For a given image the segmentation found by AortaGeomRecon should match that found by an expert using ITK-Snap. Whether to two segmentations match is something that would be judged by a medical imaging expert.

NFR5: **Consistency** The coordinate system may be modified through the calculations, but any transformations will not alter the meaning of the data.

Other NFRs that might be discussed in the future include verifiability, and reusability.

Figure 3.5: AortaGeomRecon Non- Functional Requirements

- Likely Changes and Unlikely Changes

  This section discussed the likely changes that the developer might expect a change in the future works, and the unlikely changes that is most certainly not going to change for a justified reason. The only likely change discussed in the AortaGeomRecon's SRS is regarding the segmentation method. For different

segmentation method, the inputs varies, since the segmentation method is a likely change, the inputs variables are also likely changes. The only unlikely change is the method of retrieve a region of interest. Most methods take a starting point and sizes in different dimensions to get the region of interest.

- Traceability Matrix and Graphs

  The traceability matrices are to provide easy references on what has to be additionally modified if a certain component is changed. Below shows the traceability matrices of different sections:

	DD1	DD2	DD3	IM1	IM2
DD1					
DD2	X				
DD3		X			
IM1			X		
IM2				X	

Table 2: Traceability Matrix Showing the Connections Between Items of Different Sections

Figure 3.6: AortaGeomRecon Traceability Matrix between Data Definitions and Instance Model

	IM1	IM2	R1	R2	R3	R4	NFR1	NFR2	NFR3	NFR4	NFR5
IM1				X							
IM2					X						
R1		X									
R2	X										
R3		X									
R4							X				
NFR1			X	X	X	X			X		
NFR2		X									
NFR3				X	X	X	X				
NFR4		X									
NFR5		X									

Table 3: Traceability Matrix Showing the Connections Between Requirements and Instance Models

Figure 3.7: AortaGeomRecon Traceability Matrix between Requirements and Other sections

	A1	A2	A3
DD1			
DD2			
DD3			X
IM1	X		X
IM2		X	X
LC1	X	X	X
UC1			X

Table 4: Traceability Matrix Showing the Connections Between Assumptions and Other Items

Figure 3.8: AortaGeomRecon Traceability Matrix between Assumptions and Other sections

3.3 Assurance Case for Implementation

The goal of implementation asked the developer to fully comply the design and the implementation with the Software Requirements Specification document. Since we have showed with Assurance Case that our SRS is complete, consistent, and unambiguous, the design and the implementation that fully complying with requirements is correct.

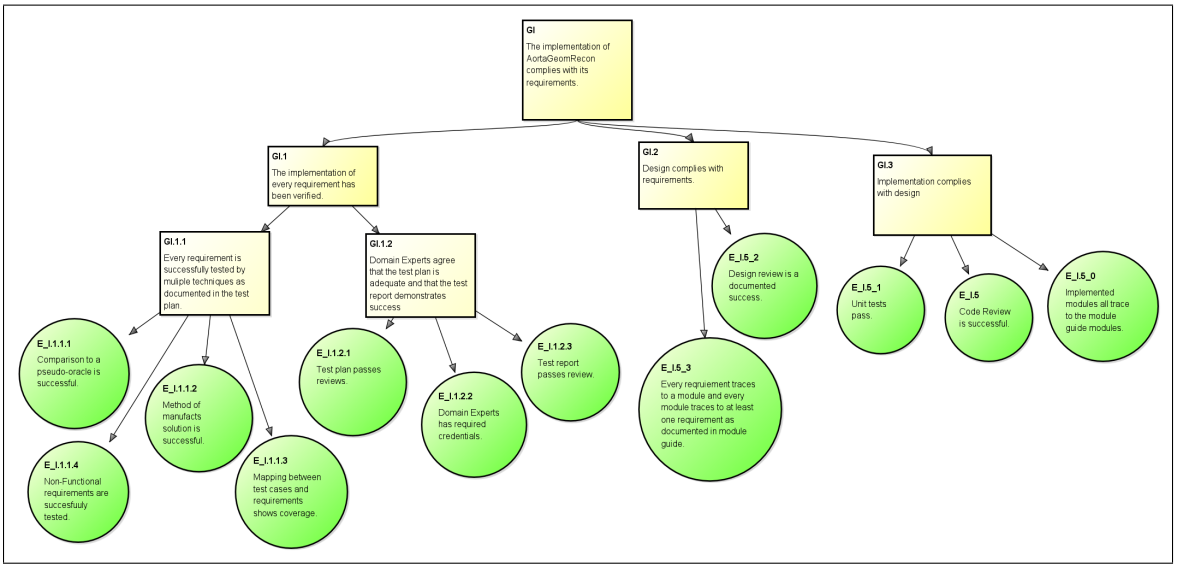


Figure 3.9: AortaGeomRecon Assurance Cases for Implementation

3.3.1 Design Document

The purpose of the Design Document (3) is to explain in details how the algorithm works, and why it worked. Similar to what section 2.2.3 wrote, the design document explains in plan text the workflow of the algorithm. Since before the domain expert can agree on the design and implementation, the person needs to understand the algorithm, we are building a Design Document that is easy to access.

3.3.1.1 Sphinx - Python Documentation Generator

To implement this Design Document, I used Sphinx, a Python Documentation Generator that can build module's documentation with the comments in the source code. Moreover, using reStructuredText to write the Algorithm Overview, we can build HTML binary which can be published on a web server, which I was able to implement and accomplish. Another important section in Design Document is the Glossary. It has rich vocabulary explanation, images, and links to the outside source to let the reader understands as much as possible.

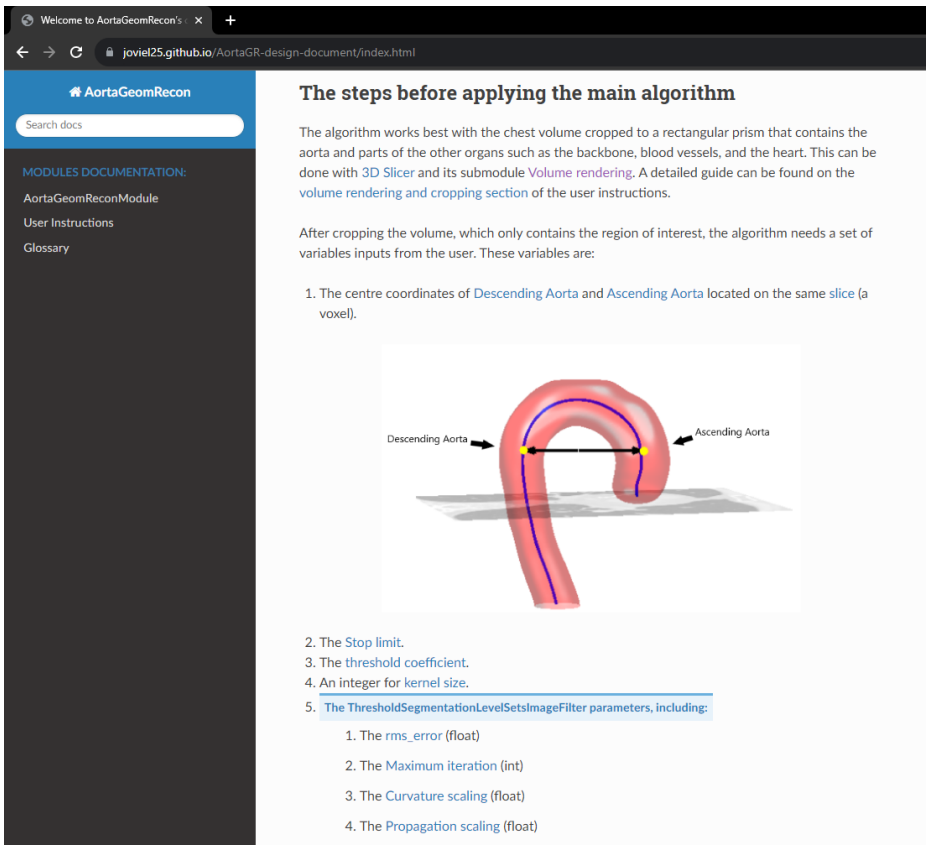


Figure 3.10: AortaGeomRecon Design Document website

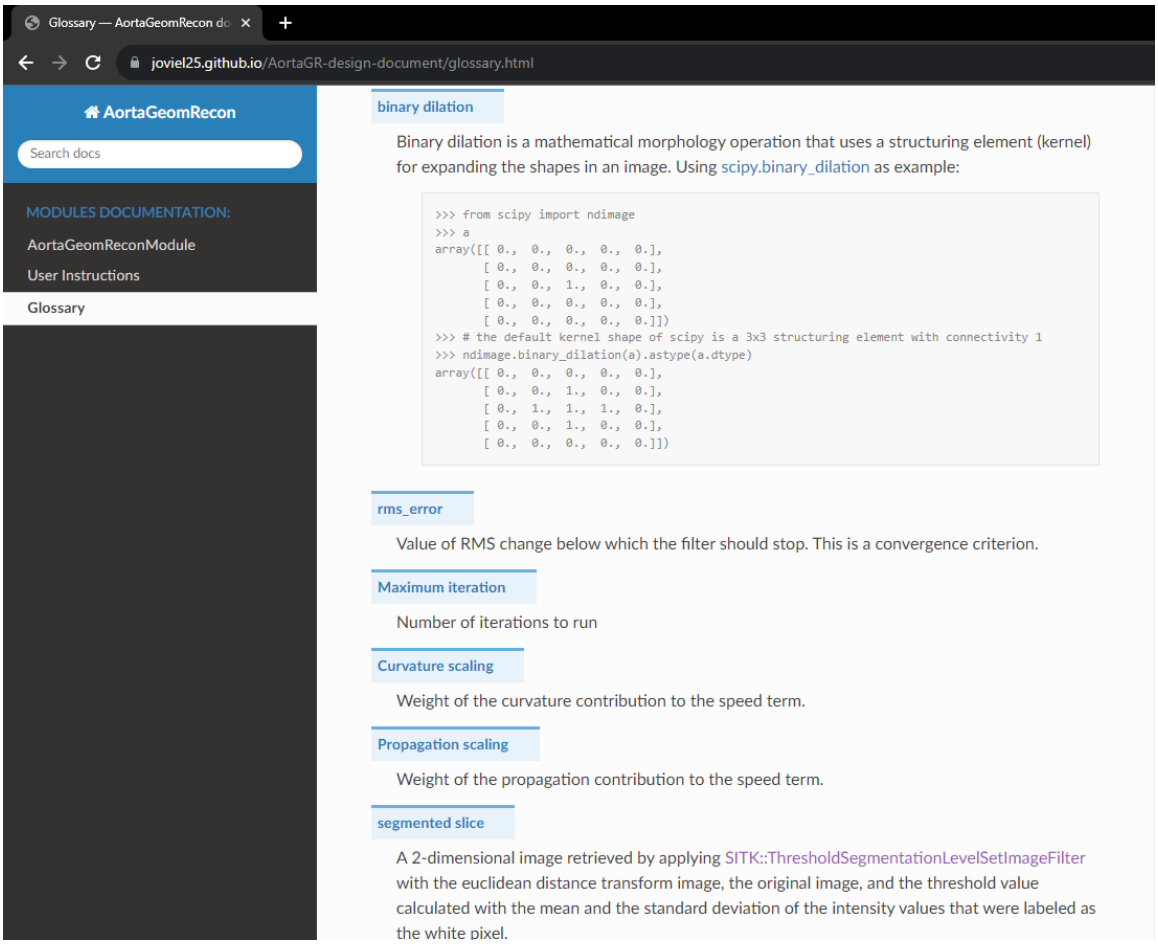


Figure 3.11: AortaGeomRecon Design Document Glossary

3.3.2 Module Guide

In the design of the software, we first list the anticipated changes such that since we are expecting a change to this piece of requirement of information, we will keep it a single module so when the changes did happen, we only need to concern about the module and its dependency modules. The anticipated changes are listed in the Figure 3.12 below.

AC1: The specific hardware on which the software is running.

AC2: The format of the initial input data.

AC3: The algorithm to segment the aorta.

AC4: The data structures to store the input parameters required to execute the algorithm.

AC5: The methods to create a user interface.

AC6: The methods to retrieve a region of interest.

AC7: The methods to visualize a volume.

AC8: How the overall control of the calculations is orchestrated.

AC9: The format of the final output data.

Figure 3.12: AortaGeomRecon Anticipated Changes

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The Secrets field in a module decomposition is a brief statement of the design decision hidden by the module. The Services field specifies what the module will do without documenting how to do it. For each module, a suggestion for the implementing software is given under the Implemented By title. If the entry is OS, this means that the module is provided by the operating system or by standard programming language libraries. AortaGeomRecon means the module will be implemented by the AortaGeomRecon software.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Input Format Module

M3: Input Parameter Module

M4: Control Module

M5: GUI Module

M6: Volume Visualization Module

M7: Crop Volume Module

M8: Aorta Segmentation Module

M9: Image Processing Module

M10: Multidimensional Array Processing Module

M11: Digital Enhancement Module

Figure 3.13: AortaGeomRecon Modules

7.2.2 Input Parameter Module (M3)

Secrets: The data structure for input parameters, how the values are input and how the values are verified. The load and verify secrets are isolated to their own access programs.

Services: Gets input from user, stores input and verifies that the input parameters comply with physical and software constraints. Throws an error if a parameter violates a physical constraint. Throws a warning if a parameter violates a software constraint. Stored parameters can be read individually, but write access is only to redefine the entire set of inputs.

Implemented By: AortaGeomRecon

Source: AortaSegmenter class' atributes

7.2.3 Control Module (M4)

Secrets: The algorithm for coordinating the running of the program.

Services: Provides the main program's entry point, the ability to jump from a program state to another.

Implemented By: AortaGeomRecon

Source: AortaGeomReconDisplayModuleWidget module

7.2.4 Volume Visualization Module (M6)

Secrets: The methods which allow users to visualize a 3D Volume.

Services: Display the aorta images and vtk 3D geometry.

Implemented By: 3D Slicer

Figure 3.14: AortaGeomRecon Module Decomposition Example

Now that we have listed the anticipated changes and the modules, we are using traceability matrices to show the relationships between the modules and the anticipated changes, and the modules between the requirements. This indicates that the design is fully complying with the requirements, as we stated in GI.2 in the Figure 3.9.

Req.	Modules
R1	M1, M2, M3, M4
R2	M3, M7
R3	M8, M9, M10
R4	M6
NFR1	M3, M4, M5
NFR2	M4, M8, M9, M10
NFR3	M3, M4, M5 M6, M7, M8
NFR4	M7, M8, M9, M10
NFR5	M3

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M2
AC3	M8
AC4	M3
AC5	M5
AC6	M7
AC7	M6
AC8	M4
AC9	M9, M10

Table 3: Trace Between Anticipated Changes and Modules

Figure 3.15: AortaGeomRecon Modules Traceability Matrices

On top of relating the modules to the requirements, we are relating the actual source code to the modules, which is a strong evidence of our implementation has

fully complying with the requirements. Since our requirements has proven to be correct, complete, and consistent, the implementation must also be trustworthy.

M6 Volume Visualization Module	3D Slicer's Volume Rendering Module 3D Slicer's Volume Rendering Source Code
M7 Crop Volume Module	3D Slicer's Crop Volume Module 3D Slicer's Crop Volume Module Source Code
M8 Aorta Segmentation Module	AortaSegmenter class
M9 Image Processing Module	SimpleITK
M10 Multi-Dimensional Array Processing Module	NumPy
M11 Digital Enhancement Module	

```
# AortaGeomReconDisplayModule.py
# https://github.com/smiths/aorta/blob/main/src/SlicerExtension/
    AortaGeometryReconstructor/AortaGeomReconDisplayModule/
    AortaGeomReconDisplayModule.py#L739-L769
    def transform_image(self, cropped_volume):
        """
        Histogram Equalization for Digital Image Enhancement.
        https://levelup.gitconnected.com/introduction-to-histogram-
            equalization-for-digital-image-enhancement-420696db9e43
        """
        cropped_image = sitkUtils.PullVolumeFromSlicer(cropped_volume)
        img_array = sitk.GetArrayFromImage(
            (sitk.Cast(sitk.RescaleIntensity(cropped_image), sitk.
                sitkUInt8)))
        histogram_array = np.bincount(img_array.flatten(), minlength=256)
        num_pixels = np.sum(histogram_array)
        histogram_array = histogram_array/num_pixels
        chistogram_array = np.cumsum(histogram_array)
        transform_map = np.floor(255 * chistogram_array).astype(np.uint8)
        img_list = list(img_array.flatten())
        eq_img_list = [transform_map[p] for p in img_list]
        eq_img_array = np.reshape(np.asarray(eq_img_list), img_array.
            shape)
        eq_img = sitk.GetImageFromArray(eq_img_array)
        eq_img.CopyInformation(cropped_image)
        median = sitk.MedianImageFilter()
        median_img = sitk.Cast(median.Execute(eq_img), sitk.sitkUInt8)
        self._cropped_image = median_img
```

Table 4: Trace Between Modules and Code

Figure 3.16: AortaGeomRecon Part of the Traceability Matrix on Modules and Code

3.3.3 Test Plan

GI.1.2 states that a test plan must be adequate and test report demonstrates success. Unlike the other algorithm that can easily be tested with a ground truth test case, our ground truth case is build by using another more accurate method such as ITK-Snap's bubble method, then manually erase the unwanted pixels.

3.3.3.1 Build "Ground Truth" data

Instead of using ITK-Snap's bubble method, we build ground truth test case with our existing algorithm and updated algorithm. If there exists a difference between two results, our domain experts can make a visual comparison of both test cases to see which algorithm to use for the future.

3.3.3.2 GitHub Actions workflows

This leads to our Continuous Integration infrastrucuture, implemented with GitHub Actions workflow. A workflow is a configurable automated process that will run one or more jobs on the desired system. GitHub Actions workflow used a YAML file to define the events and the commands to be executed on the temporary system, which has the build of the repository. (1)

We have set up two automated process which happens on each "push" event and "pull" event. A "push" event implies that something is changed in one or multiple commits, therefore there is a need to verify whether the commits has bugs that need extra fixes. A "pull" event happens when a feature branch is going to merge with the main branch. Since our main branch is protected, any update to the main branch must be merged by using a pull-request. Before a pull-request can be approved, the

continuous integration tests are examined and until there are no errors, a pull-request cannot be merged with the main branch.

The first automated process is a linter. A linter is a tool for static code analysis to flag programming errors, bugs, stylistic errors and suspicious constructs. We used Python Flake8 as our linter to find bugs and errors, and ensures that program's readability by stricting the source code with Google's published Python Style Guide. (6)

The second automated process is our continuous integration tests. By setting up Git Large File System (LFS) and upload the pre-build ground truth test data in the repository, we can now pass the cropped volume as the inputs data, the same Aorta seeds and the hyperparameters that we have used to generate the ground truth test data to the algorithm and verify the results. To compare the volumes, we used Dice similarity coefficient.

3.3.4 Algorithm Review

The algorithm review was an idea started with Code Walkthrough. The difference is that Code Walkthrough aims to increase the confidence of the reviewer such that the reviewed implementation is doing what we have discussed. For an Algorithm Review, we have not discussed what the program should do; we are presenting the algorithm to the domain expert and asking them if the design and implementation fulfill the implementation objectives.

3.3.4.1 Tool used in Algorithm Review

Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. The Variable Explorer allows the user to interactively browse the variables and the objects in debugging mode. (5)

Figure 3.17: Spyder Variable Explorer

This feature allows us to execute the program step by step, and see what happens when executing the segmentation algorithm.

3.3.4.2 Algorithm Review with Kailin Chu

The first algorithm review was done with Kailin Chu, who is a biomedical engineer and started the semi-automacial aorta segmentation algorithm. Along with Smith Spencer, we were aiming to increase the domain expert's confidence in this code

walkthrough. This code walkthrough did not increase our confidence in the software, because the code was developed by Kailin from two years ago. Some of the details and design decisions are missing, and some variables were decided by trial and error. Despite that the code walkthrough did not achieve what we wanted in the first place, this meeting was still very helpful. Knowing that the algorithm was partially based on trial and error, I was able to improve the algorithm and reaching a better result in a more efficient way.

3.3.4.3 Algorithm Review with Dr. Dean Inglis

The second algorithm review was done with Dr. Dean Inglis, who is an experienced professor, Medical Image Analyst and Software Developer. We showed him our segmentation algorithm and asked for a confirmation of this method, or a better algorithm. Dr. Dean Inglish has shared his thoughts on the algorithm, and we have noted all of his advices on GitHub issue tracker for the developer who will be in charge of improving this program. This meeting effectively increase our confidence in the software, whether the idea of developing extension module on 3D Slicer or the algorithm, our works have been affirmed.

3.4 Assurance Case for Operational Assumptions

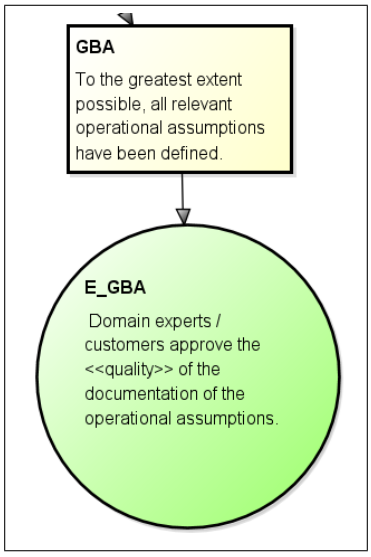


Figure 3.18: AortaGeomRecon Assurance Case Operational Assumptions

The evidence for the statement "To the greatest extent possible, all relevant operational assumptions have been defined" is quite simple as only Domain experts/customers approve the quality of the documentation. This documentation was build from the beginning of the project to the end of the project, where we want to continuously adjust the content of the document matching the most recent updates.

3.4.1 User Manual

An user manual serves the purpose of documenting the all operational assumptions. When the user getting unexpected results by using this software, they can always refer to the user manual to see what pieces are different. Our user manual is initially located in GitHub repo's README, which is only available to the developers invited as the contributors. The content includes the installation of the software, importing

the extension modules, import inputs data, and perform segmentation. The user manual is also available publicly on the design document website, assuming that the users might not be the repository contributors.

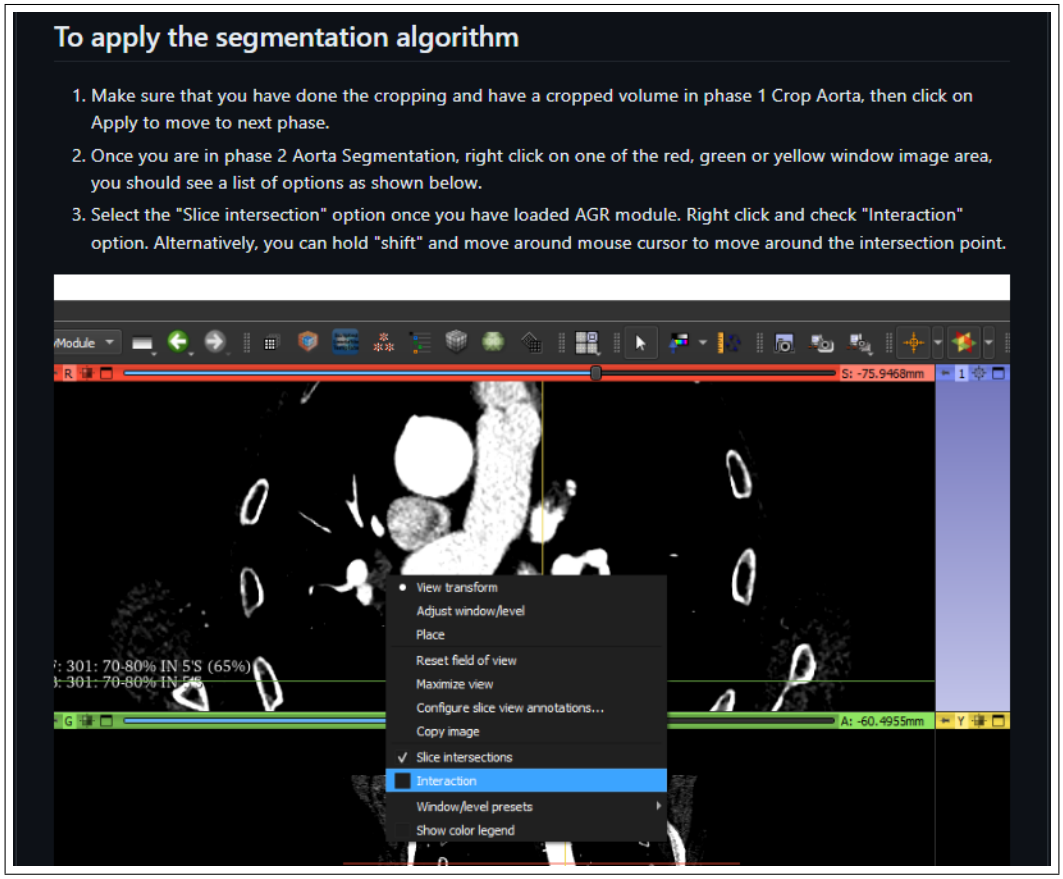


Figure 3.19: AortaGeomRecon User Manual on GitHub README

3.4.2 User Instruction Video

Videos are an effective way to engage your audience and deliver information in a way that's easy to follow along and understand. A better instructional content is a YouTube Video where I make step by step instruction with voice over to instruct user. The video is not listed publicly on YouTube, but the users who have access to

the GitHub repository or Design Document website can access this video by the url link.

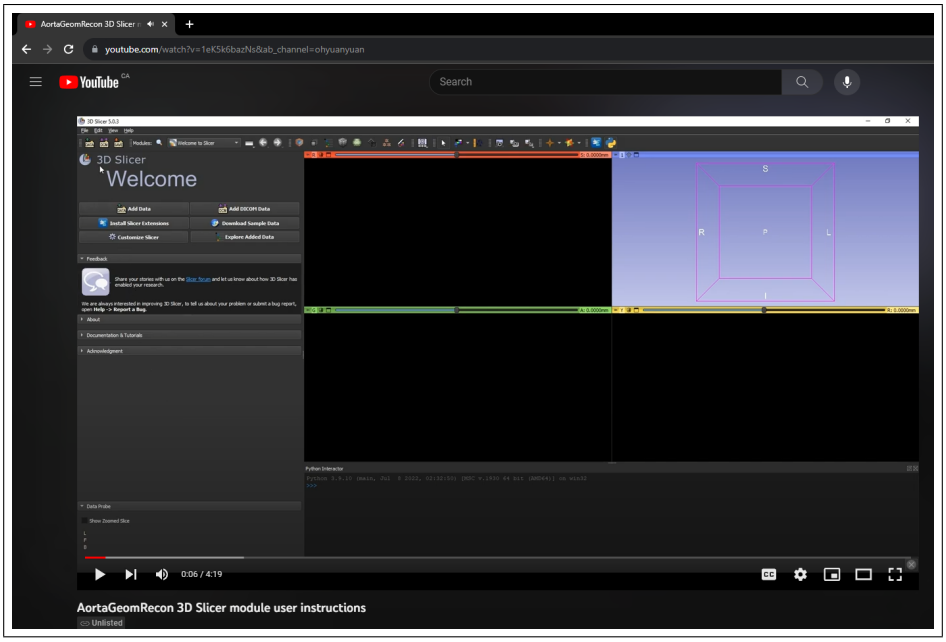


Figure 3.20: AortaGeomRecon User Instructions on YouTube

3.5 Assurance Case for Inputs Assumptions

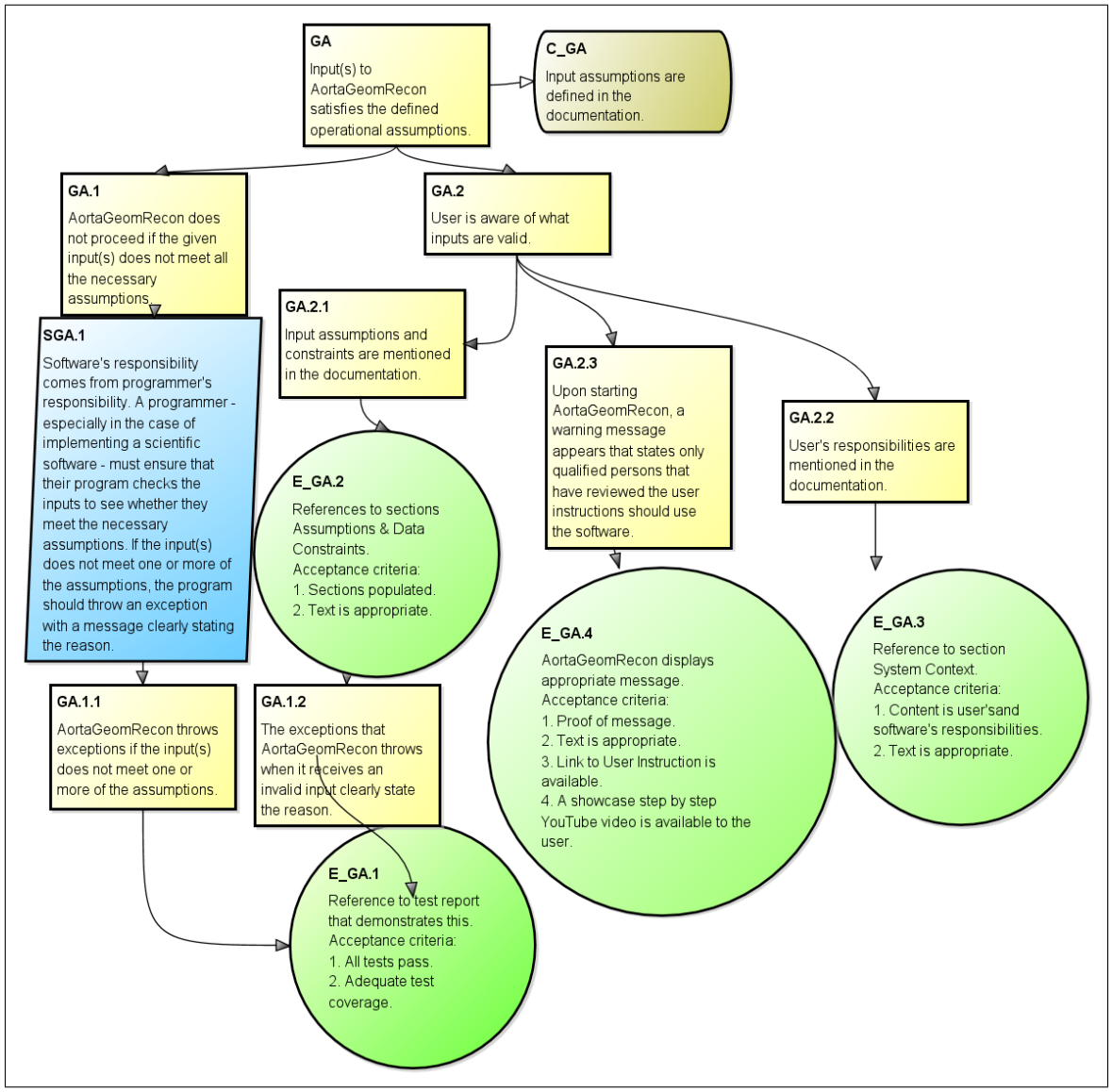


Figure 3.21: AortaGeomRecon Assurance Case Inputs Assumptions

This statement requires the user know what inputs are valid, and only used the valid inputs in the software.

3.5.1 Warning Message

As we initially planned, this piece of information is available in the User Manual and User Instruction Video, where we showed the user how to import DICOM patient's data, and operate on the inputs data till we get a segmentation result. An user who has read the User Manual and watched the instruction video should know what inputs are valid. Therefore, in the AortaGeomRecon module, we need to effectively guide the user to the User Manual, whether the user has used this software before or it is a first time user.

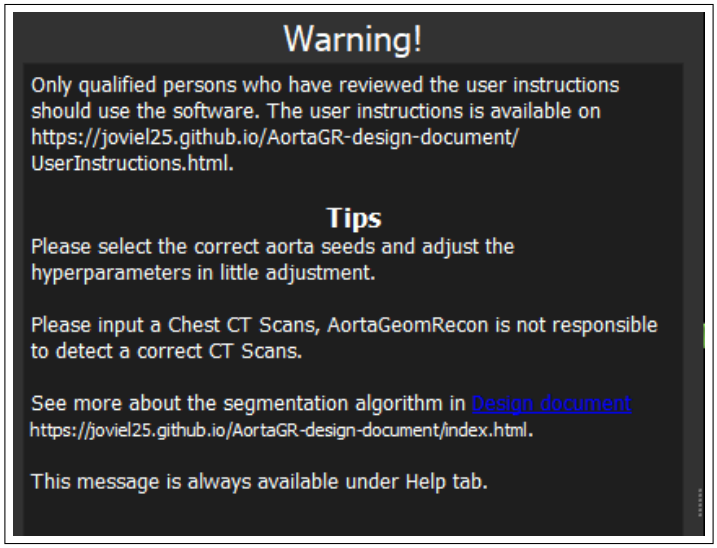


Figure 3.22: AortaGeomRecon Warning Message

As mentioned in the section 2.3.3.2, when the user first starting 3D Slicer and click on the AGR module, this warning message appears. The user must clicks on the Confirm button to continue to to the next steps. With the warning message shown to the user, it is now the user's responsibility to use the valid inputs for AGR, which the program will deliver the correct outputs if the other operations are performed correctly.

Chapter 4

Conclusion and Future Works

4.1 Thesis Summary

4.2 Future Works

(2) 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

[1] GitHub. Github actions documentation. https://docs.github.com/en/actions/using-workflows/about-workflows, 2023.

[2] Kurugol, S., San Jose Estepar, R., Ross, J., and Washko, G. R. 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* (2012), pp. 2343–2346.

[3] Lin, J. Design document. https://joviel25.github.io/AortaGR-design-document/index.html, 2023.

[4] Lin, J. System requirements specification. https://github.com/smiths/aorta/blob/main/docs/SRS/SRS.pdf, 2023.

[5] Spyder. Spyder. https://www.spyder-ide.org/, 2023.

[6] Wikipedia. Lint (software). https://en.wikipedia.org/wiki/Lint_(software), 2023.