

BUILDING AN ASSURANCE CASE FOR
AORTA GEOMETRY RECONSTRUCTION
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

© Copyright by Jingyi Lin, August 2023

All Rights Reserved

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, 105**

Abstract

Assurance cases have been proven to be effective developing a real-time system software. Another domain that requires the high standard correctness, completeness, and consistency 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

I would like to thank all the people who contributed in some way to the work described in this thesis.

First and foremost, I would like to express my sincere thanks and gratitude to my supervisor Dr. Spencer Smith for his motivation, patience, and the continuous support of my master's studies and research. His guidance helped me in all the time of research and writing of this thesis.

Contents

Abstract	iii
Acknowledgements	v
Notation, Definitions, and Abbreviations	xi
Declaration of Academic Achievement	xvi
1 Introduction	1
1.1 Background	3
1.2 Methodology	5
1.3 Thesis Outline	6
2 AortaGeomRecon Research and Development	8
2.1 Existing Methods	9
2.2 Segmentation Algorithm	12
2.3 3D Slicer Extension Development	21
3 Assurance Cases and Selected Evidence for AortaGeomRecon	28
3.1 Assurance Case Development	28

3.2	Assurance Case for Software Specification Requirements	29
3.3	Assurance Case for Implementation	38
3.4	Assurance Case for Operational Assumptions	50
3.5	Assurance Case for Inputs Assumptions	53
4	Conclusion and Future Works	55
4.1	Thesis Summary	55
4.2	Future Works	57
A	Software Requirements Specification for AortaGeomRecon	59
B	Module Guide for AortaGeomRecon	81

List of Figures

1.1	Aorta	3
1.2	Organ Segmentation	4
1.3	Simple Assurance Case Diagram	5
2.1	ITK-Snap's Bubble Segmentation UI	10
2.2	3D Slicer Built-in Segmentation UI	11
2.3	The Aorta Seeds	13
2.4	Level Sets Segmentation	15
2.5	A label image	16
2.6	A Distance Map	17
2.7	Code That Shows How To Calculate The Threshold Range	18
2.8	A Segmented Image	19
2.9	Segmentation Result	21
2.10	Jupyter Notebook Research	22
2.11	3D Slicer UI	24
2.12	AortaGeomRecon Module UI	25
2.13	AortaGeomRecon Warning message	26
3.1	AortaGeomRecon Assurance Cases Top Level	30
3.2	AortaGeomRecon Assurance Cases GR	32

3.3	AortaGeomRecon SRS Assumptions	34
3.4	AortaGeomRecon Functional Requirements	35
3.5	AortaGeomRecon Non- Functional Requirements	35
3.6	AortaGeomRecon Traceability Matrix between Data Definitions and Instance Model	36
3.7	AortaGeomRecon Traceability Matrix Between Requirements and Other sections	37
3.8	AortaGeomRecon Traceability Matrix Between Assumptions and Other sections	37
3.9	AortaGeomRecon Assurance Cases For Implementation	38
3.10	AortaGeomRecon Design Document Website	39
3.11	AortaGeomRecon Design Document Glossary	40
3.12	AortaGeomRecon Anticipated Changes	41
3.13	AortaGeomRecon Modules	42
3.14	AortaGeomRecon Module Decomposition Example	43
3.15	AortaGeomRecon Modules Traceability Matrices	44
3.16	AortaGeomRecon Part Of The Traceability Matrix On Modules And Code	45
3.17	Spyder Variable Explorer	48
3.18	AortaGeomRecon Assurance Case Operational Assumptions	50
3.19	AortaGeomRecon User Manual On GitHub README	51
3.20	AortaGeomRecon User Instructions on YouTube	52
3.21	AortaGeomRecon Assurance Case Inputs Assumptions	53
3.22	AortaGeomRecon Warning Message	54

List of Tables

Notation, Definitions, and Abbreviations

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 outside of the scope of the current work.

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.

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

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.

Binary Dilation

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

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

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).

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 [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 of the region until the maximum difference between the neighboring pixels are reached.

Segmented slice

A 2-dimensional image with interested pixels labeled as 1 and other pixels as 0.

Kernel Size The size of the kernel for binary dilation.

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.

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.

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.

Abbreviations

AC	Assurance Case
AGR	AortaGeomRecon
AortaGeomRecon	
	3D Slicer's extension module, Aorta Geometry Reconstruction
CT	computerized tomography
DD	Design Document
DICOM	Digital Imaging and Communications in Medicine
MG	Module Guide
SITK	SimpleITK
SRS	Software Requirements Specification
VTK	The Visualization Toolkit

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

Medical Software is a critical component of patient diagnosis and treatment. Medical software refers to computer programs, applications, or systems specifically designed for use within the healthcare and medical field. These software solutions are developed to assist healthcare professionals, researchers, administrators, and patients in various aspects of medical care, research, management, and education [18]. Our project focuses on medical software that yields a direct and crucial influence on patients' well-being, particularly software that contributes to diagnosing issues related to the aorta. The aorta, a vital organ responsible for transporting blood from the heart to other bodily organs, holds immense significance. Any malfunction in its blood-carrying function could yield severe and potentially life-threatening consequences for the entire body's physiology. Our objective centers around the Aorta Geometry Reconstruction (AortaGeomRecon or AGR) software, which can build the 3-dimensional model of the aorta, to help the health professional diagnosing issues related to the aorta quickly and correctly.

Given the importance of medical software like AortaGeomRecon, we need a means

to build confidence in the software. In this report, we explore the use of assurance cases. An assurance case can be thought of as a specific type of argumentation used in various cases. The main purpose of an assurance case is to establish confidence and trust in the reliability and safety of a system by presenting a well-structured argument supported by evidence [19]. Assurance cases have been applied regularly in the medical device for approval in U.S. In Europe, the assurance cases are required in systems as diverse as flight control systems, nuclear reactor shutdown systems, and railroad signaling systems, which are all critical systems [19]. Previous works [17] building assurance cases for scientific computing software such as 3dfim+, a medical imaging software analysing activity in the brain, has demonstrated a great success in showing the software's correctness and reliability. The motivation of our project is to build an assurance case for AortaGeomRecon by adding more details on the evidence needed to support our claim, thus building our confidence in AortaGeomRecon.

In this chapter, we first explain in details the contexts for the key concepts that will be discussed throughout the document, including what is organ segmentation, what is aorta, listing the diseases that aorta segmentation could detect, and demonstrates an example of assurance case by showing a simple diagram of assurance case. Next, we will briefly discuss on the methodology, especially how we achieved the objective of design, implementation of the software, and building confidence with the evidences in assurance cases. In the final section, we will explain our thesis outline covering the entire report.

1.1 Background

In this section, we present some contexts on the key concepts within the scope of our work.

1.1.1 Aorta

Aorta is the largest artery that carries blood from the heart to the circulatory system. It has a cane-like shape with ascending aorta, aortic arch and descending aorta. Figure 1.1 shows the entire aorta, but abdominal aorta is outside of the scope of the current work. Our work focus on building the 3D geometry from aortic root to descending aorta.

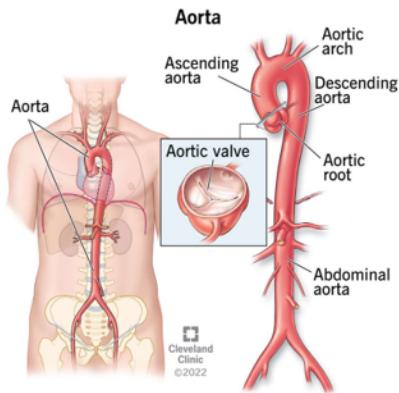


Figure 1.1: Aorta

1.1.2 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. Figure 1.2 demonstrates an example of abdominal organ segmentation.

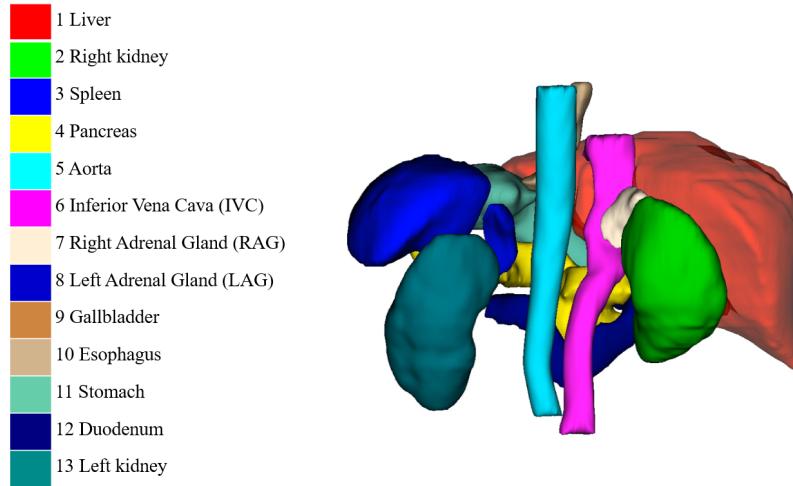


Figure 1.2: Organ Segmentation [15]

Aorta segmentation in computerized tomography (CT) scans is important for:

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

1.1.3 Assurance Case

An Assurance Case (AC) can be thought of as a specific type of argumentation used in various cases. When building an AC, you’re making a point that specific evidence backs up a particular statement. The fundamental structure is depicted in Figure

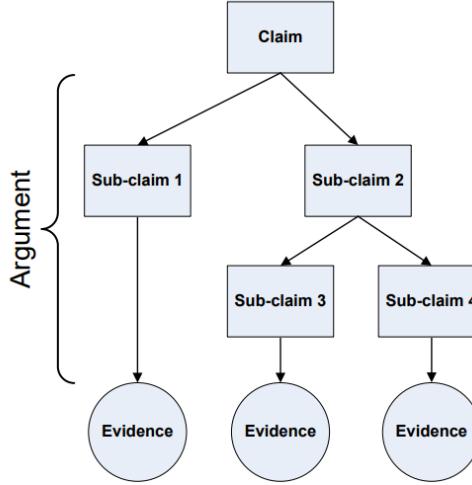


Figure 1.3: Simple Assurance Case Diagram [4]

[1.3](#). So, an AC essentially boils down to an organized collection of arguments, backed by a body of evidence, that helps validate the belief in a specific claim [4].

In a practical sense, creating an AC involves beginning with a main claim and then breaking it down into smaller claims through a step-by-step process. These smaller claims, at the most bottom, are supported by concrete evidence.

1.2 Methodology

In this study, we present the outcomes of integrating AC throughout the development of medical software to reinforce stakeholders' confidence in the software's capabilities. The software, known as AortaGeomRecon, represents a 3D Slicer [10] extension module designed to semi-automatically construct a 3D model of the aorta using CT scans from a patient's chest. We started by gathering requirements for the AortaGeomRecon from a domain expert, drafted our Software Requirements Specification (SRS), and Module Guide (MG). We researched and worked on the implementation of the

software, while building the infrastructure for continuous integration, version control, and project management using GitHub. When we have a functional prototype, we delved into our assurance cases, encompassing chosen arguments and supporting evidence. AC functions as a method to provide assurance for a system by presenting arguments that substantiate claims about the system. These arguments are based on evidence related to the system’s design, development, and tested behavior. By constructing AC, we were able to follow the best practice including documentation review on SRS and MG to finalize our documentation, ensure the documentation’s completeness and correctness. We have built a user documentation to define all operational assumptions, and guide user to use the valid inputs with a sequence of correct operations. Finally, our continuous integration tests, code review, and several algorithm reviews reinforced our confidence in the implementation of the software, which has strictly complying with the requirements that are complete and correct.

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 functions finally, and the 3D Slicer [10] extension module that the user interacts with to get the segmentation result with our algorithm. In Chapter 3, we present our AC and focusing on the evidence, including 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. In Chapter 4, future work is proposed and conclusions are drawn

based on the developed SRS, segmentation algorithm, 3D Slicer module extension, and AC.

Chapter 2

AortaGeomRecon Research and Development

This chapter will discuss the research and development of the 3D Slicer plugin AortaGeomRecon.

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 of 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 AortaGeomRecon 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 their 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” [5].

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 guaranteed 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

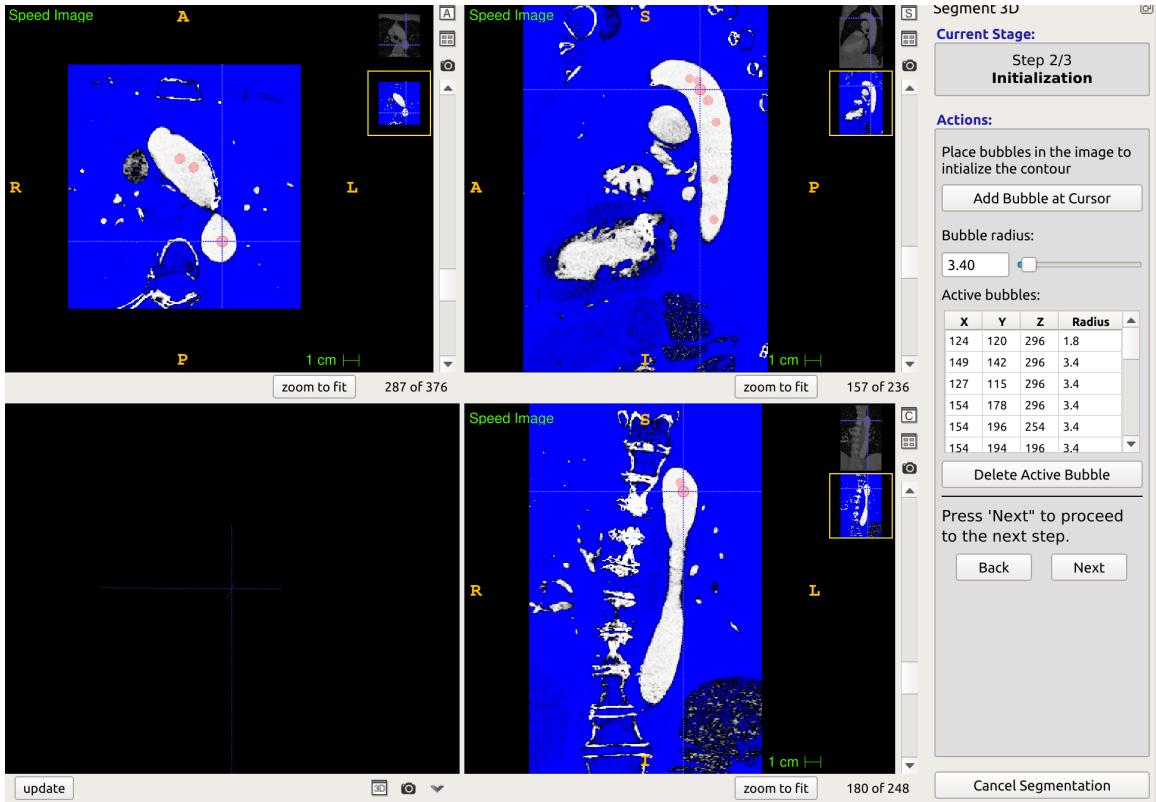


Figure 2.1: ITK-Snap’s Bubble Segmentation UI [21]

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 and Communications in Medicine (DICOM) format [6], 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.

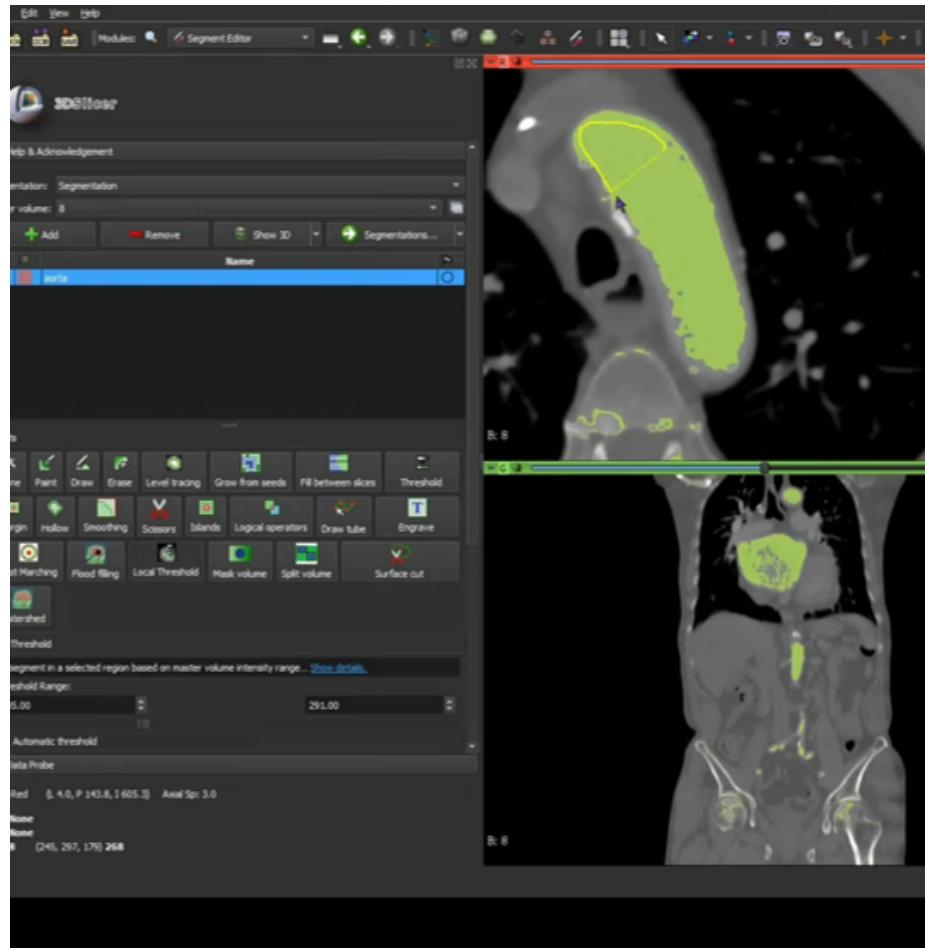


Figure 2.2: 3D Slicer Built-in Segmentation Method [10]

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’s 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 introduce the contexts for external 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][14]. NumPy is the fundamental package for scientific computing with Python, especially for the performance of multidimensional array processing [8]. 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.

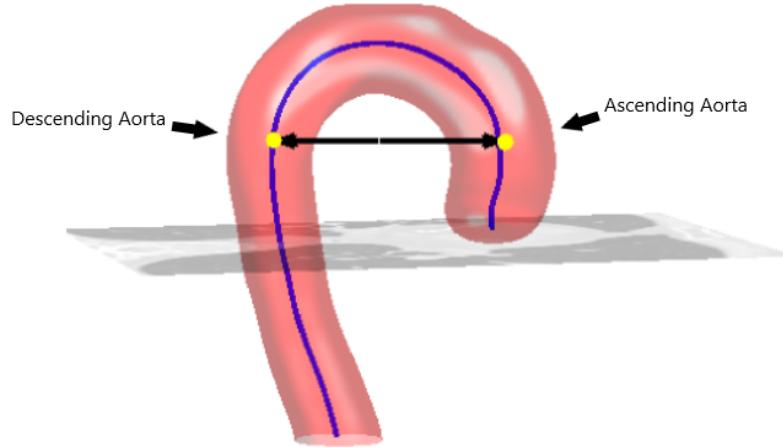


Figure 2.3: The aorta seeds [12]

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.

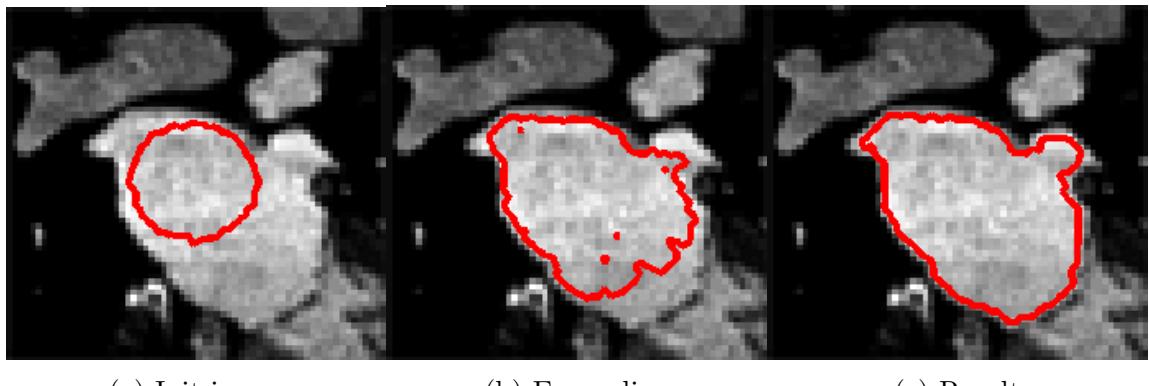


Figure 3.4: Level Sets Segmentation

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.

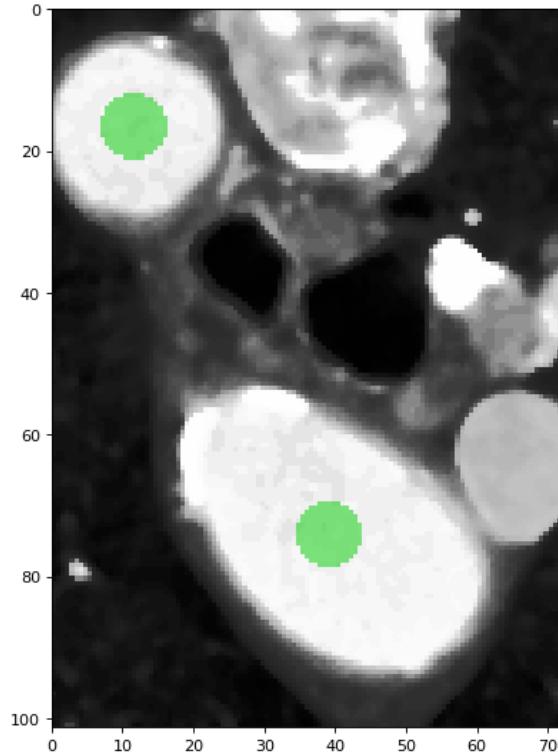


Figure 2.5: A Label Map

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.

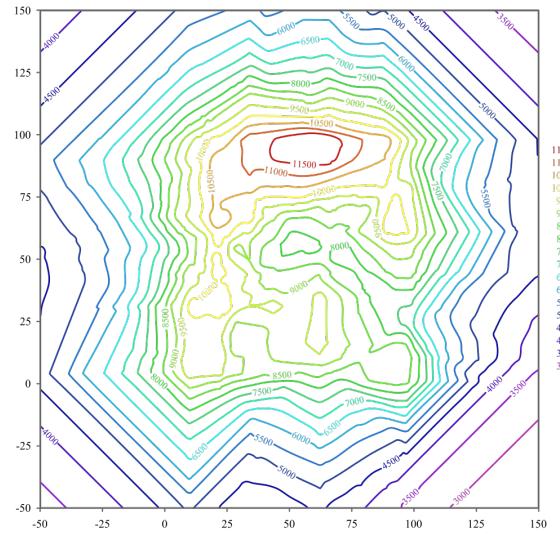


Figure 2.6: A Distance Map

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.

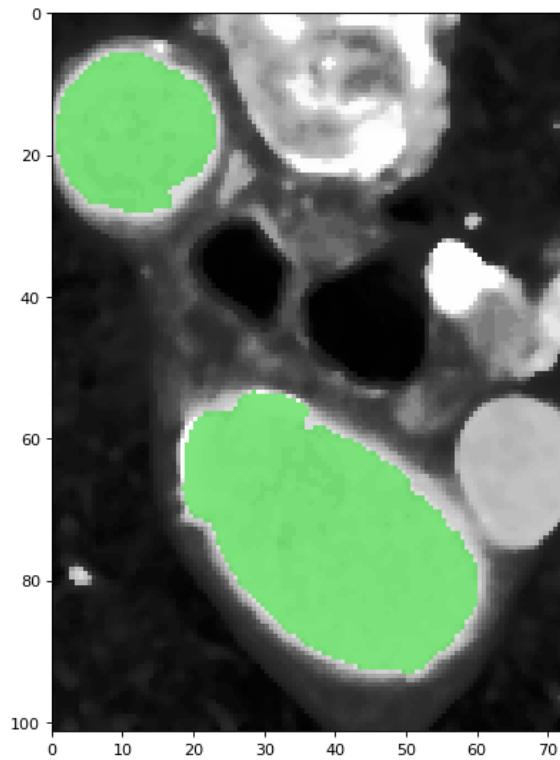


Figure 2.8: A Segmented Image On Top Of The Original Slice

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 uses 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.

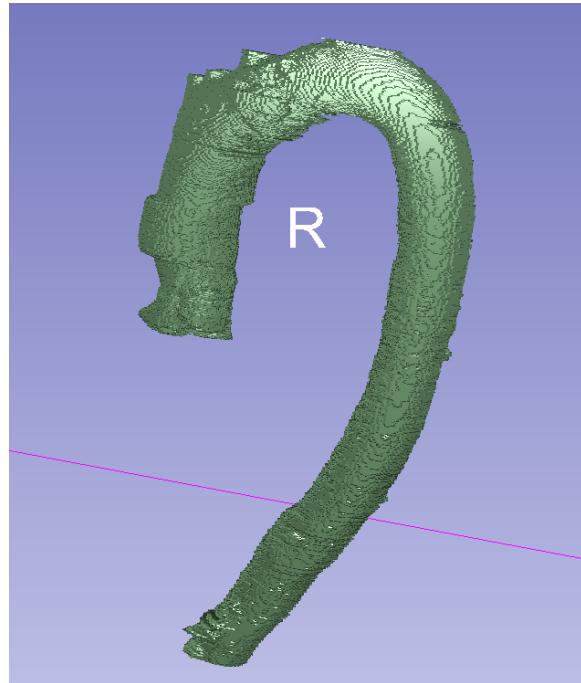
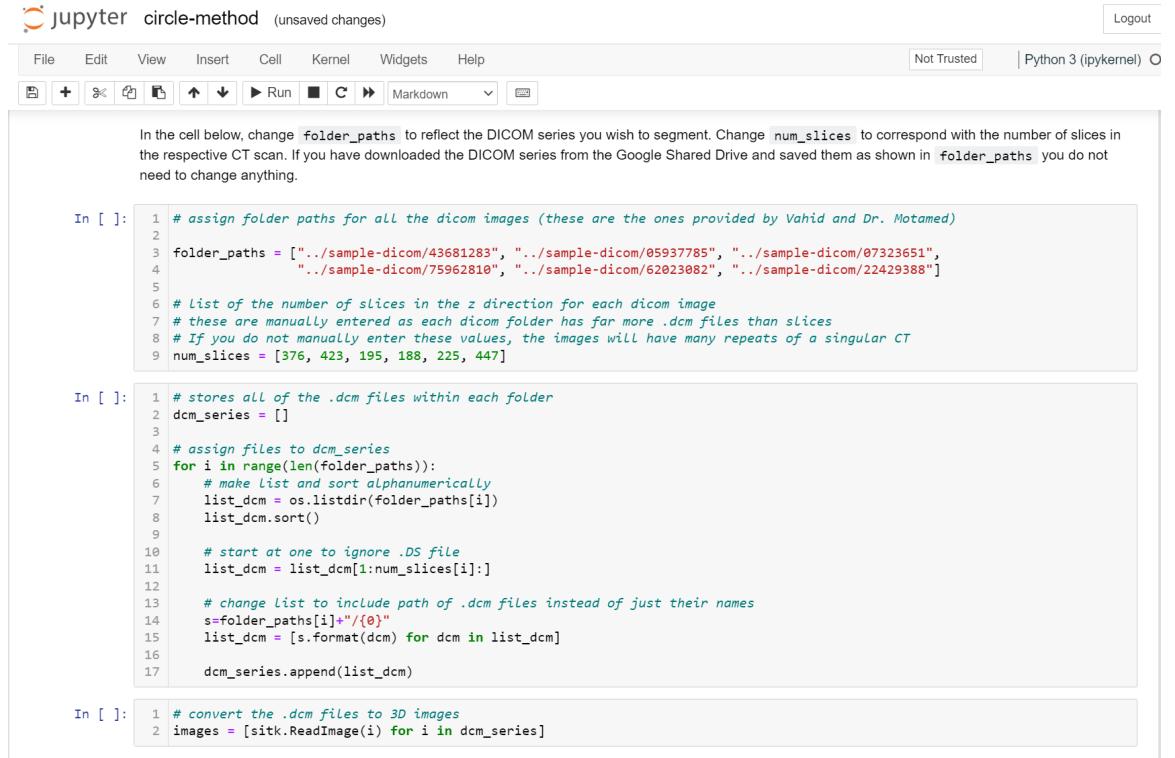


Figure 2.9: Segmentation Result

2.3 3D Slicer Extension Development

The project has started with a simple segmentation program build in Jupyter Notebook [11]. 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).



The screenshot shows a Jupyter Notebook interface with the title "circle-method" and a note "(unsaved changes)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Logout, Not Trusted, and Python 3 (ipykernel). The notebook contains three code cells:

```

In [ ]: 1 # assign folder paths for all the dicom images (these are the ones provided by Vahid and Dr. Motamed)
2
3 folder_paths = ["./sample-dicom/43681283", "./sample-dicom/05937785", "./sample-dicom/07323651",
4   "./sample-dicom/75962810", "./sample-dicom/62023082", "./sample-dicom/22429388"]
5
6 # List of the number of slices in the z direction for each dicom image
7 # these are manually entered as each dicom folder has far more .dcm files than slices
8 # If you do not manually enter these values, the images will have many repeats of a singular CT
9 num_slices = [376, 423, 195, 188, 225, 447]

In [ ]: 1 # stores all of the .dcm files within each folder
2 dcm_series = []
3
4 # assign files to dcm_series
5 for i in range(len(folder_paths)):
6   # make List and sort alphanumerically
7   list_dcm = os.listdir(folder_paths[i])
8   list_dcm.sort()
9
10  # start at one to ignore .DS file
11  list_dcm = list_dcm[1:num_slices[i]:]
12
13  # change List to include path of .dcm files instead of just their names
14  s=folder_paths[i]+"\{0\}"
15  list_dcm = [s.format(dcm) for dcm in list_dcm]
16
17  dcm_series.append(list_dcm)

In [ ]: 1 # convert the .dcm files to 3D images
2 images = [sitk.ReadImage(i) for i in dcm_series]

```

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 Segmener object with the attributes set to the parameters reading from UI component, which are inputs by the user. After completing the segmentation with Segmener 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 AortaGeomRecon. 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

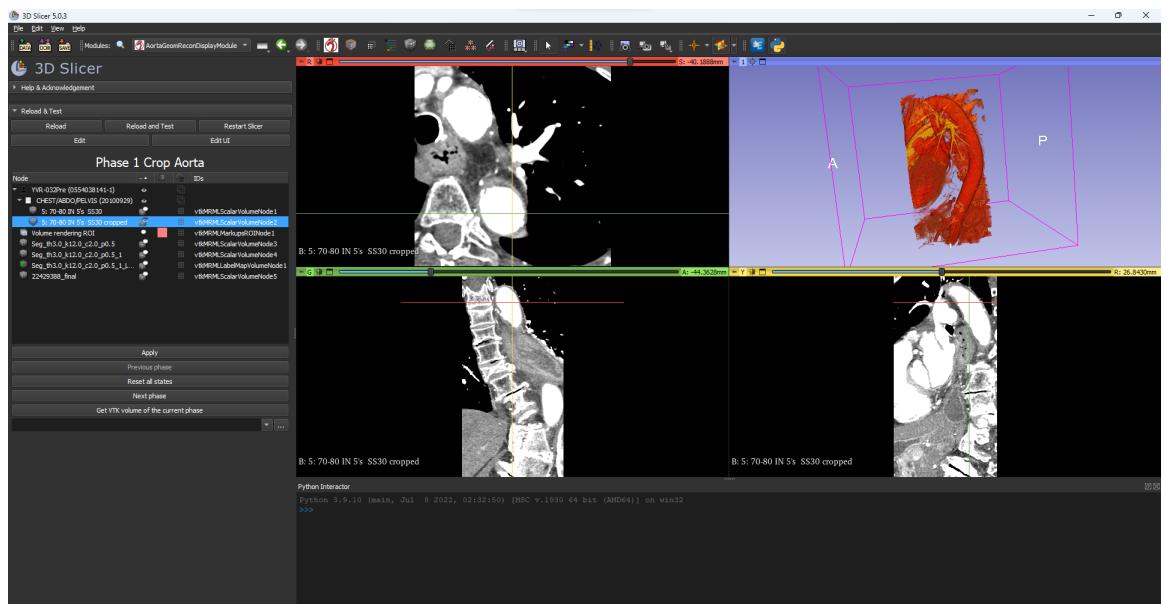


Figure 2.11: 3D Slicer UI

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 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.

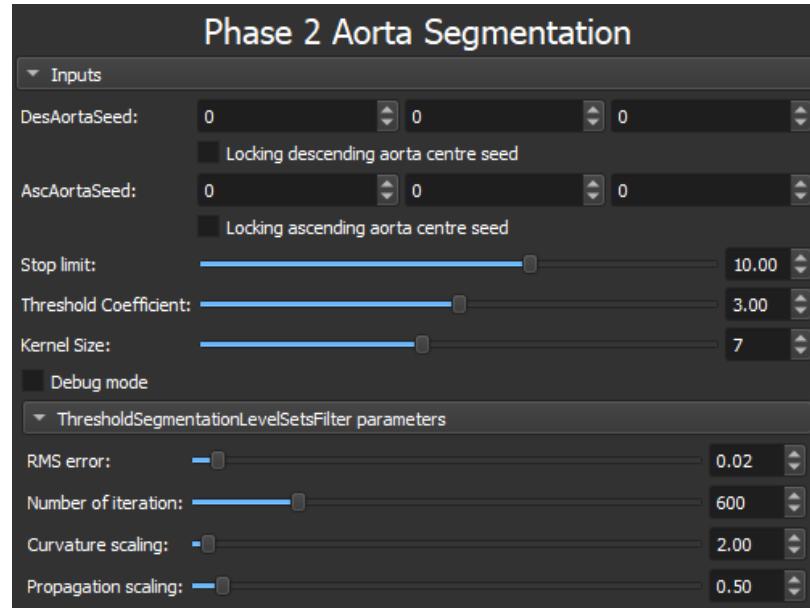


Figure 2.12: AortaGeomRecon Module UI

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.

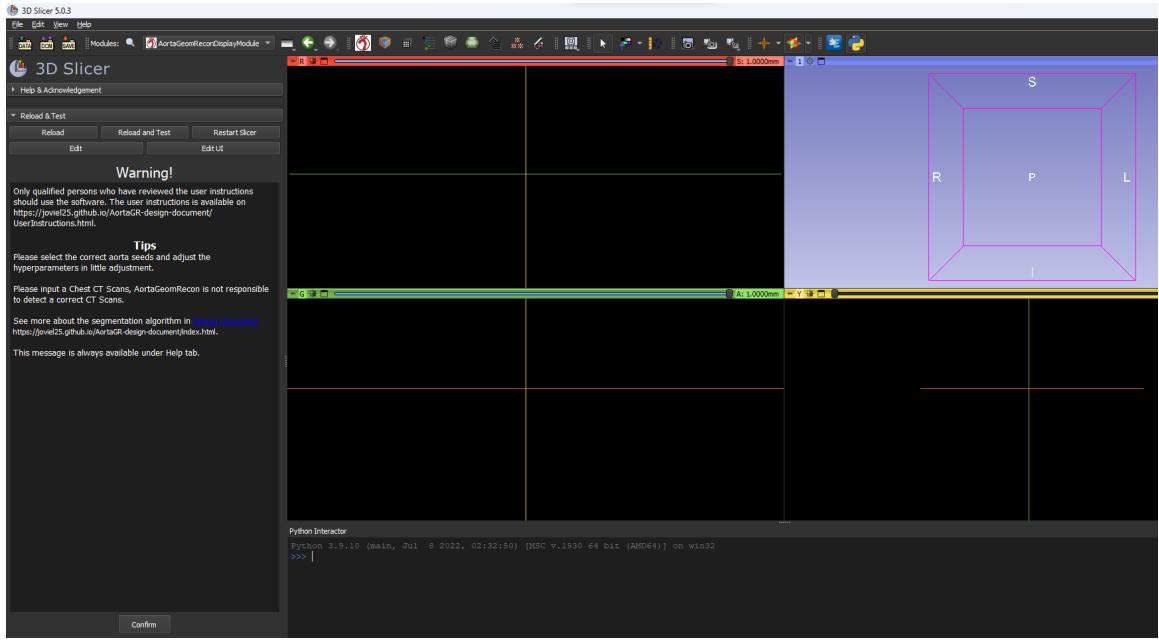


Figure 2.13: AortaGeomRecon Warning message

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 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.

Chapter 3

Assurance Cases and Selected Evidence for AortaGeomRecon

In this chapter, we discuss the scope of our work, which is building the evidence to support each claim of our AC for AortaGeomRecon. The top level claims of the AC developed in previous work [17] is correct and complete, thus we have a list of evidence that can support the arguments and the top level claims. Our work focus on providing the correct evidence for the assurance case, and the following material is presented: the Software Requirements Specification of AortaGeomRecon, the Design Document, the Module Guide, the Test Plan, the Algorithm Review, the User Manual, the User Instruction Video, and a Warning Message implemented in AortaGeomRecon.

3.1 Assurance Case Development

Assurance Case is build with claims, subclaims, contexts and the evidence. The parts of the AC add up to an argument for why the top level claim is true. By using Astah

System Safety software to present the Goal Structuring Notation (GSN) arguments [2][9], 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. With the goal of arguing that the software delivers correct outputs, we decompose the goal into 4 sub goals: GR, GI, GA and GBA, where GR stands for the goal of correct requirements of the software, GI stands for the goal of the implementation matching the requirements, GBA is the goal of that all operational assumptions have been defined, and GA is the goal that all operational assumptions are met.

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 SRS that shows the complete breakdown of the requirements with mathematical notation, data models and instance models. The SRS is the foundation of the software development, and the design and the implementation will be based on the requirement document.

The Figure 3.2 demonstrates the claims on the goal of correct requirements of AortaGeomRecon. On the left branch, GR_3C implies the goal of a complete, correct, and consistent documentation. Under this claim, the goal is separated based on each characteristic, and the corresponding evidence is presented as the leaf node. The S_Correctness and S_Completeness.4 node require a domain expert to review the quality of the document to ensure the documentation met the goal of the claims. The

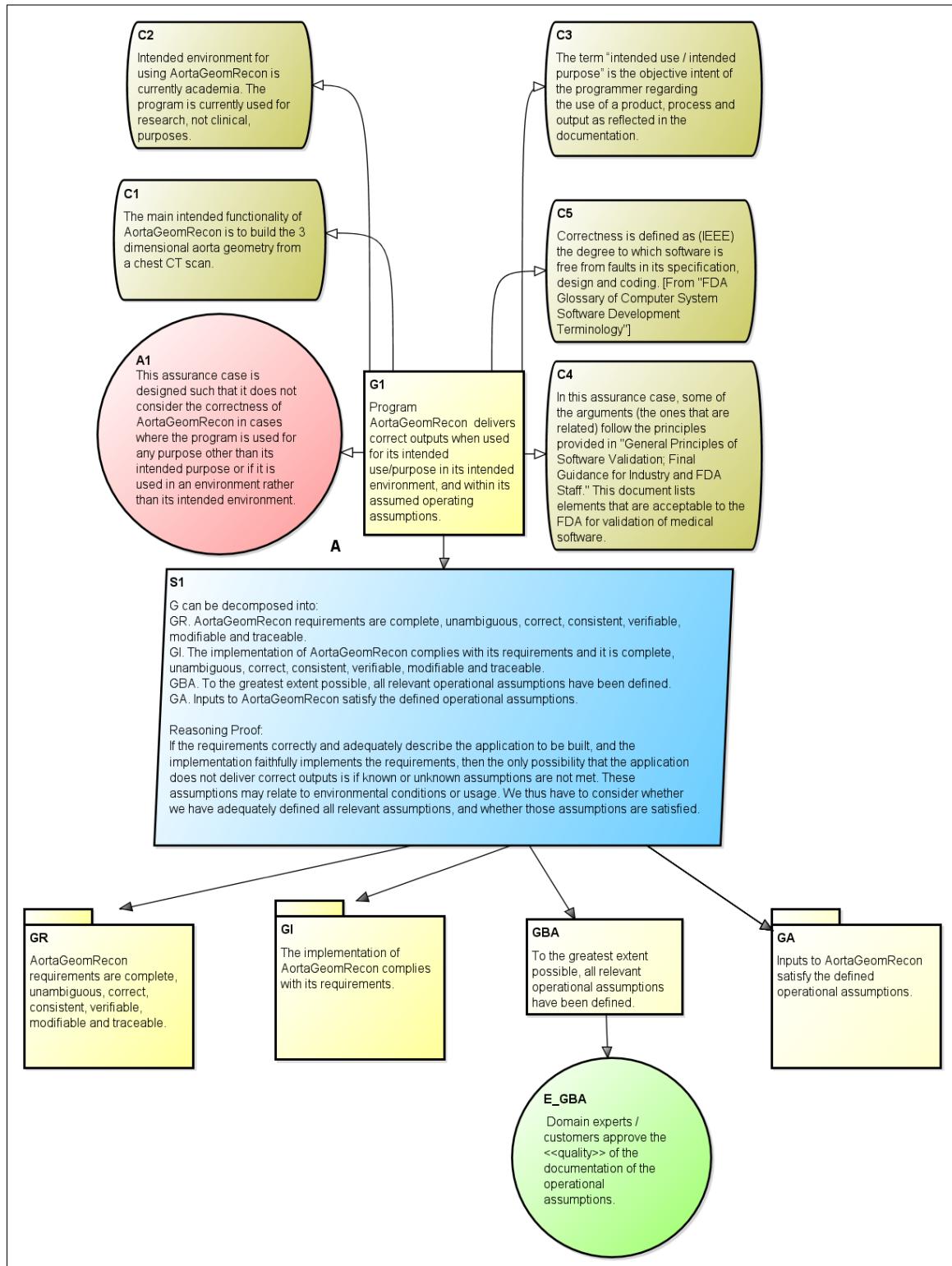


Figure 3.1: AortaGeomRecon Assurance Cases Top Level

other characteristics of the documentation are supported with evidence as shown in the other branches in GR.

As explains in the assurance case GR, one of most important statement of SRS having these characteristics is using a standard template. Using a standard template means that all necessary requirements have been defined, and it allows a domain expert who has used this template to verify the quality of the document. We used a template tailored for research software ??, which is the standard template in the evidence E_Completeness.1, and all other evidences where a template is included.

The chapters in the 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, as the evidences E_Conistency.1 and E_Unambiguous.1 shown in the Figure 3.2. 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. This is provided for the reader to eliminates unambiguous in reading the document.

- General System Description

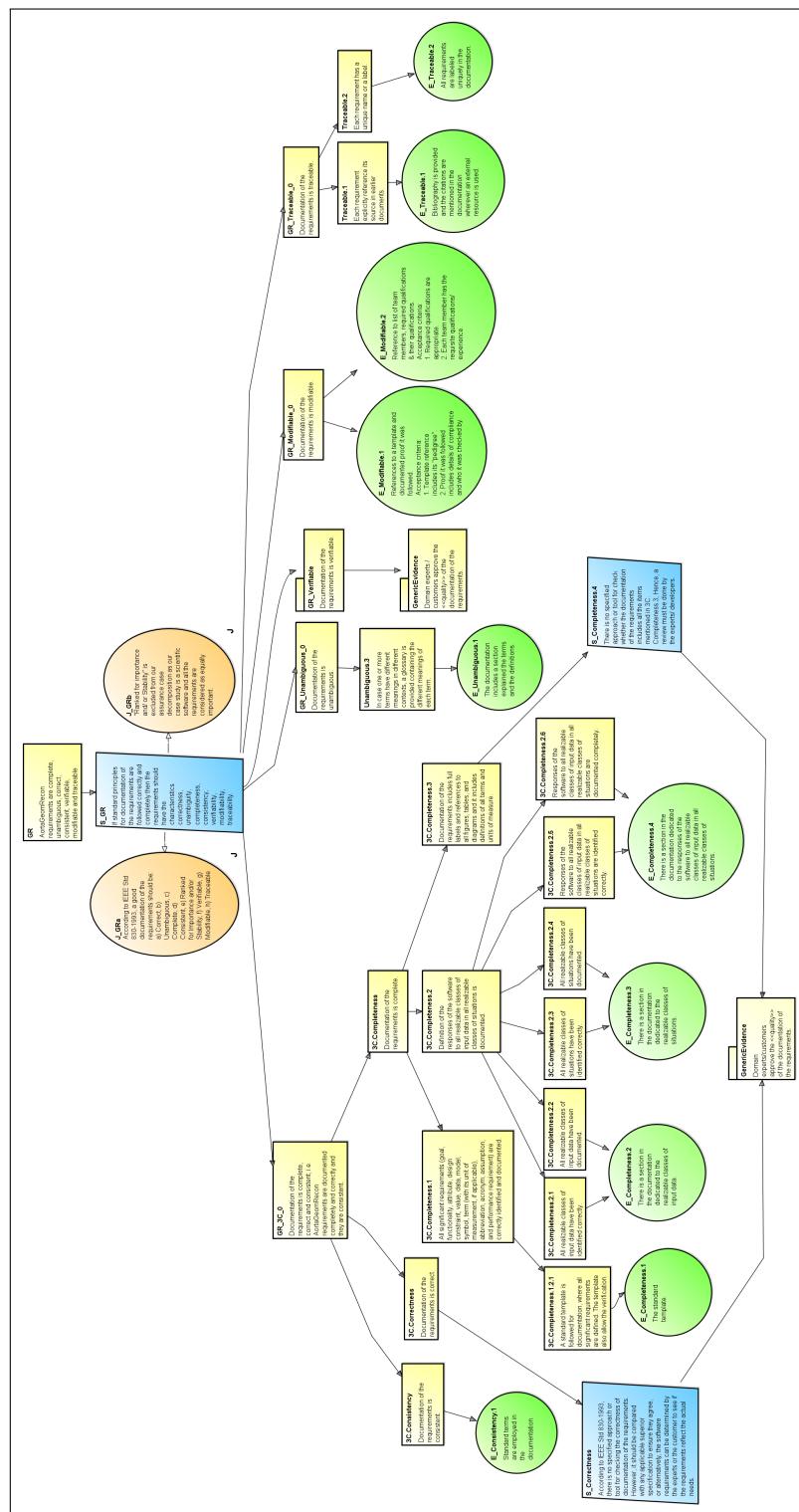


Figure 3.2: AortaGeomRecon Assurance Cases GR

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 problem, 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 the assumptions to clearly define the scope of the requirement document, as shown in Figure 3.3. 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 models 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 distinguishable 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, as shown in Figure 3.4. 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 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		
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.

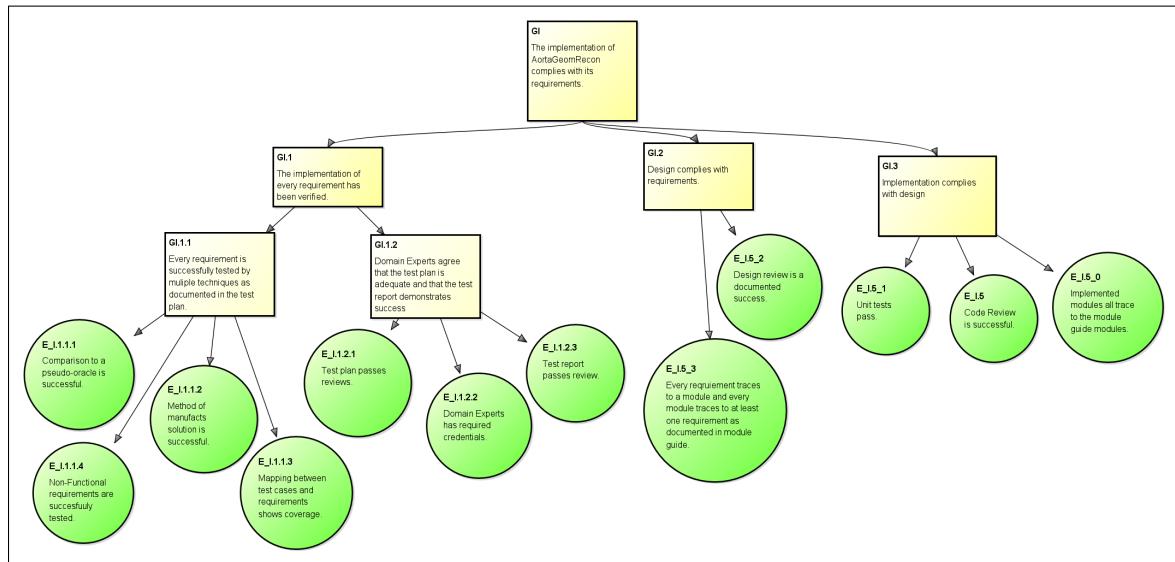


Figure 3.9: AortaGeomRecon Assurance Cases For Implementation

3.3.1 Design Document

The purpose of the Design Document [13] 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 plain text the workflow of the algorithm. The design document is a piece of evidences that demonstrate unambiguity characteristic.

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. 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.

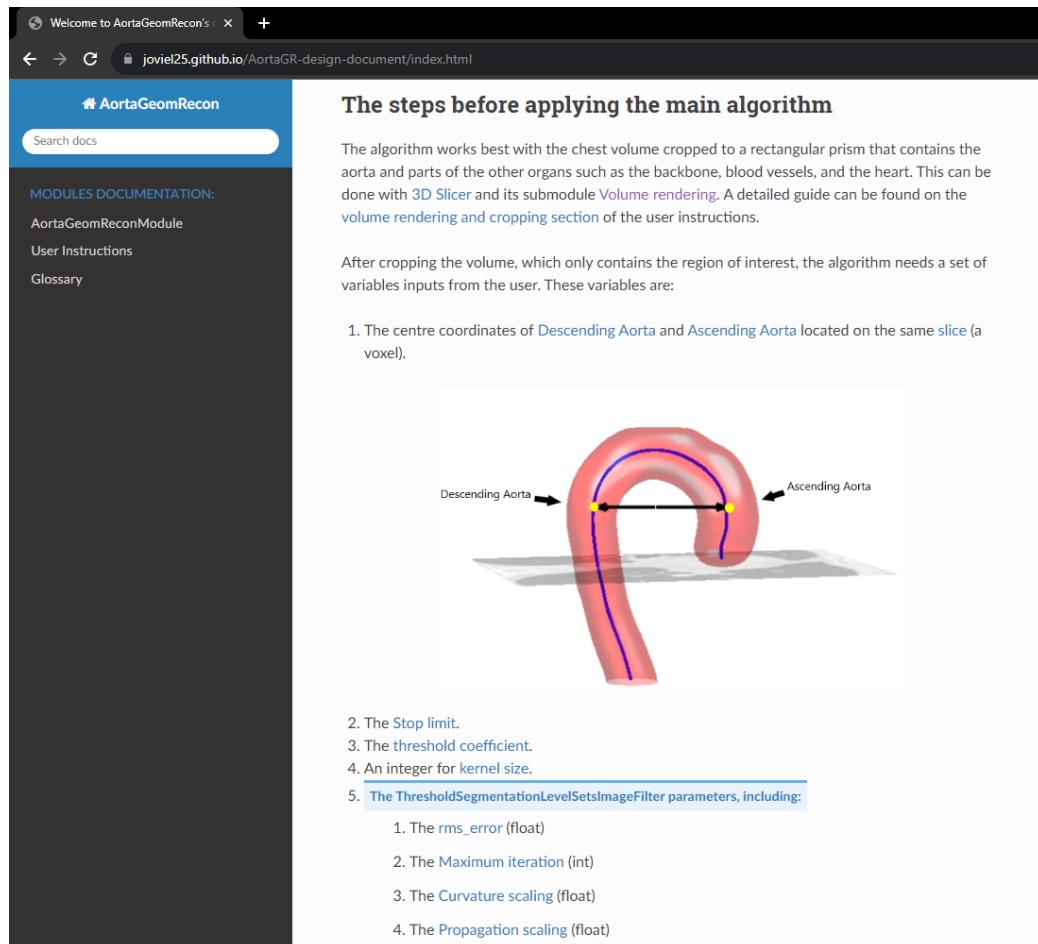


Figure 3.10: AortaGeomRecon Design Document Website

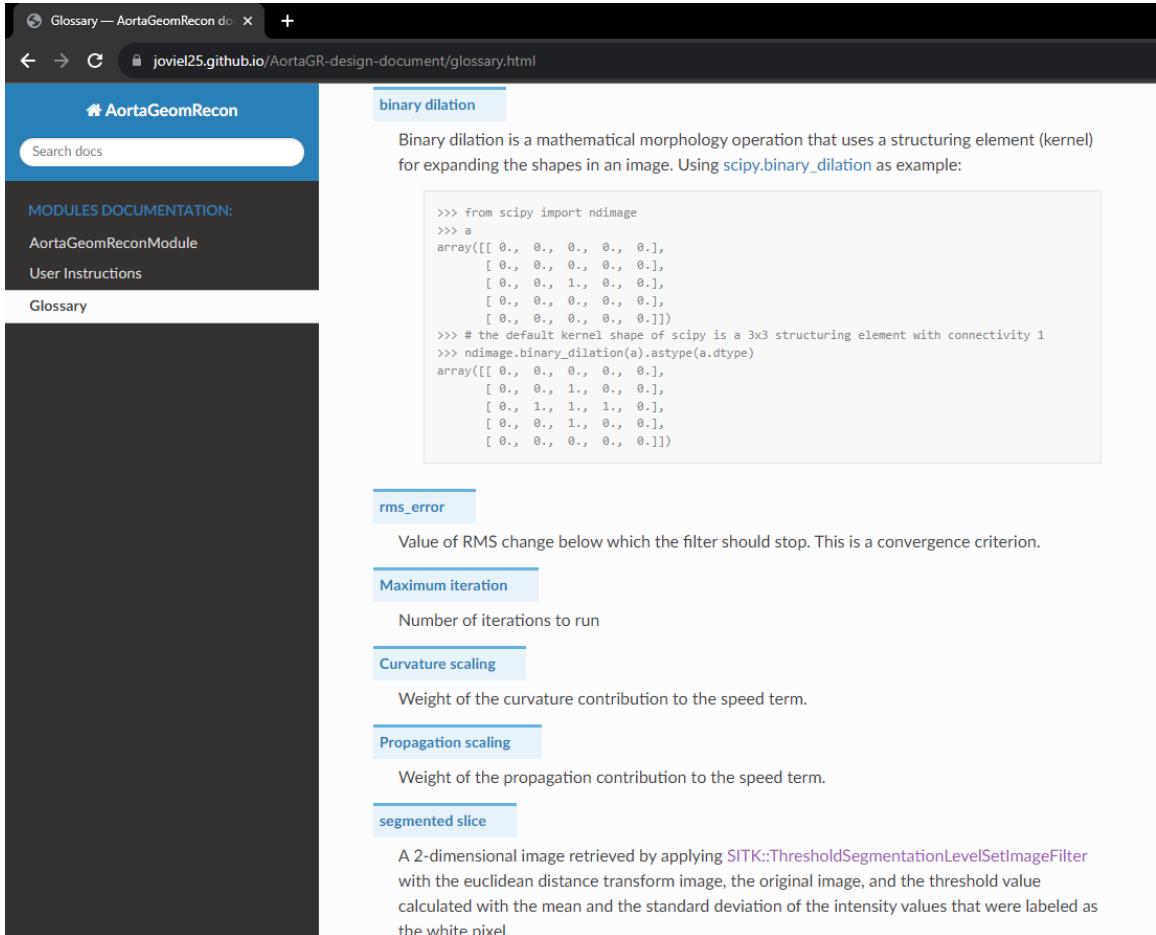


Figure 3.11: AortaGeomRecon Design Document Glossary

3.3.2 Module Guide

Another important documentation to show that the design is complete, correct, and consistent design is Module Guide. In the design of the software, we first list the anticipated changes such that 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' attributes](#)

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	<pre># 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</pre>

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 claim 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 infrastructure, implemented with GitHub Actions workflow. A workflow is a configurable and 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. [7]

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 have 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 striking the source code with Google's published Python Style Guide. [20]

The second automated process is our continuous integration tests. By setting up Git Large File System (LFS) and upload to 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.

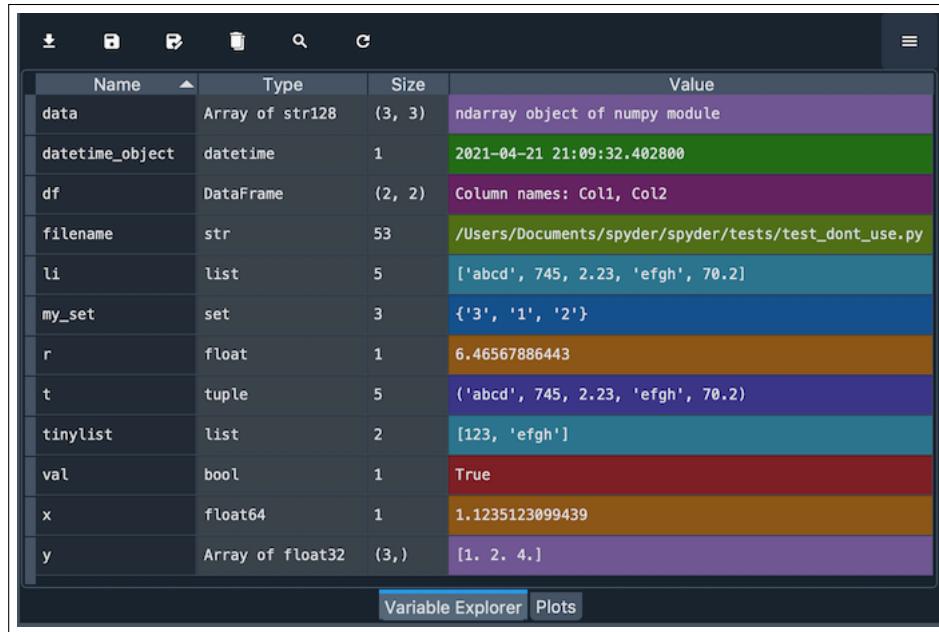
3.3.4 Algorithm Review

The algorithm review was an idea started with Code Walkthrough. A code walkthrough is one of the methods that can increase the participants' confidence, or finding program's bugs or error. 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 [?] .



The screenshot shows the Spyder Variable Explorer window. It displays a table with columns for Name, Type, Size, and Value. The table contains the following data:

Name	Type	Size	Value
data	Array of str128	(3, 3)	ndarray object of numpy module
datetime_object	datetime	1	2021-04-21 21:09:32.402800
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	53	/Users/Documents/spyder/spyder/tests/test_dont_use.py
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
my_set	set	3	{'3', '1', '2'}
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylist	list	2	[123, 'efgh']
val	bool	1	True
x	float64	1	1.1235123099439
y	Array of float32	(3,)	[1. 2. 4.]

Figure 3.17: Spyder Variable Explorer [16]

This feature allows us to execute the program step by step, and see what happens to the variable (segmentation result) 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 engineering student and started working the semi-automacial aorta segmentation algorithm as a summer researcher. Along with Smith Spencer, we were aiming to increase our 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, so some details and design decisions are missing, and some variables were decided

by trial and error. Despite that the code walkthrough has turned into an algorithm review, and it 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 conducted with Dr. Dean Inglis, an experienced professor, Medical Image Analyst, and Software Developer. We presented our segmentation algorithm to him and requested validation of our approach or suggestions for a potentially superior algorithm. Dr. Dean Inglis provided his insights on the algorithm, which we meticulously recorded on the GitHub issue tracker. These insights will guide the developer responsible for enhancing the program. This meeting significantly reinforced our confidence in both the software and our endeavors, whether the idea of developing an extension module for 3D Slicer or the algorithm itself.

3.4 Assurance Case for Operational Assumptions

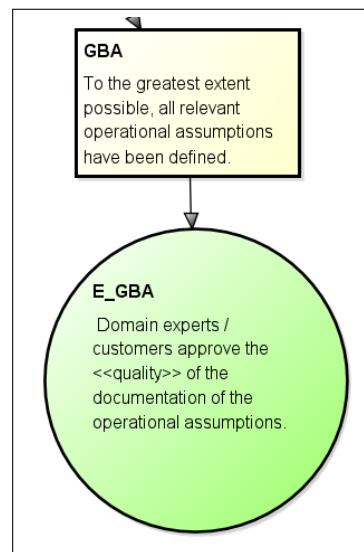


Figure 3.18: AortaGeomRecon Assurance Case Operational Assumptions

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. However, finalizing this evidence takes many efforts from the beginning of the project to the end of the project, because we want to continuously improve the quality of the content matching the most recent updates of the software.

3.4.1 User Manual

A 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.

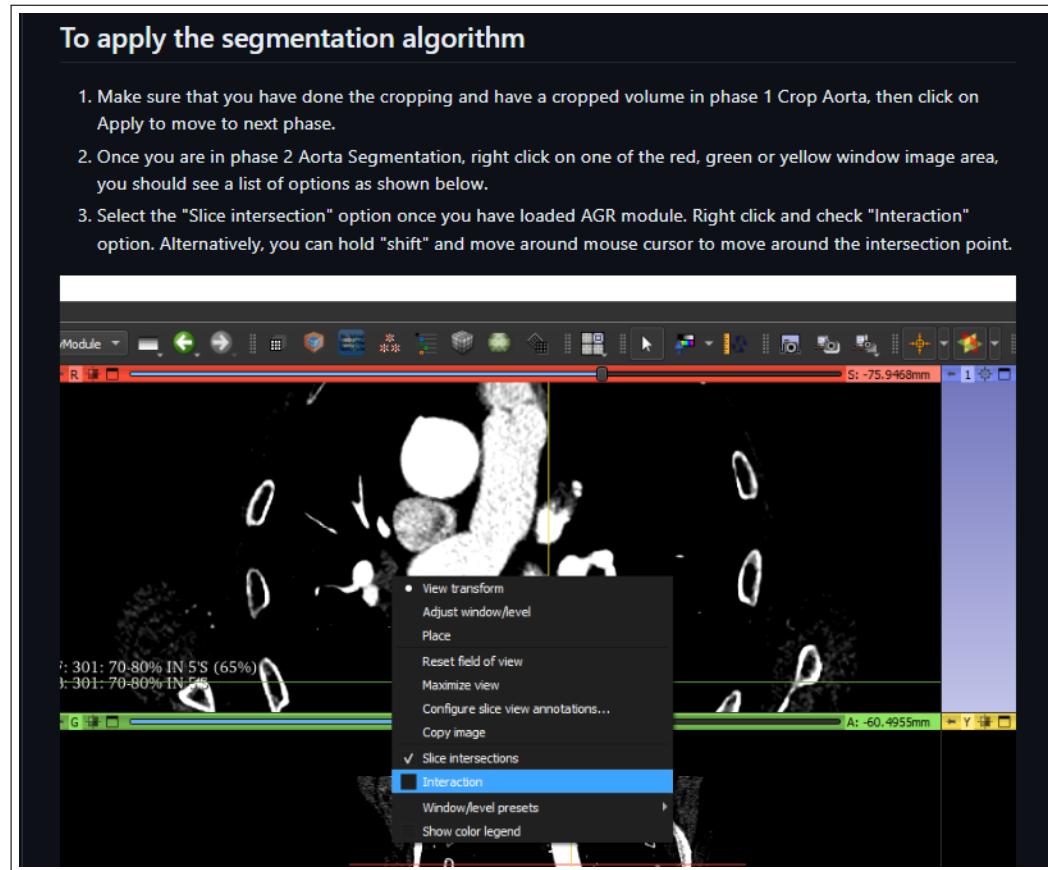


Figure 3.19: AortaGeomRecon User Manual On GitHub README

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.

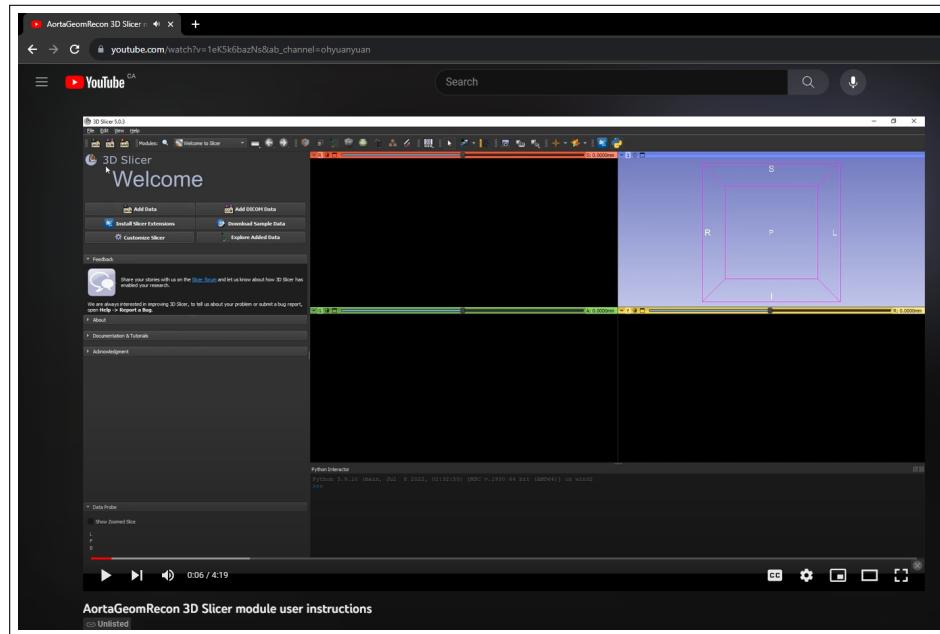


Figure 3.20: AortaGeomRecon User Instructions on YouTube

3.5 Assurance Case for Inputs Assumptions

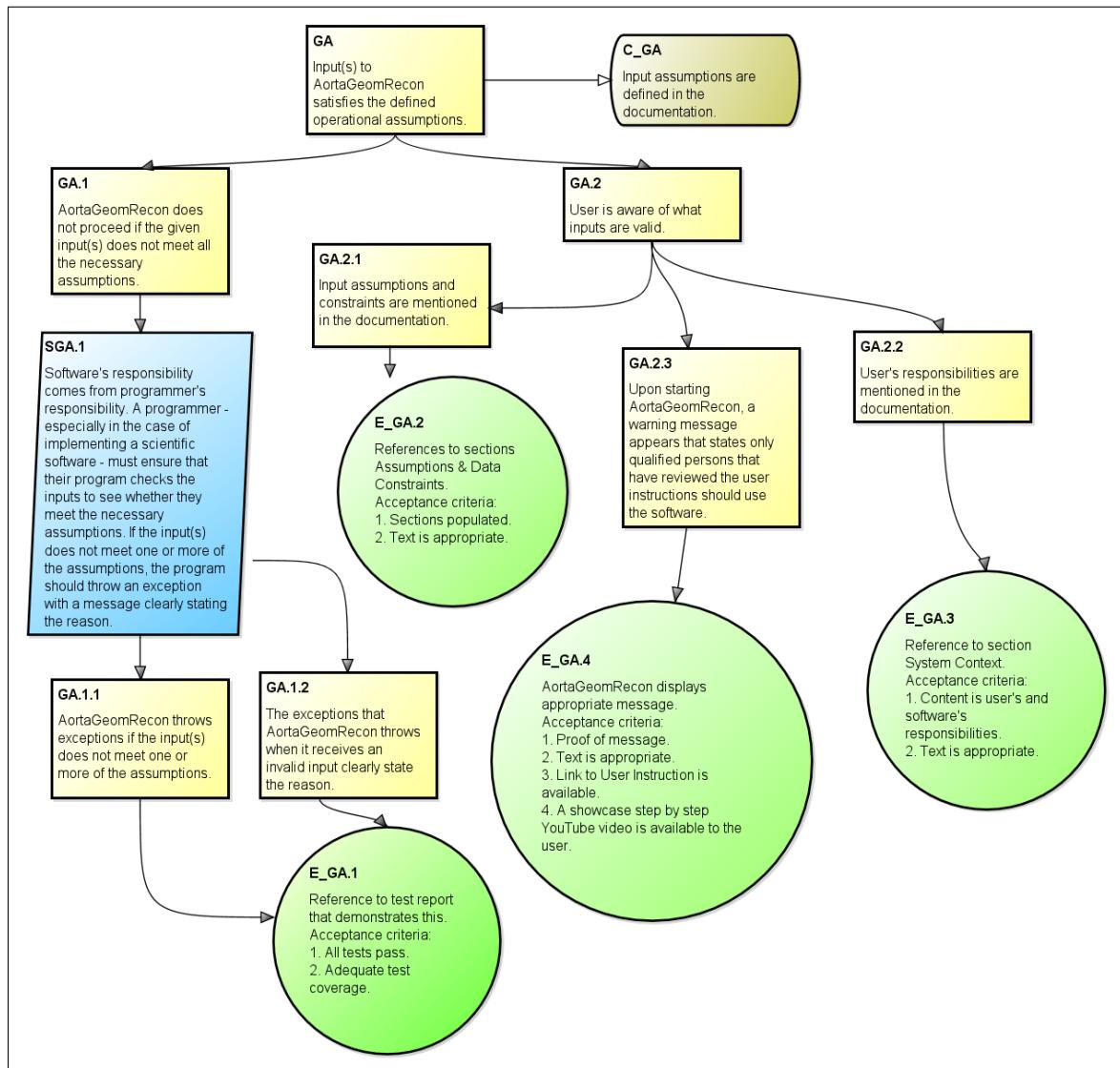


Figure 3.21: AortaGeomRecon Assurance Case Inputs Assumptions

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. A 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.

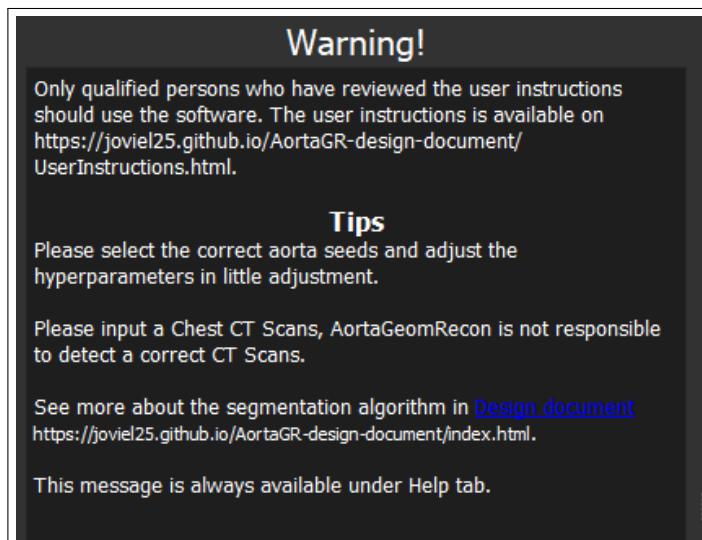


Figure 3.22: AortaGeomRecon Warning Message

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 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

In this chapter, we provide a summary of the thesis as well as the future work.

4.1 Thesis Summary

In this project, we developed a software as a 3D Slicer extension to semi-automatically extract the 3D geometry of the aorta, while applying assurance case arguments to build our confidence in this medical software. The project has started from a Jupyter Notebook program, then explored from the point of view of a patient such that what can we do to trust the software enough to let the doctor using it while being confident and certain that the software will not fail our expectation. We have then developed an idea of developing a software that is first convenient to use, at least on gathering parameters. This lead to developing a 3D Slicer extension module, because 3D Slicer already provides useful features such as Volume Rendering, Volume visualization, Crop Volume, etc. While building the software, we build the assurance cases in Goal Structuring Notation with the bottom-up approaches; we keep ask ourselves

what are the evidences that are necessary to support our claim in requirements, implementation, operational assumptions and inputs assumptions? Keep that question in mind, we have finalized our SRS document, Module Guide document, wrote user instructions, built design document in HTML and publish it on a website, and linking all assurance cases to support our arguments. Finally, we have used GitHub's features for Continuous Integration infrastructure, and project management. We have built 2 automated process that act as a linter and continuous integration tests, these processes helped a lot in detecting bugs and errors in implementation. As for project management, GitHub Issues, Discussions, and Pull requests were used throughout the development of the software. Me and Dr. Spencer Smith were able to keep up good communication through these features.

In the course of this project, we have summarized a list of challenges and tasks that we could have done better. The first challenge was looking for an ideal platform to develop AortaGeomRecon software. Until the point where we see that it is nearly impossible to build from scratch a volume visualization system like the volume visualization provided by 3D Slicer, time and efforts have been wasted in design a UI, finding the right tool to build the UI, etc. 3D Slicer itself is a very complex software, the development resource is limited and difficult to understand.

Another obstacle that we have is having a domain expert to examine the quality of our segmentation result and other documentation. This medical software's intended user is a university student studying in medical science or medicine, who likes to get an aorta's image or quantified volume. Throughout the development of the AortaGeomRecon, we did not have an intended user or a domain expert to review our software. However, me and Dr. Spencer Smith were also lacking the knowledge

and do not know the expectation of the intender user or domain expert, this causes ambiguity in understanding the true requirements of AortaGeomRecon.

Finally, it is very challenging of understanding assurance cases with a limited time, and building the assurance cases for AortaGeomRecon was unclear for me. Gathering the evidences and support our arguments was not in my imagination at the beginning of the project, without truely understanding our goals of the project, I was not certain what I was really doing for this project. Until we have several pieces linking together, I was finally understanding and making more efforts in the right direction.

4.2 Future Works

In this section, we will discuss some possible future works that can continue to make AortaGeomRecon better, the first improvement can be done in segmentation algorithm, and the second improvement can finalize our assurance case.

4.2.1 Segmentation Algorithm

In this paper [12], we were able to discover a new segmentation algorithm that also needs a cropped volume and the aorta seeds to perform segmentation. However, it required less hyperparameters such as the parameters for SimpleITK’s Threshold-SegmentationLevelSetsFilter. Using this algorithm effectively reduces the number of hyperparameters, which lead to a better and safer segmentation results.

4.2.2 Assurance Case

There is room for improvements on the arguments of the requirements of Aorta-GeomRecon. The correctness of the document is reviewed and approved by a domain expert, where there should be evidences that can support the argument.

Appendix A

Software Requirements

Specification for AortaGeomRecon

Software Requirements Specification for AortaGeomRecon

Jingyi Lin

July 2, 2023

Contents

1	Reference Material	iv
1.1	Table of Units	iv
1.2	Table of Symbols	iv
1.3	Abbreviations and Acronyms	v
2	Introduction	1
2.1	Purpose of Document	1
2.2	Scope of Requirements	1
2.3	Characteristics of Intended Reader	1
2.4	Organization of Document	1
3	General System Description	2
3.1	System Context	2
3.2	User Characteristics	3
3.3	System Constraints	3
4	Specific System Description	3
4.1	Problem Description	3
4.1.1	Organ Segmentation	3
4.1.2	Coordinate Systems	4
4.1.3	Physical System Description	6
4.1.4	Goal Statements	7
4.2	Solution Characteristics Specification	7
4.2.1	Assumptions	7
4.2.2	Theoretical Models	7
4.2.3	General Definitions	7
4.2.4	Data Definitions	7
4.2.5	Data Types	9
4.2.6	Instance Models	9
4.2.7	Input Data Constraints	11
4.2.8	Properties of a Correct Solution	11
5	Requirements	11
5.1	Functional Requirements	11
5.2	Nonfunctional Requirements	11
6	Likely Changes	12
7	Unlikely Changes	12
8	Traceability Matrices and Graphs	12

Revision History

Date	Version	Notes
2023-02-12	1.0	Notes
2023-03-01	1.01	Modified system context image, coordinate systems, and goal statements.
2023-04-29	1.02	Added requirements, instance models, data definitions
2023-06-05	1.03	Added Traceability Matrices
2023-06-18	1.04	Include the missing sections, modified the equations of DD, and IM. Added 4 new NFRs.

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with existing documentation for 3D Slicer program. The symbols are listed in alphabetical order.

symbol	type	description
m	\mathbb{N}	The first dimension of the segmentation volume.
m_i	\mathbb{N}	The first dimension of V_{in} .
m_o	\mathbb{N}	The first dimension of V_{out} .
n	\mathbb{N}	The second dimension of the segmentation volume.
n_i	\mathbb{N}	The second dimension of V_{in} .
n_o	\mathbb{N}	The second dimension of V_{out} .
p	\mathbb{N}	The third dimension of the segmentation volume.
p_i	\mathbb{N}	The third dimension of V_{in} .
p_o	\mathbb{N}	The third dimension of V_{out} .
slice	$\mathbb{R}^{m \times n}$	A slice is a 2 dimensional image view from the superior to inferior direction.
v	\mathbb{R}	A voxel reports the intensity of a single point on a grey-scale three-dimensional images.
$HIGH$	\mathbb{N}	A high intensity values means 1 on a scale of 0 and 1, or 255 on a scale of 0 to 255.
LOW	\mathbb{N}	A low intensity values means 0 on a scale of 0 and 1, or 0 on a scale of 0 to 255.
$Seed_a$	\mathbb{N}^3	The initial ascending aorta centre coordinates.
$Seed_d$	\mathbb{N}^3	The initial descending aorta centre coordinates.
$Start$	\mathbb{N}^3	A coordinate indicates the indexes of a starting voxel.
V	$\mathbb{R}^{m \times n \times p}$	Volume formed by a sequence of slice

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
AortaGeomRecon	Aorta Geometry Reconstructor
DD	Data Definition
DICOM	Digital Imaging and Communications in Medicine
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
T	Theoretical Model

2 Introduction

This document provides an overview of the Software Requirements Specification (SRS) for the AortaGeomRecon. AortaGeomRecon provides a semi-automatically aorta segmentation method, a highly customizable aorta segmentation module, and an interactive user interface to apply the segmentation workflow.

One of the existing methods involves the use ITK-Snap software and its segmentation module. First, the user needs to convert the DICOM data files (or any other file type) to VTK file. Then, the user can load the VTK file to ITK-Snap, and use its segmentation module to perform aorta segmentation. This segmentation method lets user initiate several voxels within the aorta volume and expand with a user's given size in each iteration. After the aorta volume has been filled by the "bubble", the user needs to cut the parts that are not within the aorta.

2.1 Purpose of Document

The main purpose of this document is to provide sufficient information to understand what AortaGeomRecon module does. The goals and theoretical models used in the AortaGeomRecon segmentation module implementation are provided, with an emphasis on explicitly identifying assumptions and unambiguous definitions.

2.2 Scope of Requirements

The scope of requirements only covers for the segmentation of the organ, more specifically the ascending aorta, the aortic curvature and the descending aorta. The requirements assume that the source of the data is accurate, and the user can manipulate (read, change dimensions) the data.

2.3 Characteristics of Intended Reader

The readers of the SRS should have taken the university level introduction to computational mathematic course, and be capable of understand the mathematical notation in the instance model section. The readers might have taken the university level introduction to software engineering course, have learned at least the waterfall software development model, and understands the purpose of the software specification requirement document, and other documents.

2.4 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by [Koothoor \(2013\)](#) and [Smith and Lai \(2005\)](#). The presentation follows the standard pattern of presenting goals, theories, definitions and assumptions. The goal

statements are refined to the theoretical models, and theoretical models to the instance models. For readers that would like a more bottom-up approach, they can start reading the instance models in Section 4.2.6 and trace back to find any additional information they require.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

Figure 1 shows the system context. A circle represents an external entity outside the software, the user in this case. A rectangle represents the software system itself. Arrows are used to show the data flow between the system and its environment.

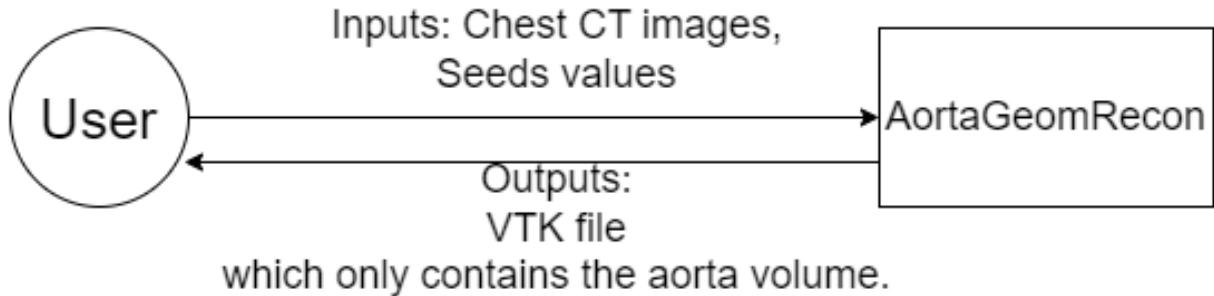


Figure 1: System Context

- User Responsibilities:
 - Provide the input data to the system
 - Ensure the input meets the necessary assumptions
 - Verify the result meets their requirements, otherwise repeat the process with a different seed values.
- AortaGeomRecon Responsibilities:
 - Provide DICOM data reader which can take a path to a folder containing DICOM files.

- Provide crop functionality to easily select a region of interest.
- Provide simple interactions to obtain and store the users' inputs. This includes a data probe to read voxel location which stored as a coordinate, and text inputs for real numbers.
- Provide visualization on the result data.

3.2 User Characteristics

The end user of AortaGeomRecon should have taken the university level anatomy introduction course, and be capable of finding the center of the descending aorta and the ascending aorta.

3.3 System Constraints

There are no system constraints of AortaGeomRecon.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

The main purpose of AortaGeomRecon is to semi-automatically segment a 3D aorta geometry from a chest CT scan.

4.1.1 Organ Segmentation

Organs are the body's recognizable structures (for example, the heart, lungs, liver, eyes, and stomach) that perform specific functions. Figure 2 below shows all the organs within a human body. The organ segmentation or the organ boundary segmentation is useful for orientation and identification of the regions of interests inside the organ during the diagnostic or treatment procedure. The aorta segmentation is important for aortic calcification quantification and to guide the segmentation of other central vessels. [Villa-Forte \(2022\)](#)

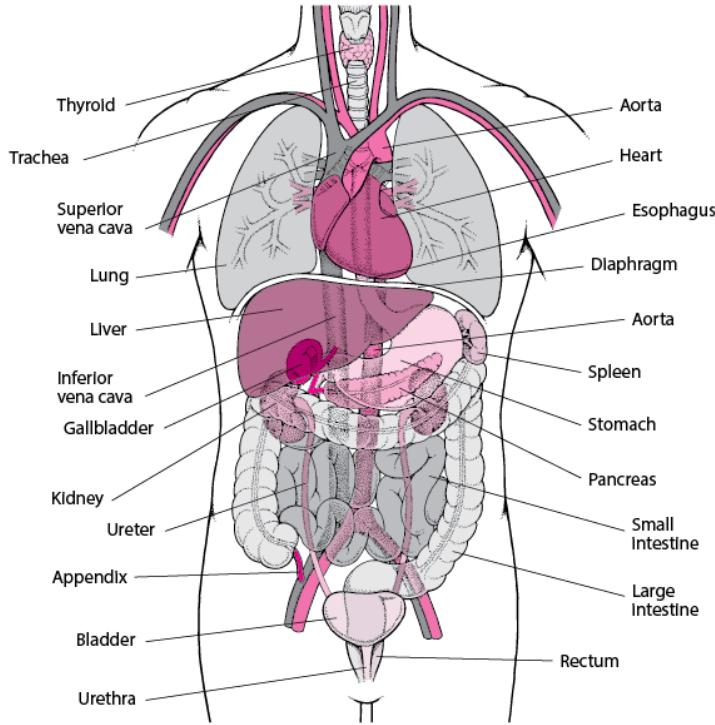


Figure 2: Human organs [Villa-Forte \(2022\)](#)

4.1.2 Coordinate Systems

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements. [Nejad \(2017\)](#)

While working with medical images, it is necessary to be familiar with the different coordinate systems of the medical literature and how data (voxels' orientation) is interpreted in different medical and nonmedical software. Each coordinate system uses one or more numbers (coordinates) to uniquely determine the position of a point (in the medical context, we refer to each point as a voxel). The purpose of this section is to introduce some coordinate systems related to the medical imaging. There are different coordinate systems to represent data. A knowledge of the following coordinate systems is needed to work with the medical images.

Cartesian Coordinate System A Cartesian coordinate system is a coordinate system that specifies each point uniquely in a 2D plane by a pair of numerical coordinates or in a 3D space by three numerical coordinates. We assume a right-hand Cartesian coordinate system throughout this document.

World Coordinate System World Coordinate System (WCS) is a Cartesian coordinate system that describes the physical coordinates associated with a model such as an MRI scanner or a patient. While each model has its own coordinate system, without a universal coordinate system such as WCS, they cannot interact with each other. For model interaction to be possible, their coordinate systems must be transformed into the WCS. Figure 3 shows the WCS corresponding space and axes.

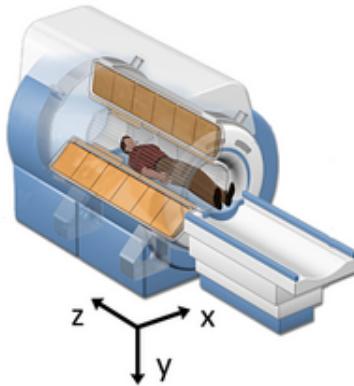


Figure 3: World Coordinate System Space and Axes [sli \(2014\)](#)

Anatomical Coordinate System Anatomical coordinate system, also known as patient coordinate system, is a right-handed 3D coordinate system that describes the standard anatomical position of a human using the following 3 orthogonal planes:

- Axial / Transverse plane: is a plane parallel to the ground that separates the body into head (superior) and tail (inferior) positions.
- Coronal / Frontal plane: is a plane perpendicular to the ground that divides the body into front (anterior) and back (posterior) positions.
- Sagittal / Median plane: is a plane that divides the body into right and left positions.

Figure 4 shows this coordinate system.

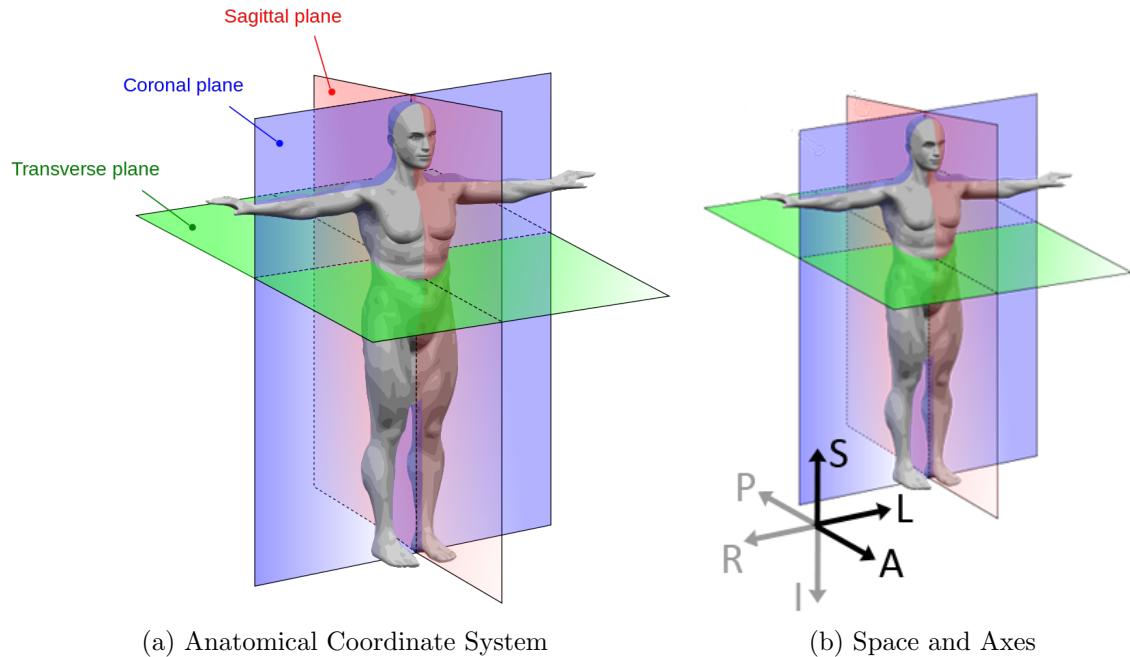


Figure 4: Anatomical Coordinate System Space and Axes [sli \(2014\)](#)

Medical applications follow an anatomical coordinate system to store voxels in sequences. Depending on how the data is stored, this coordinate system can be divided into different bases. The most common ones are:

- LPS Coordinate System:

The LPS coordinate system is used in DICOM images. In this system, voxels are ordered from left to right in a row, rows are ordered from posterior to anterior, and slices are stored from inferior to superior. LPS stands for Left-Posterior-Superior which indicates the directions that spatial axes are increasing.

- RAS Coordinate System:

The RAS coordinate system is the preferred basis for Neurological applications such as 3dfim+, and 3D Slicer. RAS stands for Right-Anterior-Superior is similar to LPS with the first two axes flipped.

Image Coordinate System To specify locations in an image we need to know to which coordinate system it is referenced. Different software may use different orders as their index convention.

Each of the coordinate systems mentioned above are used by different systems.

4.1.3 Physical System Description

We do not study the physical system for the images or how the data is actually generated.

4.1.4 Goal Statements

Given the DICOM image that includes patient's chest, the descending aorta center voxel coordinate, and the ascending aorta center voxel coordinate, the goal statements are:

GS1: Extract the three-dimensional segmentation of the aorta.

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 distinguishable 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].

4.2.2 Theoretical Models

There are no theoretical models used in this document.

4.2.3 General Definitions

There are no general definition used in this document.

4.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models.

Number	DD1
Label	Voxel
Symbol	$v : \mathbb{R}$
SI Units	-
Equation	-
Description	A slice (DD2) consists of $n \times n$ voxels. A real number is assigned to each voxel to reports the intensity on a grey-scale image.
Sources	Nejad (2017)
Ref. By	DD2
Number	DD2
Label	Image/Slice
Symbol	$slice : \mathbb{R}^{m \times n}$
SI Units	-
Equation	-
Description	A visual representation that is using only two spatial dimensions with a sequence of arrays where a voxel (DD1) represents the color or intensity. Each move in the transverse plane (Figure 4) is considered as one slice
Sources	Nejad (2017)
Ref. By	DD3
Number	DD3
Label	Volume
Symbol	$V : \mathbb{R}^{m \times n \times p}$
SI Units	-
Equation	-
Description	A three-dimensional image is a sequence of some images/slices (DD2).
Sources	-
Ref. By	IM1

4.2.5 Data Types

There are no additional data types used in this document.

4.2.6 Instance Models

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models. There are no theoretical models or general definitions used in this document.

The goals GS1 are solved by finding IM1 and perform IM2 on the aorta.

Number	IM1
Label	Region of interest
Inputs	$V_{\text{in}} : \mathbb{R}^{m_i \times n_i \times p_i}$, $Start : \mathbb{N}^3$, $m_o, n_o, p_o : \mathbb{N}$, with the following constraints: $\begin{aligned} 0 &\leq Start[0] < (m_i - 1) \\ 0 &\leq Start[1] < (n_i - 1) \\ 0 &\leq Start[2] < (p_i - 1) \\ 0 &< m_o \leq (m_i - Start[0]) \\ 0 &< n_o \leq (n_i - Start[1]) \\ 0 &< p_o \leq (p_i - Start[2]) \end{aligned}$
Output	$V_{\text{out}} : \mathbb{R}^{m_o \times n_o \times p_o}$ such that $\begin{aligned} \forall(i, j, k : \mathbb{N} \mid \\ i \in [Start[0]..Start[0] + m_o] \wedge \\ j \in [Start[1]..Start[1] + n_o] \wedge \\ k \in [Start[2]..Start[2] + p_o] : \\ V_{\text{out}}[i][j][k] = V_{\text{in}}[i][j][k]) \end{aligned}$
Description	The regions of interest is a subset (shaped like a box) of the 3D V_{out} . This subset contains the anatomical structure that the users wants to read, process or extract.
Sources	
Ref. By	IM2

Number	IM2
Label	Segmentation
Input	$V_{\text{in}} : \mathbb{R}^{m \times n \times p}$, $Seed_a : \mathbb{N}^3$, $Seed_d : \mathbb{N}^3$
Output	$V_{\text{out}} : \mathbb{R}^{m \times n \times p}$ such that $\forall (i, j, k : \mathbb{N} \mid i \in [0..m - 1] \wedge j \in [0..n - 1] \wedge k \in [0..p - 1] : (V_{\text{in}}[i, j, k] \in \text{structure} \implies V_{\text{out}}[i, j, k] = HIGH \mid V_{\text{in}}[i, j, k] \notin \text{structure} \implies V_{\text{out}}[i, j, k] = LOW))$ <p>The inputs $Seed_a$ and $Seed_d$ are used to determine whether a given element of V_{in} is in structure or not.</p>
Description	The process of extract an anatomical structure from the original 3D volume. The extracted anatomical structure is represented with high intensity pixel value. The rest of the image should have a lower intensity pixel value. The segmentation needs the region of interest from IM1 to process less noise data. A seed is what the algorithm needed as the inputs to perform segmentation, the type of seed is different among different algorithm. The seeds in this section are the centre coordinate of the descending aorta and the ascending aorta. The yellow dots shown in Figure 5 are the example of the seed.
Sources	
Ref. By	R3, LC1

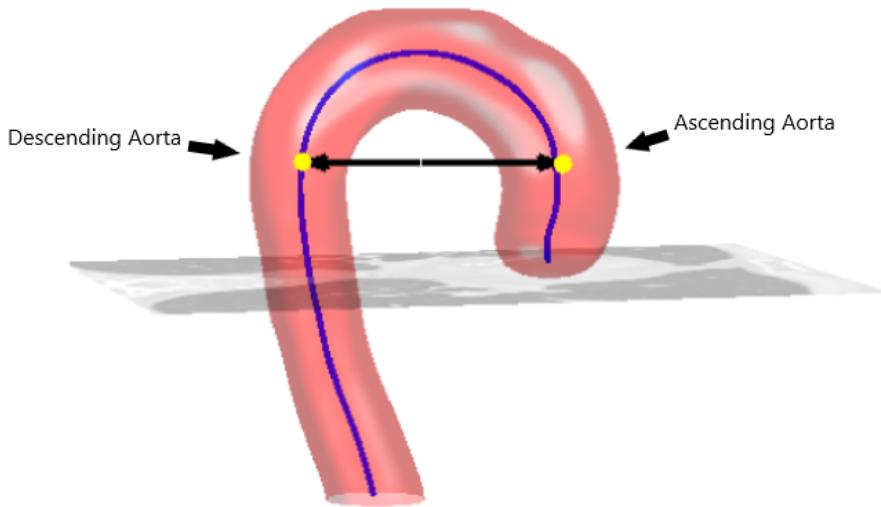


Figure 5: Aorta Seeds [Kurugol et al. \(2012\)](#)

4.2.7 Input Data Constraints

The only software constraint is the input volume data. It must be an acceptable file type to the system processing the data. For example, using ITK-Snap software to perform organ segmentation, the input data must be of VTK file.

4.2.8 Properties of a Correct Solution

A correct solution cannot be measured, but it can be confirmed by visually comparing the intersection of the extracted anatomical structure and the original volume.

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

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).

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).

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 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.

6 Likely Changes

LC1: IM2 There are various segmentation algorithms, each has a different procedure and inputs.

7 Unlikely Changes

UC1: IM1 The method to retrieve a region of interest from a volume is fixed.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an “X” may have to be modified as well. Table 2 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 3 shows the dependencies of instance models, requirements, and data constraints on each other. Table 4 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at

the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed.

	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

	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

	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

9 Reference

References

Coordinate systems, June 2014. URL https://www.slicer.org/wiki/Coordinate_systems.

Nirmitha Koothoor. A document drive approach to certifying scientific computing software. Master's thesis, McMaster University, Hamilton, Ontario, Canada, 2013. URL <http://hdl.handle.net/11375/13266>.

Sila Kurugol, Raul San Jose Estepar, James Ross, and George R. Washko. 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*, pages 2343–2346, 2012. doi: 10.1109/EMBC.2012.6346433.

Mojdeh Sayari Nejad. A case study in assurance case development for scientific software. Master's thesis, McMaster University, Hamilton, ON, Canada, July 2017. <http://hdl.handle.net/11375/23075>.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

Alexandra Villa-Forte. Tissues and organs. <https://www.merckmanuals.com/en-ca/home/fundamentals/the-human-body/tissues-and-organs>, 2022.

Appendix B

Module Guide for AortaGeomRecon

Module Guide for AortaGeomRecon

Jingyi Lin

July 5, 2023

1 Revision History

Date	Version	Notes
2022-10-18	1.0	First draft of Module Guide
2023-04-21	1.1	Second draft of Module Guide
2023-07-05	1.2	Added Traceability matrix between modules and source code. Added User Hierarchy between modules.

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

Symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
NFR	Non-Functional requirements
SC	Scientific Computing
SRS	Software Requirements Specification
AortaGeomRecon	Aorta Geometry Reconstruction
UC	Unlikely Change

Contents

1 Revision History	i
2 Reference Material	ii
2.1 Abbreviations and Acronyms	ii
3 Introduction	1
4 Anticipated and Unlikely Changes	2
4.1 Anticipated Changes	2
4.2 Unlikely Changes	2
5 Module Hierarchy	3
6 Connection Between Requirements and Design	3
7 Module Decomposition	3
7.1 Hardware Hiding Modules (M1)	4
7.2 Behaviour-Hiding Module	4
7.2.1 Input Format Module (M2)	4
7.2.2 Input Parameter Module (M3)	5
7.2.3 Control Module (M4)	5
7.2.4 Volume Visualization Module (M6)	5
7.2.5 Crop Module (M7)	6
7.2.6 Aorta Segmentation Module (M8)	6
7.2.7 Digital Enhancement Module (M11)	6
7.3 Software Decision Module	6
7.3.1 GUI Module (M5)	6
7.3.2 Image Processing Module (M9)	7
7.3.3 Multi-Dimensional Array Processing Module(M10)	7
8 Traceability Matrix	8
9 Use Hierarchy Between Modules	14
10 References	15

List of Tables

1 Module Hierarchy	4
2 Trace Between Requirements and Modules	8
3 Trace Between Anticipated Changes and Modules	8
4 Trace Between Modules and Code	13

List of Figures

1	Use hierarchy among modules	14
---	---------------------------------------	----

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the “secrets” that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules laid out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module’s data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers’ understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

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.

...

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: The input volume data's dimensionality is unlikely to change.

...

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

...

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

7 Module Decomposition

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.

Only the leaf modules in the hierarchy have to be implemented. If a dash (-) is shown, this means that the module is not a leaf and will not have to be implemented.

Level 1	Level 2	Level 3
Hardware-Hiding Module		
	Input Format Module	
	Input Parameter Module	
	Control Module	
Behaviour-Hiding Module	Volume Visualization Module	
	Crop Module	
	Aorta Segmentation Module	
	Digital Enhancement Module	
	GUI Module	
Software Decision Module	Image Processing Module	
	Multi-Dimensional Array Processing Module	

Table 1: Module Hierarchy

7.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviors.

Services: Includes programs that provide externally visible behavior of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Input Format Module (M2)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: AortaGeomRecon

Source: [AortaGeomReconDisplayModuleLogic module's process function](#)

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' attributes](#)

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

7.2.5 Crop Module (M7)

Secrets: The parameters, libraries to retrieve a region of interest from a volume.

Services: Import the necessary libraries, store the input parameter, and coordinate the uses of these data and libraries to retrieve a region of interest.

Implemented By: 3D Slicer

7.2.6 Aorta Segmentation Module (M8)

Secrets: The parameters, libraries to perform segmentation.

Services: Import the necessary libraries, store the input parameter, and coordinate the uses of these data and libraries to perform segmentation.

Implemented By: AortaGeomRecon

Source: [AortaSegmenter module](#)

7.2.7 Digital Enhancement Module (M11)

Secrets: The algorithm to enhance a digital image.

Services: Accepts a 2D image, output a 2D image with the same dimension as the input, improves visual quality of its greyscale image

Implemented By: AortaGeomRecon

Source: [AortaGeomReconDisplayModuleLogic's transform_image function](#)

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

7.3.1 GUI Module (M5)

Secrets: The necessary library/framework to build a GUI software.

Services: Provide Graphical User Interface for user to write/read inputs, and send commands to the program. It could include the rendering a windows, inputs, keyboard and mouse interaction with the GUI elements.

Implemented By: 3D Slicer

7.3.2 Image Processing Module (M9)

Secrets: The libraries and the APIs to perform image analysis, and image segmentation.

Services: Provides useful APIs such as ThresholdSegmentationLevelSetsImageFilter, LabelStatisticImageFilter to perform aorta segmentation.

Implemented By: SITK

7.3.3 Multi-Dimensional Array Processing Module(M10)

Secrets: The libraries and the APIs to perform element-wise mathematic operations on multidimensional array.

Services: Provides useful APIs such as calculating the max, min, average of multidimensional array. NumPy.where function provides the search in the multidimensional array functionality which will return any element's indexes if it satisfies the given condition.

Implemented By: NumPy

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes. The list of requirements can refer to the AortaGeomRecon's SRS. [Lin \(2023\)](#)

Req.	Modules
R1	M ₁ , M ₂ , M ₃ , M ₄
R2	M ₃ , M ₇
R3	M ₈ , M ₉ , M ₁₀
R4	M ₆
NFR1	M ₃ , M ₄ , M ₅
NFR2	M ₄ , M ₈ , M ₉ , M ₁₀
NFR3	M ₃ , M ₄ , M ₅ M ₆ , M ₇ , M ₈
NFR4	M ₇ , M ₈ , M ₉ , M ₁₀
NFR5	M ₃

Table 2: Trace Between Requirements and Modules

AC	Modules
AC ₁	M ₁
AC ₂	M ₂
AC ₃	M ₈
AC ₄	M ₃
AC ₅	M ₅
AC ₆	M ₇
AC ₇	M ₆
AC ₈	M ₄
AC ₉	M ₉ , M ₁₀

Table 3: Trace Between Anticipated Changes and Modules

Modules	Code
M1	-
M2 Input Format Module	<pre> # AortaGeomReconDisplayModule.py # https://github.com/smiths/aorta/blob/main/src/SlicerExtension/ # AortaGeometryReconstructor/AortaGeomReconDisplayModule/ # AortaGeomReconDisplayModule.py#L543-L554 stop_limit = self._parameterNode.GetParameter("stop_limit") threshold_coef = self._parameterNode.GetParameter("threshold_coef") rms_error = self._parameterNode.GetParameter("rms_error") no_ite = self._parameterNode.GetParameter("no_ite") curv_scaling = self._parameterNode.GetParameter("curv_scaling") prop_scaling = self._parameterNode.GetParameter("prop_scaling") kernel_size = self._parameterNode.GetParameter("kernel_size") # https://github.com/smiths/aorta/blob/main/src/SlicerExtension/ # AortaGeometryReconstructor/AortaGeomReconDisplayModule/ # AortaGeomReconDisplayModule.py#L824-L855 def process(self, des_seed, asc_seed, stop_limit, threshold_coef, kernel_size, rms_error, no_ite, curvature_scaling, propagation_scaling, debug): """Convert the parameters to the correct format and call begin_segmentation from AortaSegmenter. Returns: SITK::image: The processing image, or the segmentation label image """ des_seed = des_seed.split(",") asc_seed = asc_seed.split "," asc_seed = [int(i) for i in asc_seed] des_seed = [int(i) for i in des_seed] now = datetime.now() if not self._cropped_image: volume = slicer.mrmlScene.GetFirstNode("cropped", None, None, False) self.transform_image(volume) logging.info(f"{now} processing") segmenter = AortaSegmenter(cropped_image=self._cropped_image, des_seed=des_seed, asc_seed=asc_seed, stop_limit=float(stop_limit), threshold_coef=float(threshold_coef), kernel_size=int(float(kernel_size)), rms_error=float(rms_error), no_ite=int(no_ite.split(".")[0]), curvature_scaling=float(curvature_scaling), propagation_scaling=float(propagation_scaling), debug=debug) segmenter.begin_segmentation() now = datetime.now() logging.info(f"{now} Finished processing") return segmenter.processing_image </pre> <hr/>

M3 Input Parameter Module

```
# AortaSegmenter.py
# https://github.com/smiths/aorta/blob/main/src/SlicerExtension/
# AortaGeometryReconstructor/AortaGeomReconDisplayModule/
# AortaGeomReconDisplayModuleLib/AortaSegmenter.py#L53-L82
def __init__(self, cropped_image, des_seed, asc_seed, stop_limit=10,
             threshold_coef=3, kernel_size=6, rms_error=0.02, no_ite=600,
             curvature_scaling=2, propagation_scaling=0.5, debug=False):
    self._des_seed = des_seed
    self._des_prev_centre = des_seed[:2]
    self._asc_seed = asc_seed
    self._asc_prev_centre = asc_seed[:2]
    self._stop_limit = stop_limit
    self._threshold_coef = threshold_coef
    self._cropped_image = cropped_image
    self._kernel_size = kernel_size
    self._debug_mod = debug
    self._stats_filter = sitk.LabelStatisticsImageFilter()
    self._segment_filter = sitk.ThresholdSegmentationLevelSetImageFilter()
    self._segment_filter.SetMaximumRMSError(rms_error)
    self._segment_filter.SetNumberOfIterations(no_ite)
    self._segment_filter.SetCurvatureScaling(curvature_scaling)
    self._segment_filter.SetPropagationScaling(propagation_scaling)
    self._segment_filter.ReverseExpansionDirectionOn()
    self._k = 2
```

M4 Control Module

```
# https://github.com/smiths/aorta/blob/main/src/SlicerExtension/
# AortaGeometryReconstructor/AortaGeomReconDisplayModule/
# AortaGeomReconDisplayModule.py#L537-L583
def onApplyButton(self):
    """
    Go to next phase if on phase 1 crop aorta or perform segmentation if
    on phase 2 aorta segmentation.
    """

    with slicer.util.tryWithErrorDisplay(errorMessage, waitCursor=True):
        if self._parameterNode.GetParameter("phase") == "1":
            size = len(slicer.util.getNodes("*cropped*", useLists=True))
            if not size:
                logging.info("Cannot find cropped volume")
            else:
                self.showPhaseAS()
        elif self._parameterNode.GetParameter("phase") == "2":
            descAortaSeed = self._parameterNode.GetParameter(
                "descAortaSeed")
            ascAortaSeed = self._parameterNode.GetParameter(
                "ascAortaSeed")
            volume = sceneObj.GetFirstNode("cropped", None, None, False)
            self.logic.transform_image(volume)
            image = self.logic.process(
                descAortaSeed, ascAortaSeed, stop_limit,
                threshold_coef, kernel_size, rms_error, no_ite,
                curv_scaling, prop_scaling, self.ui.debugBox.checked
            )
            sitkUtils.PushVolumeToSlicer(
                image,
                name="Seg_th{}_k{}_c{}_p{}".format(
                    threshold_coef,
                    kernel_size,
                    curv_scaling,
                    prop_scaling),
                className="vtkMRMLScalarVolumeNode"
            )
```

M5 GUI Module

```
# AortaGeomReconDisplayModule.AortaGeomReconDisplayModuleWidget class
# https://github.com/smiths/aorta/blob/main/src/SlicerExtension/
#   AortaGeometryReconstructor/AortaGeomReconDisplayModule/
#     AortaGeomReconDisplayModule.py#L139-L245
class AortaGeomReconDisplayModuleWidget(ScriptedLoadableModuleWidget,
                                         VTKObservationMixin):
    def setup(self):
        ScriptedLoadableModuleWidget.setup(self)
        uiWidget = slicer.util.loadUI(self.resourcePath('UI/
            AortaGeomReconDisplayModule.ui')) # noqa: E501
        self.layout.addWidget(uiWidget)
        self.ui = slicer.util.childWidgetVariables(uiWidget)
        uiWidget.setMRMLScene(slicer.mrmlScene)
        self.logic = AortaGeomReconDisplayModuleLogic()
        scene = slicer.mrmlScene
        self.crosshairNode = slicer.util.getNode("Crosshair")
        self.crosshairNode.AddObserver(
            slicer.vtkMRMLCrosshairNode.CursorPositionModifiedEvent,
            self.onMouseMoved
        )

        self.addObserver(scene, scene.StartCloseEvent, self.
            onSceneStartClose)
        self.addObserver(scene, scene.EndCloseEvent, self.onSceneEndClose
        )
        self.ui.ascaAortaSeed.connect(
            "coordinatesChanged(double*)", self.
            updateParameterNodeFromGUI)
        self.ui.descAortaSeed.connect(
            "coordinatesChanged(double*)", self.
            updateParameterNodeFromGUI)
        self.ui.stopLimit.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)
        self.ui.kernelSize.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)
        self.ui.thresholdCoefficient.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)
        self.ui.rmsError.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)
        self.ui.noIteration.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)
        self.ui.curvatureScaling.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)
        self.ui.propagationScaling.connect(
            "valueChanged(double)", self.updateParameterNodeFromGUI)

        # Buttons
        self.ui.applyButton.connect('clicked(bool)', self.onApplyButton)
        self.ui.revertButton.connect('clicked(bool)', self.onRevertButton
        )
        self.ui.resetButton.connect('clicked(bool)', self.onResetButton)
        self.ui.skipButton.connect('clicked(bool)', self.onSkipButton)
        self.ui.getVTKButton.connect('clicked(bool)', self.onGetVTKButton
        )

        sliceDisplayNodes = slicer.util.getNodesByClass(
            "vtkMRMLSliceDisplayNode")
        for sliceDisplayNode in sliceDisplayNodes:
            sliceDisplayNode.SetIntersectingSlicesVisibility(1)

        sliceNodes = slicer.util.getNodesByClass('vtkMRMLSliceNode')
        for sliceNode in sliceNodes:
            sliceNode.Modified()
            self.initializeParameterNode()
```

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

9 Use Hierarchy Between Modules

In this section, the uses' hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

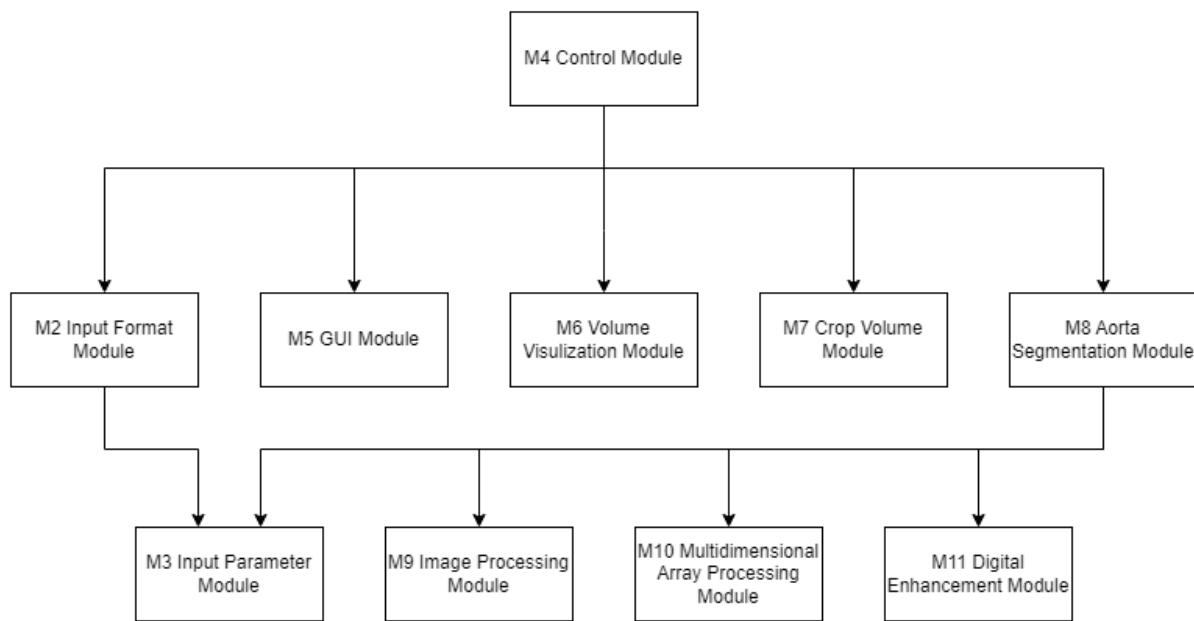


Figure 1: Use hierarchy among modules

10 References

References

Jingyi Lin. System requirements specification. <https://github.com/smiths/aorta/blob/main/docs/SRS/SRS.pdf>, 2023.

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.

Bibliography

- [1] Introduction: Slicer tutorials#segmentation. <https://www.slicer.org/wiki/Documentation/Nightly/Training#Segmentation>.
- [2] Astah system safety. <https://astah.net/products/astah-system-safety/>, May 2023.
- [3] BEARE, R., LOWEKAMP, B., AND YANIV, Z. Image segmentation, registration and characterization in r with simpleitk. *Journal of Statistical Software* 86, 8 (2018), 1–35.
- [4] BLANCHETTE, S. *Assurance Cases for Design Analysis of Complex System of Systems Software*.
- [5] GERIG, G. Snap tutorial and user's manual. <http://www.itksnap.org/docs/fullmanual.php>. [Online; accessed 2023; Section 6. Automatic Segmentation using Region Competition Snakes].
- [6] GIBAUD, B. The dicom standard: A brief overview. In *Molecular Imaging: Computer Reconstruction and Practice* (Dordrecht, 2008), Y. Lemoigne and A. Caner, Eds., Springer Netherlands, pp. 229–238.

- [7] GITHUB. Github actions documentation. <https://docs.github.com/en/actions/using-workflows/about-workflows>, 2023.
- [8] HARRIS, C. R., MILLMAN, K. J., VAN DER WALT, S. J., GOMMERS, R., VIRTANEN, P., COURNAPEAU, D., WIESER, E., TAYLOR, J., BERG, S., SMITH, N. J., KERN, R., PICUS, M., HOYER, S., VAN KERKWIJK, M. H., BRETT, M., HALDANE, A., DEL RÍO, J. F., WIEBE, M., PETERSON, P., GÉRARD-MARCHANT, P., SHEPPARD, K., REDDY, T., WECKESSER, W., ABBASI, H., GOHLKE, C., AND OLIPHANT, T. E. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362.
- [9] KELLY, T., AND WEAVER, R. The goal structuring notation—a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases* (2004), vol. 6, Citeseer.
- [10] KIKINIS, R., PIEPER, S. D., AND VOSBURGH, K. G. *3D Slicer: A Platform for Subject-Specific Image Analysis, Visualization, and Clinical Support*. Springer New York, New York, NY, 2014, pp. 277–289.
- [11] KLUYVER, T., RAGAN-KELLEY, B., PÉREZ, F., GRANGER, B., BUSSONNIER, M., FREDERIC, J., KELLEY, K., HAMRICK, J., GROUT, J., CORLAY, S., IVANOV, P., AVILA, D., ABDALLA, S., AND WILLING, C. Jupyter notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (2016), F. Loizides and B. Schmidt, Eds., IOS Press, pp. 87 – 90.
- [12] 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.

- [13] LIN, J. Design document. <https://joviel25.github.io/AortaGR-design-document/index.html>, 2023.
- [14] LOWEKAMP, B., CHEN, D., IBANEZ, L., AND BLEZEK, D. The design of simpleitk. *Frontiers in Neuroinformatics* 7 (2013).
- [15] MA, J., ZHANG, Y., GU, S., ZHU, C., GE, C., ZHANG, Y., AN, X., WANG, C., WANG, Q., LIU, X., CAO, S., ZHANG, Q., LIU, S., WANG, Y., LI, Y., HE, J., AND YANG, X. Abdomenct-1k: Is abdominal organ segmentation a solved problem? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 6695–6714.
- [16] RAYBAUT, P. Spyder-documentation. Available online at: pythonhosted.org (2009).
- [17] SMITH, S., NEJAD, M. S., AND WASSYNG, A. Building confidence in scientific computing software via assurance cases. *CoRR abs/1912.13308* (2019).
- [18] VENTOLA, C. Mobile devices and apps for health care professionals: Uses and benefits. *P & T : a peer-reviewed journal for formulary management* 39 (05 2014), 356–64.
- [19] WEINSTOCK, C. Assurance cases and confidence. <https://insights.sei.cmu.edu/blog/assurance-cases-and-confidence/>, Aug 2013.
- [20] WIKIPEDIA. Lint (software). [https://en.wikipedia.org/wiki/Lint_\(software\)](https://en.wikipedia.org/wiki/Lint_(software)), 2023.

- [21] YUSHKEVICH, P. A., PIVEN, J., CODY HAZLETT, H., GIMPEL SMITH, R., HO, S., GEE, J. C., AND GERIG, G. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage* 31, 3 (2006), 1116–1128.