2008

# Extraction of quantitative measures of the aorta from four dimensional segmented MR data

Matthew T Thomas
*University of Iowa*

# EXTRACTION OF QUANTITATIVE MEASURES OF THE AORTA FROM FOUR DIMENSIONAL SEGMENTED MR DATA

by

Matthew T. Thomas

A thesis submitted in partial fulfillment of the
requirements for the Master of Science degree
in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

May 2008

Thesis Supervisor: Professor Milan Sonka

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

---

MASTER'S THESIS

---

This is to certify that the Master's thesis of

Matthew T. Thomas

has been approved by the Examining Committee for the thesis requirement for the Master of Science degree in Electrical and Computer Engineering at the May 2008 graduation.

Thesis Committee: _____
Milan Sonka, Thesis Supervisor

_____
Andreas Wahle

_____
Thomas Scholz

To Howard Baich and Theodore J. Thomas

We keep moving forward, opening up new doors and doing new things, because we're curious...and curiosity keeps leading us down new paths. Walt Disney

## ACKNOWLEDGEMENTS

I would like to acknowledge the following people in my lab, Milan Sonka, Andreas Wahle, and Thomas Scholz for advising me during the course of this project. I would also like to thank Honghai Zhang, Fei Zhao, Senthil Premraj, Mona (Haeker) Garvin, Richard Downe and Nick Walker for their time and expert knowledge; Ryan Johnson for verifying my results; Natalie Van Waning for maintaining the database of studies and the National Institute of Health for funding this project. An appreciation goes to all the professor that I have interacted with during my years as a student. I would like to say thank you Courtney Connor, Ted, Jan and Jeremy Thomas for their love, support, time, effort and encouragement during the more challenging times. And finally for their support Adam Thomas, the CCF church, and Jason Lassner.

## TABLE OF CONTENTS

## LIST OF TABLES

Table

## LIST OF FIGURES

Figure

# CHAPTER 1
# INTRODUCTION

The purpose of this master's work is to provide a method to obtain measurable information from graphed searched magnetic resonance images (MRI). By providing this information one's intention is to show that through the use of both three dimensional and four dimensional measurements, one can identify the differences in patients with Marfan Syndrome and those who do not have the disease. Marfan Syndrome is a connective tissue disorder that can affect the aorta. The aorta is the largest blood vessel in the human body, that supplies blood throughout the body. If Marfan Syndrome is not diagnosis the aorta can tear; if this tear is not fixed immediately death will occur. Prior to obtaining this information one must show that the extraction process can accurately take a given dataset and produce a set of contour points that represents a given segmented volume. From these points one would like to produce and store these measurements. Prior to storing and creating these measures one has to show that these points accurately measure a given data set. In certain cases, either due to local inaccuracies in the segmentation or misaligned planes in the extraction process a verification and editing tool needs to be created to fix these errors.

## 1.1 Aims of the Research Work

1. Extract a set of contour points that accurately represent a given aortic volume
2. Create a set of measures based on the above contour points
3. Verify that the given non-modal indexes are correct
4. Provide a program to modify the extracted contour points, if needed

## 1.2 Multiple Languages

In order to achieve these aims multiple programming languages were utilized. To obtain the data, Perl Scripts were created to search a location on a server and then

download data from this server. Next the C++ programming language was used to load, process and save the contour points. In conjunction with the standard C++ library, the Visual Toolkit (VTK) and the Insight Toolkit (ITK) were used. After the points were extracted they were processed using Perl Scripts and from these scripts a previously developed program was executed to create the measures. After the measures were made they were stored in a Structured Query Language (SQL) database using the Perl Database Interface (DBI) and SQLite as database interface.

# CHAPTER 2
# MEDICAL BACKGROUND

## 2.1 Anatomy of the Aorta

The aorta is a large blood vessel that begins at the aortic valve located at the end of the left ventricular outflow track. On average this vessel has a diameter of three centimeters when it leaves the ventricle and decreases to an average size of one and three-fourth centimeters at the junction of the left and right iliac arteries located in the pelvic region. For this master's work the region of interest goes from the aortic value to a location near the diaphragm. This area of study includes the aortic sinus, the ascending aorta, the arch of the aorta and a small portion of the descending aorta 2.1 [3]. Of particular interest to this work is the aortic root, aortic sinus, and the sino-tubular junction. The aortic root is located at the junction of the aorta and aortic valve. The aortic sinus is an area of dilation that is located between the aortic root and sino-tubular junction. This location is a common area of aortic dilation in Marfan Syndrome. The sino-tubular junction is located at the junction of the the sinus and the ascending aorta.

## 2.2 Introduction to Marfan Syndrome

Marfan syndrome (MFS) is a systemic disorder of the connective tissue that affects multiple body systems. Connective tissue is tissue in the body that strengthens structures. For example, MFS patients typically have long arms, legs, and fingers which gives a lanky appearance. MFS is caused by a mutation in gene FBN1 which is located on the chromosome 15. This gene is responsible for the encoding of the protein fibrillin-1 [2]. Fibrillin helps build elastic fibers which are important for the function of flexible structures such as the blood vessels [6]. MFS affects about in 1 in 5000 people. The defect caused by MFS is inherited from the patients parents, but this does not mean that the patient will have the symptoms related to the disorder.[2].

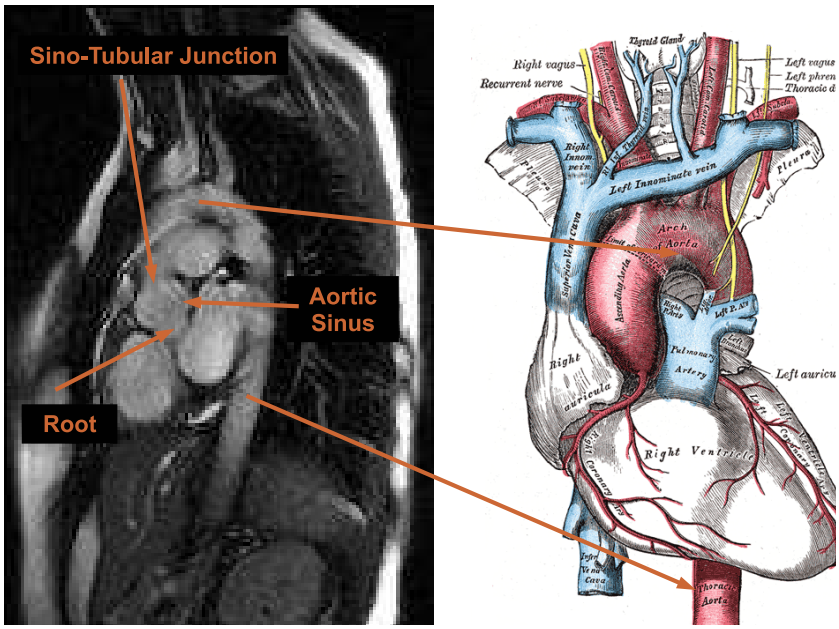



Figure 2.1: Anatomy of the Aorta [3]

### 2.3 Effects on the Aorta

One of the aims of this master's work was to study the effects of MFS on the aorta. One of the major problems of MFS it that it causes the aorta to stretch which can then lead to an aortic dissection. Figure 2.2 demonstrates the different types of aortic dissection based on DaBakey and Stanford method [4]. Aortic dissection occurs when the wall of the aorta begins to tear, creating two areas for blood to flow. The first area is the normal lumen and allows blood to continue through the aorta. The second area is abnormal and allows for blood to pool is this area thus creating an enlargement of the aorta, or an aneurysm. Figure 2.3 shows the different types of aneurysms that can occur near the root. If not treated the aneurysm can cause the aorta to rupture. The death rate is one percent per hour in the first forty-eight hours or approximately fifty percent. Less then fifty percent of patients with a raptured aorta will survive [8].

MFS can also cause mitral valve prolapse which is when this valve does not open

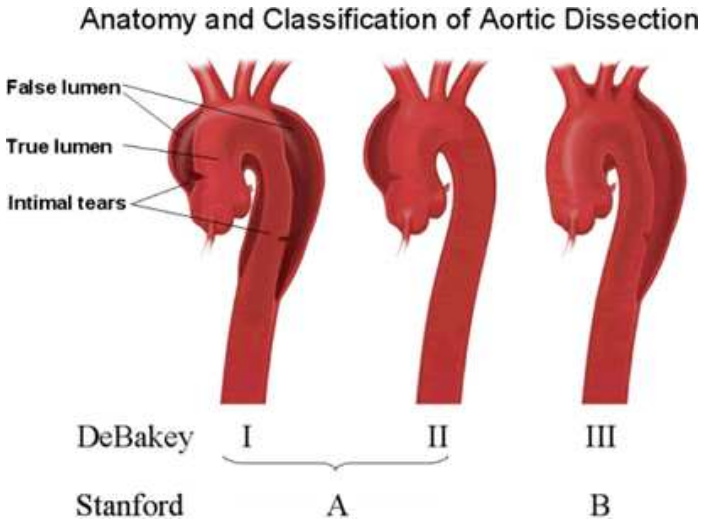



Figure 2.2: Aortic Dissection [4]

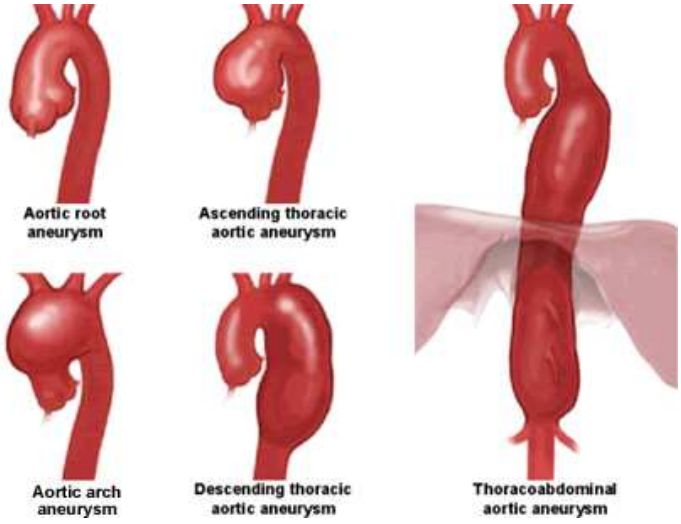



Figure 2.3: Aortic Aneurysms [4]

or close properly and sometimes leads to regurgitation. Regurgitation occurs when blood is allowed to flow backwards into the left atium. One reason that the mitral valve fails to close or open is due to the enlarged aorta.

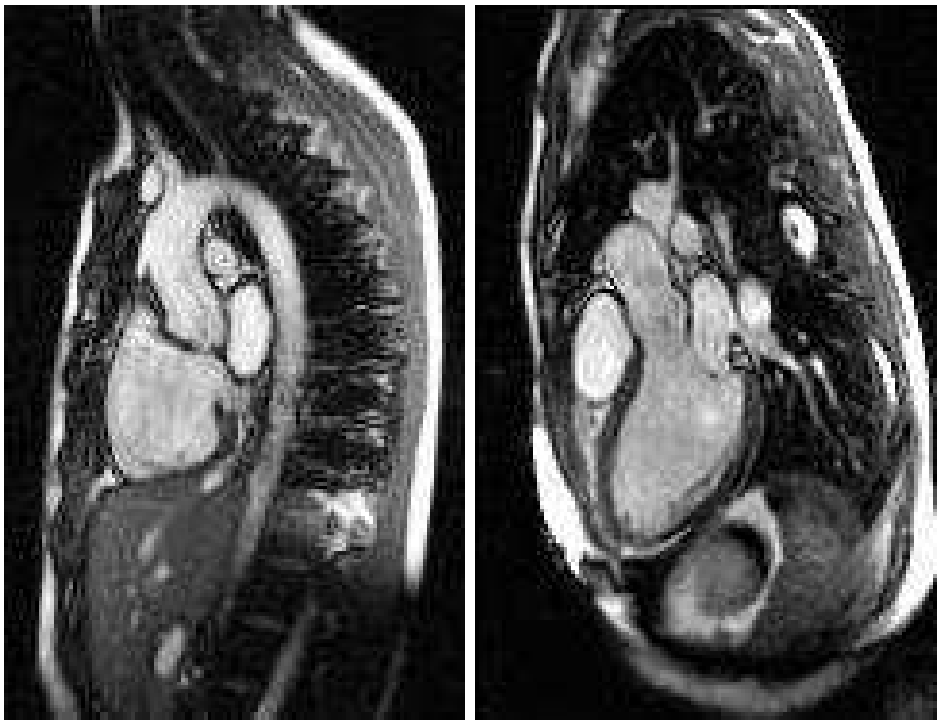
### 2.4 Diagnosis and Treatment

In order to prevent the above complications of MFS, one needs to have regular echocardiograms (an ultrasound of the aorta) to diagnose the current progression of the syndrome. Using an echocardiogram, doctors can approximately measure the current size of the aorta at distinct locations. If the aorta has reached a certain size, surgery is needed to replace the dilated part of the aorta (typically the root) or the aortic valve or both. Although the current method of diagnosis is the gold standard, it only uses one two dimensional slice to determine whether surgery is warranted. The purpose of this work is to provide a method that uses both three dimensional and four dimensional measures to chart the progression of MFS.

# CHAPTER 3
# PRIOR WORK

## 3.1 Segmentation

Prior to performing the extraction and calculation of the measures a model of the aorta needs to be generated. These volume were created using the algorithm found in [10], a brief summary of this method follows. To start the process a Left Ventricle Outflow Track (LVOT), figure 3.1(a) and Candy Cane view, figure 3.1(b) are acquired from either a General Electric (GE) or Siemens MR scanner.



(a) Candy Cane View (b) LVOT View

Figure 3.1: Views used in the creation of the aorta model

In order to benefit from the information of both views are merged together and recreated using a B-Spline interpolation and geometric information acquired a scan time. The average isotropic voxel size for these sets are 1.5 $mm^4$ for GE scanners

and 1.9 $mm^3$ for Siemens scanners. After merging a graph searched is applied to the merged LVOT data set to produce a better approximation of the aorta from the root to sino-tubular junction. This result was then mapped back to the CC data set and then registered across all sixteen phases. Using this initial surface estimation a level set algorithm is applied to the CC data set and an approximate surface representing the entire aorta from the root to the diaphragm. A centerline is then generated from this surface and used to re-sample the CC data set producing a volume perpendicular to the centerline. A four dimensional graph search is then performed on this data set producing a model of the given aorta. This model is then saved in a three dimensional data set and used during the extraction process.

# CHAPTER 4
# DATA EXTRACTION PIPELINE

## 4.1 Goals of Pipeline

The primary goal of the extraction pipeline was to create surface points located on the segmented volume. The secondary goal was to create a set of transforms that when applied to a given volume produced a volume perpendicular to the centerline. This perpendicular volume allows for validation of both this method and the segmentation results. These two goals are accomplished using the Insight Toolkit and Kitware's Visualization Toolkit. A general overview of the pipeline is given in figure 4.1.

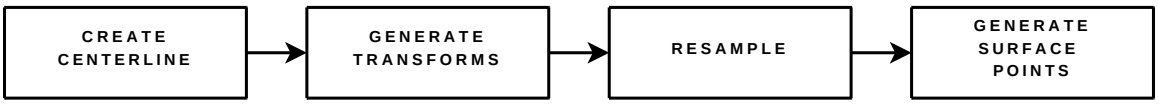



Figure 4.1: Extraction Pipeline

## 4.2 Input into the Pipeline

In order to perform an analysis on a given segmented data set, the user needs to identify a root and diaphragm point. These points was used to start and end the centerline extraction. It was also used to prevent the centerline from going beyond the segmentation results. Another use of these points was to adjust the direction of the centerline so it remained in the middle of the vessel. Figure 4.2(a) illustrates what happened when the diaphragm point was not set. If this centerline was used it would generate planes shown as green lines in the figure. These planes produce inaccurate contours at those locations. Figure 4.2(b) demonstrated the output when a the diaphragm point was set.

(a) No diaphragm point (b) Diaphragm point

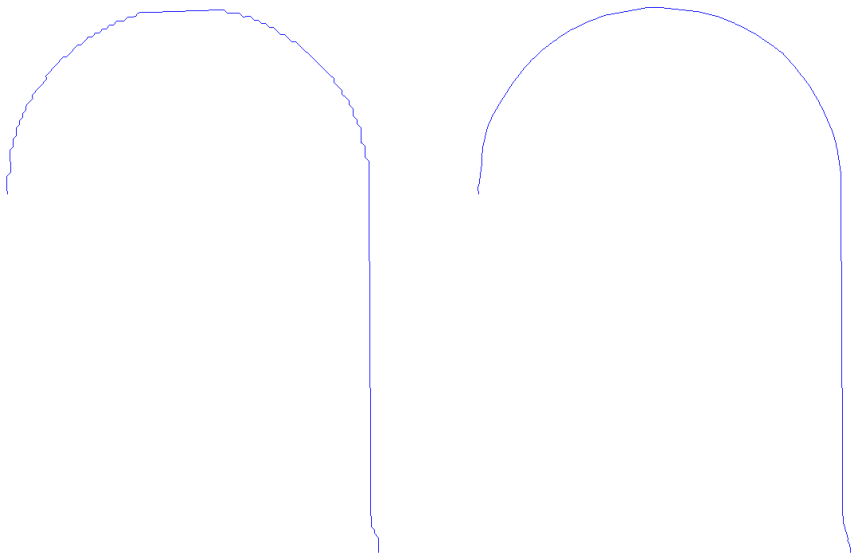
Figure 4.2: Inaccurate centerline

The final set of input parameters was used to define the dimensions of the perpendicular volume. The first parameter that was needed was the re-sampling window size or the dimension of the perpendicular images in the xy-plane. The default value for this parameter was fifty voxels by fifty voxels. This value typically captures the entire aorta in cross section at a given centerline location. The user also specidied the number of cross sectional slices to create in the re-sampled volume and the number of control points to generate on each cross sectional slice. By increasing these values one could generate a denser representation of the surface. However if the number of cross sectional slices is too large intersection will occur between slices, which will create artifacts in the final results. If the number of control points was too large then local noise will appear in the results, but given the correct number of points this noise can be removed using a spline. The default value for these parameters was one hundred slices and one hundred control points.

### 4.3 Centerline Generation

The centerline was created using the algorithm found in 4.3(a) and an example is seen in figure 4.3. As it can be seen the generated centerline had a "'stair-step"' effect. If no smoothing filter was applied the probability of incorrect transformed is

increased. An example of the smooth centerline is shown in figure 4.3(b). In order to calculate a smooth centerline a window size was chosen to average a centerline point. Then the gradient in the x-direction, y-direction and z-direction were calculated from the current centerline point to the end of the window. An average centerline point was calculated in the direction perpendicular to the gradient with the greatest magnitude. When the average was calculated in a perpendicular direction to the greatest gradient one did not loose the directional information of the centerline [10]. The generated centerline was then up sampled by a factor of two using the midpoint formula. The up sampled version of the centerline guarantees that number of centerline points is greater then the number of slices specified by the user.



(a) Un-smooth centerline (b) Smoothed centerline

Figure 4.3: Centerline

### 4.4 Rolling Sphere Algorithm

In order to determine an approximate normal at a given centerline point an iterative rolling spheres algorithm was used. The first step in the rolling spheres algorithm was to choose an initial radius for each sphere at a given centerline point (CP). Next an initial normal direction (N) is calculated using CP to the next point on the centerline. Starting from CP to the previous point(A), a distance is calculated, normalized, and added to N, figure 4.4. This process was repeated using the previous point A and A's previous point(B). This process was continued to the end of the centerline or the euclidean distance from the current B to CP was greater than the radius of the sphere. This process was known as backwards averaging [9].

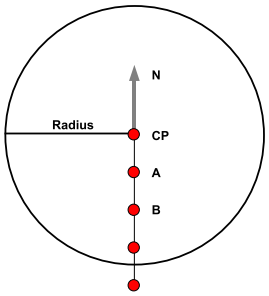



Figure 4.4: Backward Averaging

After the backwards averaging is utilized, a forward average was applied. This was started two points forward from the centerline point(CP), an euclidean distance is calculated from this initial point(A) to its previous point (B) it is then normalized, and added to N. This process was repeated using A as the new B and A's next point(C) as the new A, figure 4.5. This process terminated if it reached the end of the centerline or the euclidean distance from the current C to CP was greater then the radius of sphere.

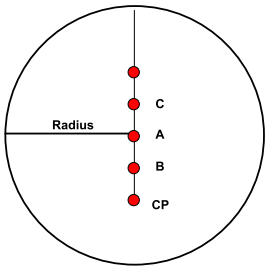
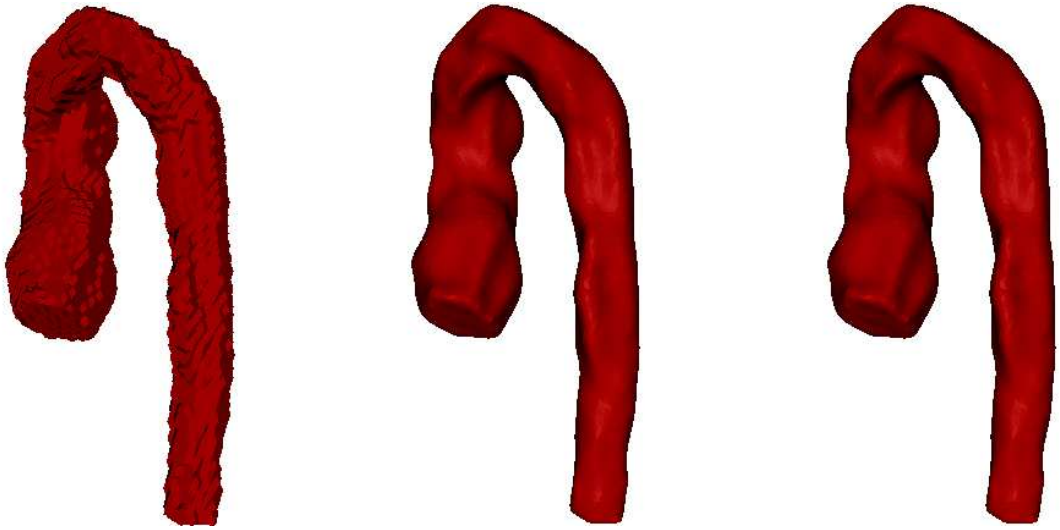



Figure 4.5: Forward Averaging

## 4.5 Radius Estimation

The above rolling sphere algorithm, was applied to the centerline; this gave an initial normal direction for each centerline point. The initial radius for each sphere at a given centerline point was three voxels. A *vtkImageMarchingCubes* filter [5] was applied to the segmented binary volume, thus outputting a single iso-surface 4.6(a). This surface was inputed into a *vtkWindowedSincPolyDataFilter* filter [5] to remove any noise from inputed the volume 4.6(b). The results of this filter were inputted into a *vtkDecimatePro* filter to reduce the number of triangles produced by the marching cubes algorithm 4.6(c). This decimation did not effect the topology of the iso-surface but allows for an decrease in compution time.



(a) Original Iso-Surface (b) After smoothing (c) After decimation

Figure 4.6: Steps in the creation of iso-surface

Using a given centerline point and its respective normal a *vtkPlane* source is created and inputted into a *vtkCutter* filter [5] 4.7(a). This filter was then applied to the isosurface, and its output is connected to a *vtkStripper* filter [5] thus producing one or more connected isolines 4.7(b). The above images are similar because the filters worked on the structure and ordering of the points and not the lines. By manipulating the points, one prevented the distortion of the surface and thus preserved information needed for the analysis. These lines were connected to a *vtkSplineFilter* filter [5] outputting a spline containing N equally distant points representing a contour at the given centerline point 4.7(c). In certain locations the created results contained two splines due to a infinite plane being used in the *vtkCutter* filter. In order to correct for this, an euclidean distance was calculated from each point to the current centerline point. These distances were sorted in order from smallest to largest and N/2 closest points were kept. In order to prevent distortion of the contours the original location on the spline were stored prior to sorting the distances. After the removal of N/2 points

the kept points were reordered according to their original location 4.7(d). Using the previously calculated distances the maximum distances were found and used as the new radius in rolling sphere iteration at the current centerline point. This process was repeated for each centerline point thus producing a set of contour points.

It was likely that this initial set of contours would have some intersection with each other. This error is caused by the use of the same radius at each centerline point during the rolling sphere algorithm. In order to correct for this error, the estimated radius found for a given centerline point in above method was used in a second iteration of the rolling sphere algorithm. By using a unique radius for each centerline point one could limit the effect that an outlying normal had on the current centerline point.

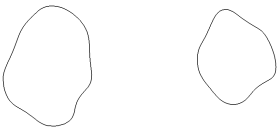
The process of calculating normals, slicing the surface, and estimating new set of the radii was performed three times. The first iteration determined a unique set of radii, the second and third iteration created the best estimation of each centerline point's normal. If the number of iteration was larger then three, the radii typically fluctuated between two values, meaning that a better estimation of the radii was between these points. However to decrease computation time the set of radii found during the third iteration was used to calculate a set of transforms.

## 4.6 Calculation of the orthogonal vectors of a plane perpendicular to the centerline

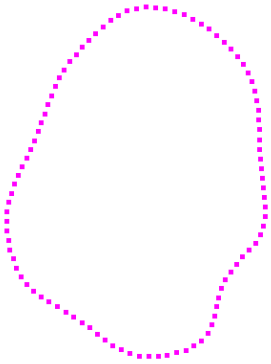
Using an initial direction of (0,0,1) in the original candy cane space or (x,y,z) space and the direction of the first normal, a cross product was computed, that produced a perpendicular vector $\vec{U}$. A second cross product was performed on $\vec{U}$ and the first normal, which produced a second perpendicular vector $\vec{V}$. Vectors $\vec{U}$ and $\vec{V}$ orientated the first slice in the new (u,v,w) space such that the normal was (0,0,1). Using the current $\vec{U}$, $\vec{V}$, and the normal at the next centerline point, a plane perpendicular to the centerline was defined as follows. By computing a cross product with the current

(a) After cutter

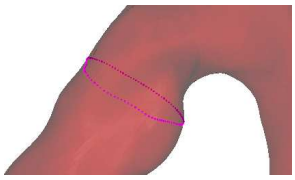
(b) After stripper



(c) After spline



(d) Contour Points



(e) Contour Points on the Surface

Figure 4.7: Radius Estimation

$\vec{U}$, from the above process, and the current normal at the next centerline location a third perpendicular vector $\vec{V}$ was created. A second cross product was performed on the current normal and the new $\vec{V}$; this result was stored in vector $\vec{U}$. These two vectors and the current centerline point were stored in a file for later use. This process was repeated for each centerline point. When completed, each centerline point had a $\vec{U}$ and $\vec{V}$ that defined a plane perpendicular to that point.

### 4.7 Re-sampling

A signed distance map of the input binary volume is calculated using the ITK *SignedMaurerDistanceMapImageFilter* [7]. Using the transforms calculated in section 4.6 and equations 4.1 - 4.3 the distance map is trilinearly interpolated such that the output volume contains slices perpendicular to the centerline.

$$O_x = C_x - U_x(x - \frac{W_x}{2}) - V_x(y - \frac{W_y}{2}) \tag{4.1}$$

$$O_y = C_y - U_y(x - \frac{W_x}{2}) - V_y(y - \frac{W_y}{2}) \tag{4.2}$$

$$O_z = C_z - U_z(x - \frac{W_x}{2}) - V_z(y - \frac{W_y}{2}) \tag{4.3}$$

Figure 4.8: Transform Equation

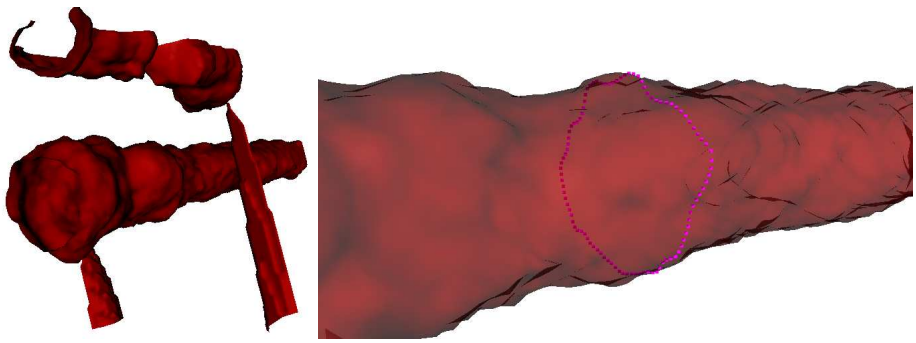
Equations equations 4.1 - 4.3 can be represented in vector form as given in equation 4.4. In equation 4.4 $\vec{O}$ represents a voxel in the distance map, $\vec{U}$ and $\vec{V}$ are the vectors calculated in section 4.6,$\vec{C}$ is the current centerline point and (x,y) is the pixel of interested in the straighten volume. $W_x$ and $W_y$ are the dimensions in the xy-plane. The z dimension of the straighten volume is equal to the number of slices determined by the user.

$$\vec{O} = \vec{C} - \vec{U}(x - \frac{W_x}{2}) - \vec{V}(y - \frac{W_y}{2}) \tag{4.4}$$

Figure 4.9: Transform Equations in Vector Notation

### 4.8 Contour Creation

An iso-surface was created using the *vtkMarchingCubes* filter [5]. The iso-surface was then cut using *vtkCutter* and a *vtkPlane* source [5], such that the plane had a normal of (0,0,1) and an origin at the center of the xy-plane. For every z-slice in the iso-surface a plane was created, splined and stored for later use. Figure 4.10(a) shows the iso-surface created from the *vtkMarchingCubes* filter. The extra surfaces located above and below the vessel were caused by the use of larger then needed sample window, $W_x$ and $W_y$ in equation 4.4. However these extra surfaces did not affect the generated contours as seen in 4.10(b).



(a) Straighten Iso-Surface (b) Contour Points

Figure 4.10: Contour Creation

# CHAPTER 5
# DATA ANALYSIS PIPELINE

## 5.1 Goals of the Pipeline

Given a set of contours from a given binary volume, one created a set of measures. Some examples of these measures are diameter, distance, cross section area, and centroid displacement. After reorganizing the contours with the Perl Scripting language the non-modal indexes were calculated using the Coronary Vessel and Plaque Morphology Analysis with 3-D Visualization Tool (CVPA3D)[1]. The output of this program was modified again with Perl Scripts and stored in an Standard Query Language (SQL) database for later use.

## 5.2 Reorder of Contour Points

In order to create the non-modal indexes, the generated contour points were translated back to the original Candy Cane (CC) space. Through the use of equation 4.4 and a Perl Script these points were mapped from the perpendicular space back to CC space. When the points were outputted from the above process they were unordered in the sense the point P on slice N is not located near point P on N+1 or N-1, thus producing twisted or misaligned triangular meshed surface. These twisted surfaces caused inaccurate results when entered into the CVPA3D. To correct this error, slice zero's contours were stored in the final output array and point zero on slice zero was used as a reference to straighten the surface. An euclidean distance was measured from point one on slice one to the zero point on the previous slice. This distance was then compared to current minimum and if it is was found to be less then the current minimum, the current minimum was set equal to this distance and location of this point was stored for later use. This process was repeated for each point on the current slice. Starting from the minimum point to the last point each point was stored in the final array. This process was repeated starting from the first point up to, but not including, the minimum location. This processes was then repeated on

each slice thus producing non-twisted contours.

### 5.3 Construction and Loading of Database

By implementing a database of the given output of CVPA3D tool one would be able to access an endless amount of data in a matter of seconds. Not only would the data be quickly accessible, but with a few Perl scripts, the data could be modified and stored in an Microsoft Excel file. The first step in the creation of the database was the design of the tables. A database table describes how the data should be stored. Figure 5.1 shows the Entity Relationship(ER) diagram of the database. The ER diagram is a convenient way of showing each table or entity (shown by rectangles), attributes (shown by circles) and how the tables relate to each other (shown as diamonds). The double rectangle around the data table indicates that in order to find a certain value is this table one needs to use both the table and the studies table, this is a called a weak entity set.

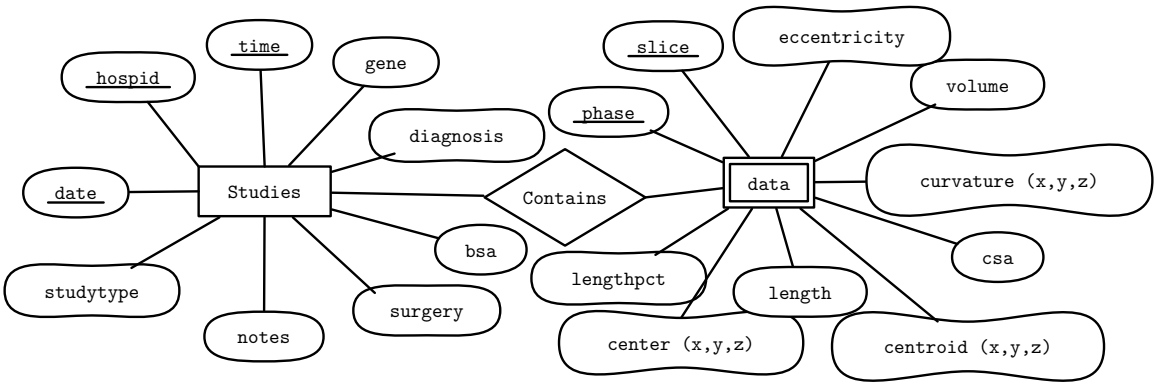



Figure 5.1: Entity Relationship Figure

Using these two tables, further tables can be created using pure SQL syntax, or through a Perl script. As an example, of one such table that was the maximum and minimum cross sectional area(CSA) across phases at a given centerline point. To

construct this table a query was made to obtain all the hospids, time, and date. Using these results, a query was made on the data to obtain a maximum and minimum CSA across all sixteen phases for a given slice. To account for any movement in a slice across phases an average and standard deviation was calculated for each slice located on the centerline. All the results are stored in another table in the database for later use.

### 5.4 Example Perl Script

Figure 5.1 shows a example of a Perl Script to load data into the database. The first step in loading the data was to create the SQL query statement to be executed at a latter time (line 7). The next step was to read a result file, modify one line at time so it can be stored in the database and then execute the SQL query (lines 19-33). In order to increase the speed of insertion but guarantee that the results are being stored, a commit to the database occurs after one thousand lines (line 35).

### 5.5 Root and Diaphragm Adjustment

Due to local noise in the segmentation or a faulty centerline generation an outlying CSA measure could occur at the root or diaphragm slice. To fix this error a linear interpolated value was calculated using the two nearest neighbors on the centerline. If the absolute percent difference between the interpolated value and true measure was above a percentage difference threshold the interpolated value was chosen as the new CSA measure. In order to determine the optimal threshold, an experiment was done using a set of thresholds ranging from five to fifty percentage. Using the above method, the threshold was applied to the current set of data and the number of root and diaphragm points that were changed as well as the total number of points was stored. Figure 5.2 shows that as the threshold increased the percentage change decreases exponential, based on this evidence, a threshold of twenty-five percent was chosen.

Listing 5.1: 'Example Perl Code'

```
my $sth = $self->{'dbh'}->prepare($command) or die $self->{'dbh'};
$sth->execute($hospid,$date) or die $sth->errstr; #add the data

open(FILE,$file) or die $!;
my $ques = "?,"x17;
chop($ques); #remove the last , from the above string
$command = qq{ REPLACE INTO orgdata(
          hospid,date,time,phase,slice,
          centerx,centery,centerz,
          centroidx,centroidy,centroidz,
          csa,curvx,curvy,curvz,
          volumepolytope,eccentricty)
          VALUES($ques);
};

$sth = $self->{'dbh'}->prepare($command) or die $self->{'dbh'};

my $j = 0;
foreach my $line(<FILE>)
{
          #remove the end line character
     chomp($line);
     #split the line using more then one string as the pattern
     my @para = split(/\s+/,$line);
     #increase the phase so the phases start at 1 and not 0
     ++$para[0];
     #add the hospid,date and time to the beginning of the list
     unshift(@para,($hospid,$date,$time));
     #add the data
     $sth->execute(@para) or die $sth->errstr;
     ++$j;
     #commit every 1000 records
     if($j%1000 == 1)
     {
          $self->{'dbh'}->commit();
     }
}
close(FILE);
```

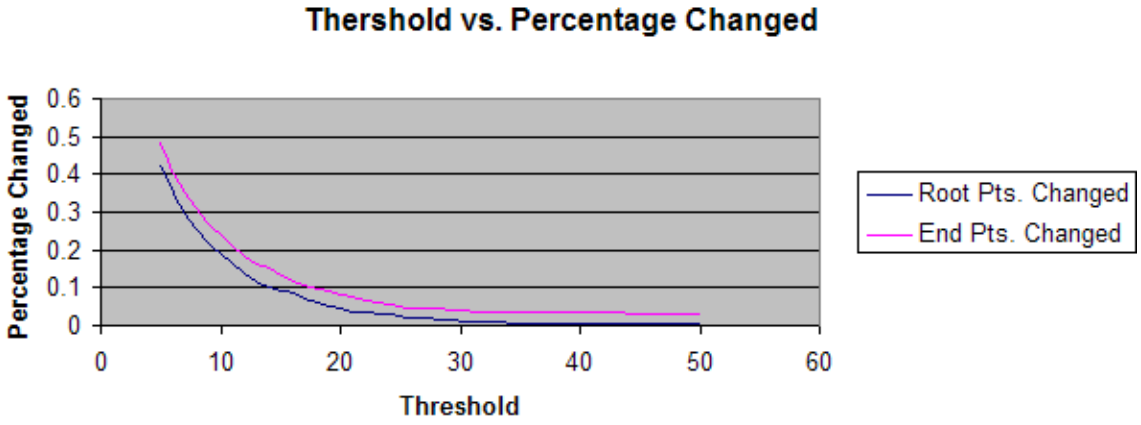



Figure 5.2: Threshold Graph

# CHAPTER 6
# EXPERIMENTAL METHOD

## 6.1 Phantom Generation

In order to verify the accuracy of the extraction and creation process a set of computerized phantoms were created. These phantoms were constructed using the Cartesian equation of a torus and a cylinder. Phantom radii were designed in millimeters and then mapped to voxel space using a predetermined voxel size. Radius sizes ranged from 20 mm up to 60 mm and voxel sizes of 1.1, 1.4, 1.7 and 2.0 $mm^3$. Figure 6.1 is an example of one the phantoms used in verification.

Figure 6.1: Example of a Phantom

## 6.2 Distance Error For Each Phantom

A signed difference between the known radius and the extracted radius was calculated for each slice. These values were averaged together to give an error for a phantom with known radius and voxel size. A signed difference was calculated to

determine if the chosen threshold was underestimating or overestimating. Had an unsigned difference been used the direction of the error would be unknown.

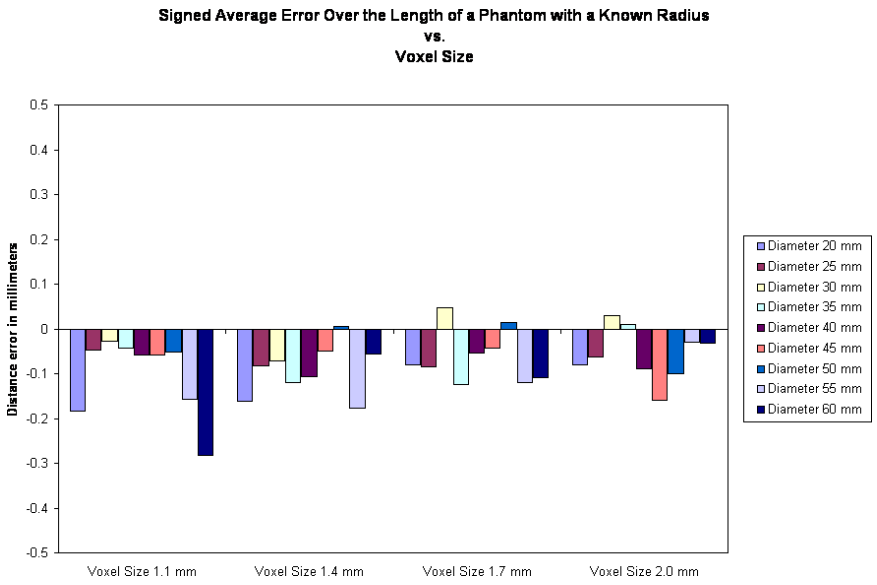



Figure 6.2: Signed error of the phantom test

Figure 6.2 demonstrates how the method performed for a given voxel size and radius size. As it can be seen on average this method can correctly measure a given diameter. One free parameter that was set during this analysis was the location of the edge for the (u,v,w) distance map. When one calculated a distance map in (x,y,z) space the edge was given a voxel intensity of zero. The following method was used to determine if the transformation distorted the edge location and if this information was lost were on the aorta did it occur. First, a range of threshold were used to set the (u,v,w) edge location on each slice of the re-sampled volume, this result was used as input into the normal pipeline. The resulting contours were mapped back and the resulting diameter was measured, figure 6.3 shows how the method performed for the curved section of the aorta versus the straighten parts. As it can be see the edge location of zero was preserved during the transform irregardless of the location on

the aorta.

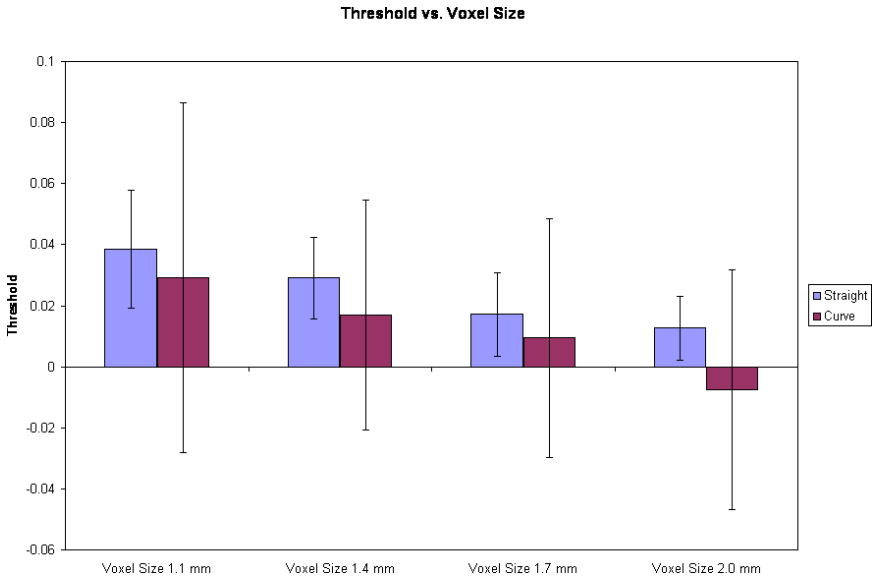



Figure 6.3: Threshold based on location

## 6.3 Extraction of True Segmented Volumes

The extraction process was applied to a set of one hundred sixty-three segmented volumes. This set consisted of eighty-six normals and seventy-seven patients. The root and diaphragm points were taken from a set of files made during the creation of segmented volume. A rigid affine registration with linear interpolation was used to mapped a single root and diaphragm point across all sixteen phases. Table 6.1 provides a detailed description of the extraction process on the given segmented volumes. In all, the extraction process was able to extract one hundred forty-one data sets.

In order for an extraction to be marked as incorrect it must meet one or more of the following criteria:

1. The process failed to complete

| Result | Segmented | Extracted | Percentage Correct |
|---|---|---|---|
| Patient | 77 | 61 | 79.22 |
| Normal | 86 | 69 | 80.23 |

Table 6.1: Summarization of Extractions from Segmented Data

2. The root or end cross sectional area (CSA) measure is approximately zero

3. In the middle of graph comparing CSA versus distance from the root there is a sudden spike.

The reason that an extraction processed failed to complete was due to a root or end point being located outside the given segmented volume. As stated before, these points were based on a rigid registration which can fail in cases were the aorta moves in fashion that was beyond the parameters of this type of registration. One possible solution was to use non-rigid or a deformable registration method, such as Demons Registration. When the CSA versus distance graph had a spike in the middle of it, it typically implies a poor segmentation in that region. The root and diaphragm errors are fixed using the method found in section 5.5. These errors occurred when the registration points fell near the edge of the segmented volume.

# CHAPTER 7
# VERIFICATION AND EDITING TOOL

## 7.1 Verfication Tool

A verification tool was created using C++ and wxWidgets to provide a user with the possibility to verify both the contours created in this work and the segmentation results discussed in [10]. This tool has the ability to view the Candy Cane (CC) volume in the xy-plane, xz-plane, and yz-plane. It can display slices of the perpendicular volume and overlaid contour points. There is a third view of an oblique plane through the perpendicular volume producing a longitudinal view of the aorta. Mouse clicks in the perpendicular slices are mapped back to the correspong CC slice. Mouse clicks in the longitudinal view are mapped to the corresponding perpendicular slice, CC slice, and a CSA versus length from the root graph. By using these three views, a user can determine if the extraction was performed correctly. If it was not, the user can further investigate the source of error and if possible, modify the results as needed. Figure 7.1 illustrates the layout of this tool when loaded with data.

### 7.1.1 Design

In order to correctly implement the design of the given tool, the codes needs to be reliable, reusable, and understandable; in computer science this is called high cohesion. With high cohesion comes low coupling or code that is in separatable into modules that can be changed with little or no affect on the rest of the code. In order to accomplish this task an object was chosen as the hub to handle all events and data in the code and each view as its own object or window. This way if the code of one of the windows is modified then it does not affect another window. In figure 7.2 the hub is the Visual Results Panel and each view is its own module. The double arrow headed lines show that events and data flow in a bidirectional manner.

By using this design one can easily disable windows, add other windows, or add more editing or verification tools. One major design flaw with this layout is code

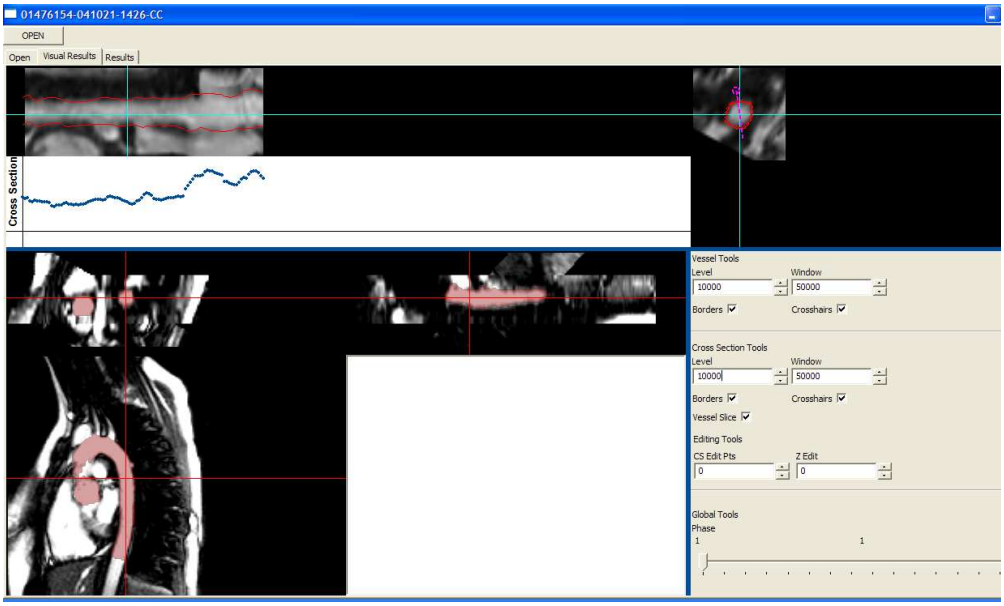



Figure 7.1: Verfication Tool

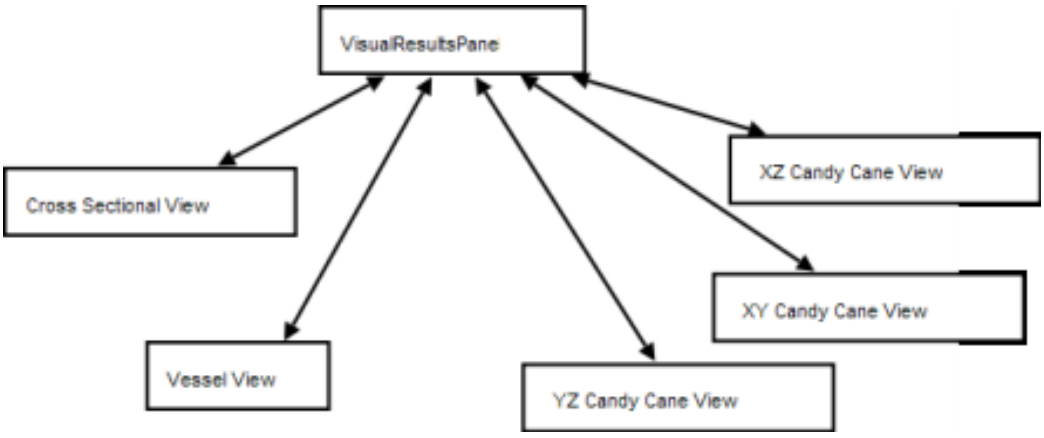



Figure 7.2: Verfication Tool Design

bloat that can occur in Visual Results Panel. Code bloat occurs when there is an excessive amount of code in one location. One possible solution to this is to use interfaces between new windows or tools and the Visual Results Panel. Interfaces would provide methods for a window or tool to communicate with the Visual Results Panel 7.3 while placing the code outside the Visual Results Panel.

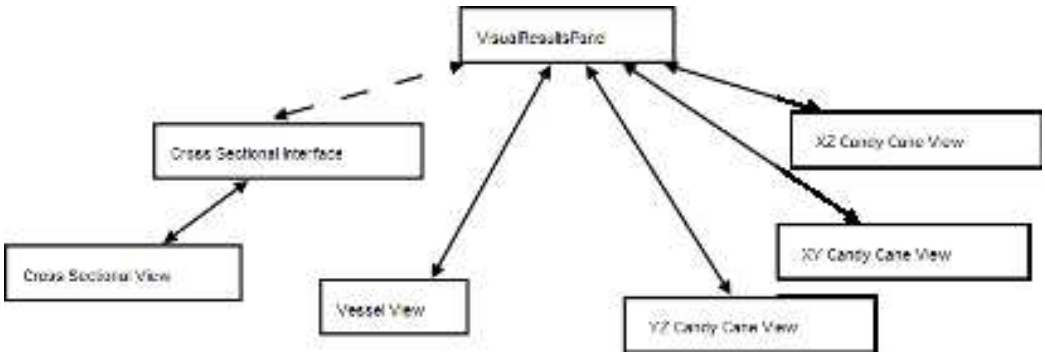



Figure 7.3: Verfication Tool with an Interface

# CHAPTER 8
# CONCLUSION

## 8.1 Restatement of the Aims

1. Extract a set of contour points that represent a given aortic volume
2. Create a set of non-modal indexes based on the above contour points
3. Verify that the given non-modal indexes are correct
4. Provide a program to modify the extract contour points,if needed

## 8.2 Aim 1

As seen in chapter 4 Aim 1 was met. Using a binary volume and a set of C++ programming libraries a successful set of contour points was created. From a set of root and diaphragm root points a centerline was created and used to generate a set of transforms to generate a volume perpendicular to the centerline.

## 8.3 Aim 2

In chapter 5 Aim 2 was successfully accomplished. Using the set of contour points generated in Aim 1, a set of measures were created using a previously developed software package and Perl scripts. In order to maintain and access this large set of data a Standard Query Language (SQL) database was setup and accessed via Perl Scripts and Perl Database Interface. This database removed the amount of overhead needed to access each result separately and in the future could be used in an on-line database to allow multiple users to perform their own analysis.

## 8.4 Aim 3

By constructing a set of phantoms with a known radius, it was soon realized that this method produced similar results irregardless of the voxel size and diameter. It was also shown that the typically edge location in a distance map is changed during the construction of a perpendicular volume.

### 8.5 Aim 4

After the data was extracted and processed the resulting data may have errors. These errors can be caused by local inaccuracies in the segmentation data or by misaligned plane in the extraction process. To fix these errors a verification and modification tool was created using the wxWidgets library. This tool has high cohesion making it reliable, reusable, and understandable. Due to fact that this verification tool has high cohesion further tools could added. For example the segmentation process, a manual tracing tool for those data sets that could not be segmented and way to access the data in the database.

## REFERENCES

[1] R. Medina Universidad de Los Andes Facultad de Ingeniería A. Wahle K. Lee. Morphological analysis. source code.

[2] The National Marfan Foundation. Marfan syndrome facts. http://www.marfan.org. Day Accessed: February 28,2008.

[3] H. Gray. *Anatomy of the human body*. Philadelphia: Lea and Febiger, 1918. Bartleby.com 2000 www.bartleby.com/107.

[4] Massachusetts General Hospital. Aortic dissection. http://www.massgeneral.org/, May 2008. Day Accessed: May 7,2008.

[5] Kitware Incorporated. Visualization toolkit (vtk). http://www.vtk.org, 2008.

[6] MedicineNet. Definition of fbn1. http://www.medterms.com. Day Accessed: February 28,2008.

[7] National Library of Medicine. Insight segmentation and registration toolkit (itk). http://itk.org/, 2008.

[8] Medline Plus. Aortic dissection. http://www.nlm.nih.gov, May 2006. Day Accessed: February 29,2008.

[9] A. Wahle, E. Wellnhofer, I. Mugaragu, H. U. Sauer, H. Oswald, and E. Fleck. Assessment of diffuse coronary artery disease by quantitative analysis of coronary morphology based upon 3-D reconstruction from biplane angiograms. *IEEE Transactions on Medical Imaging*, 14(2):230–241, June 1995.

[10] F. Zhao. *Congenital Aortic Disease: 4D Magnetic Resonance Segmentation and Quantitative Analysis*. PhD thesis, University of Iowa, December 2007.