Building a Parameterized 4D Cardiac Model with Respiratory Motion from 2D MR Time Series

by

Marc Queralt Madrigal

Northeastern University, June 2010

A FINAL PROJECT SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Superior Engineering in Telecommunication

at

Technical University of Catalonia

(ETSETB, UPC)

Abstract

Atrial fibrillation (AF) is a growing problem in modern societies with an enormous impact in both short term quality of life and long term survival. A recently developed promising approach to cure AF uses radiofrequency (RF) ablation to carry out "pulmonary vein antrum isolation" (PVAI), electrically isolating the pulmonary veins from the rest of the atrium. However, the lack of proper 3D surgery training, planning, and guidance, along with current limitations in understanding of the true causes and mechanisms of AF, makes this surgery a very difficult task for the surgeons. Therefore recurrence rates and even failures of the procedure, as well as the risk for the patient, increase.

The purpose of this work is to develop methods for automatically segmenting and tracking the heart in 4-D cardiac MRI datasets. The reconstructed heart surface will serve as a virtual computer model for the 3D surgery training, planning and guidance. The method used in this project is based on an active contour model for segmentation, followed by a spatial-time post-filtering and processing of the data obtained by the segmentation.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge all the people who have helped me develop this thesis.

Firstly, my advisors Dr. Dana Brooks and Dr. Gilead Tadmor deserve a special mention, because they have been very helpful during the whole project. Having discussions with them almost every day has proved as a valuable mean to help bringing abstract ideas to reality. I will always be grateful to them for making this experience one of the most enriching of my life. I have learned to think, I have learned to analyze but, above all, I have learned to overcome difficulties in a search for a great objective.

Secondly, I would like to thank Dr. Rob MacLeod and Dr. Eugene Khomovski from the University of Utah for providing us the data sets. This thesis wouldn't have been a reality without their collaboration.

And last but not least, I would like to thank my family, girlfriend and friends who have always been by my side. All their support has been extremely valuable to succeed in this exciting project.

Dedication

To my sister Andrea.

Chapter 1

Introduction

This work is focused on images of the human heart over time. The movement of the heart is directly present in those images obtained from a set of non-gated MRI. Thus, it is important to know which are the internal processes of the heart, as well as of other organs, that contribute to the creation of a continuous motion.

1.1 Problem Statement: Atrial Fibrillation

1.1.1 Definiton

The normal electrical conduction system of the heart, which is represented in the left image of 1.1, allows the impulse that is generated by the sinoatrial node (SA node) of the heart to be propagated to and electrically stimulate the myocardium (muscle of the heart). When the myocardium is electrically stimulated, it contracts, due to the consequent increase of intracellular calcium. It is the ordered stimulation of the myocardium that allows efficient contraction of the heart, thereby allowing blood to be pumped between the heart chambers and out to the body.

In Atrial Fibrillation (AF), the regular impulses produced by the sinus node for a normal heartbeat, are overwhelmed by rapid electrical discharges produced in the atria and adjacent parts of the pulmonary veins. Sources of these disturbances are either automatic foci, often localized at one of the pulmonary veins, or a small number of reentrant sources (rotors) harbored by the posterior wall of the left atrium near the junctions with the pulmonary veins, as shown in the right image of Figure 1.1. The pathology progresses from paroxysmal to permanent AF as the sources multiply and localize anywhere in the atria. [1]

Stagnant blood can result in a clot, which is called a thrombus while it is immobile at its place of origin. If the clot becomes mobile and is carried away by the blood circulation, it is called an embolus. An embolus proceeds through smaller and smaller arteries until it plugs one of them and prevents blood from flowing any farther in that artery. The result can be damage to tissue that depends on that supply of blood to live. This can occur in various parts of the body, depending on where the embolus ends up. An embolus lodged in an artery of the brain produces the most feared complications, such as a stroke or a heart attack. The formation of a thrombus, movement of the embolus, and plugging of an artery, is called a thromboembolism.

Thus in atrial fibrillation, the lack of an organized atrial contraction can result in stagnant blood in the left atrium (LA) or left atrial appendage (LAA), and this can lead to a thromboembolism. The LAA is the site of thrombus formation in more than 90% of cases of thrombi associated with non-valvular atrial fibrillation, but if the LA

is enlarged, there is an increased risk and percentage of thrombi that originate in the LA.[2] The LAA lies in close relation to the free wall of the left ventricle and thus the LAA's emptying and filling, which determines its degree of blood stagnation, may be significantly affected by left ventricular function. [3, 4]

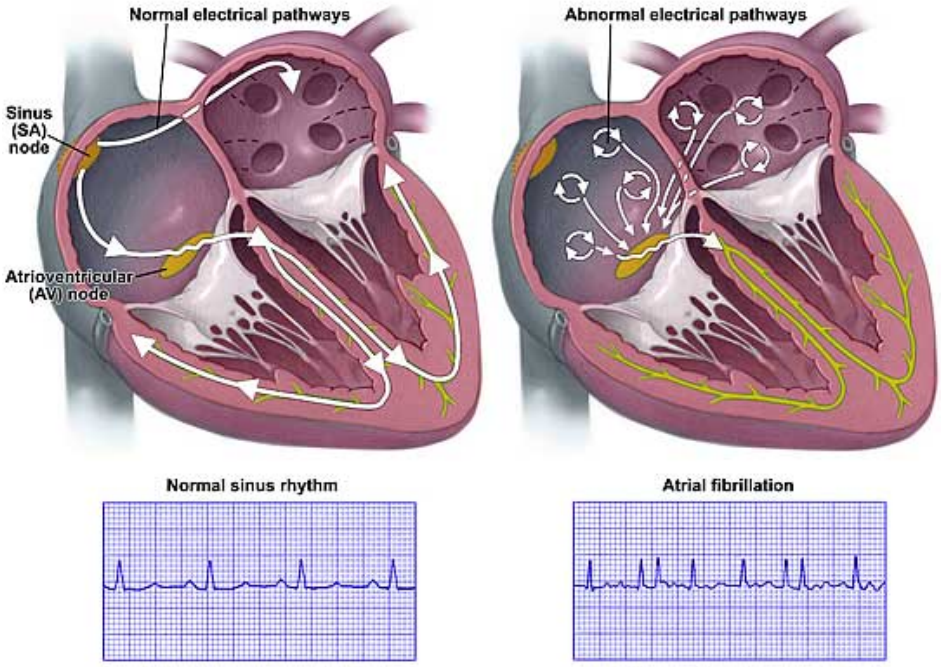

Figure 1.1: Atrial fibrillation. Figure taken from [5].

Atrial Fibrillation is the most common cardiac arrhythmia (abnormal heart rhythm). About 15% of strokes occur in people with atrial fibrillation, and the likelihood of developing atrial fibrillation increases with age. Three to five percent of people over 65 have atrial fibrillation. [6]

1.1.2 Treatment

In the last decade the mechanisms that cause the initiation and perpetuation of atrial fibrillation have been studied. For nearly a century, it has been known that atrial fibrillation exists in various forms of heart disease in which irregularities in the atrial myocardium seem to cause electrical disorganization of atrial depolarization. It has also been recognized that atrial fibrillation occurs in individuals without overt heart disease, but the mechanisms are only now becoming understood. Atrial fibrillation is diagnosed on an electrocardiogram (ECG), an investigation performed routinely whenever an irregular heart beat is suspected. Characteristic findings are the absence of P waves, with unorganized electrical activity in their place, and irregular R-R intervals

due to irregular conduction of impulses to the ventricles. [7, 8]

Cardioversion

Ideally, to treat atrial fibrillation, the heart rate and rhythm are reset to normal. To correct this condition, doctors may be able to reset a heart to its regular rhythm (sinus rhythm) using a procedure called cardioversion, depending on the cause of atrial fibrillation and how long the patient has had it. Cardioversion can be done in two ways:

- Cardioversion with drugs: This form of cardioversion uses medications called anti-arrhythmics to help restore normal sinus rhythm. Depending on the heart condition, the doctor may recommend trying intravenous or oral medications to return a heart to normal rhythm. This is often done in the hospital with continuous monitoring of the heart rate. If the heart rhythm returns to normal, the doctor often will prescribe the same anti-arrhythmic or a similar one to try to prevent more spells of atrial fibrillation.
- Electrical cardioversion: In this brief procedure, an electrical shock is delivered to the heart through paddles or patches placed on the chest. The shock momentarily stops the electrical activity of the heart. When the heart beating begins again, the hope is that it resumes its normal rhythm. The procedure is performed during anesthesia.

Heart Rate Control

Sometimes, though, atrial fibrillation can not be converted to a normal heart rhythm. Then the goal is to slow the heart rate to between 60 and 100 beats a minute (rate control). Heart rate control can be achieved two ways:

- Medications. Traditionally, doctors have prescribed medications that can control heart rate at rest, but not as well during activity. Most people require additional or alternative medications, such as calcium channel blockers or beta blockers.
- Atrioventricular (AV) node ablation. If medications don't work, or they have side effects, AV node ablation may be another option. The procedure involves applying radio frequency energy to the pathway connecting the upper and lower chambers of the heart (AV node) through a long, thin tube (catheter) to destroy this small area of tissue. The procedure prevents the atria from sending electrical impulses to the ventricles. The atria continues to fibrillate, though, and anticoagulant medication is still required. A pacemaker is then implanted to establish a normal rhythm. After AV node ablation, the patient will still need to take blood-thinning medications to reduce the risk of stroke, because the heart rhythm is still atrial fibrillation.

Destructive Procedures

In worst case scenario, medications or cardioversion to control atrial fibrillation do not work. In these cases, the doctor may recommend a procedure to destroy the area of

heart tissue which is causing the erratic electrical signals, or to isolate regions of tissue electrically or otherwise prevent sustainable circuits, in order to restore the heart to a normal rhythm. These options include:

- Radiofrequency catheter ablation. In many people who have atrial fibrillation and an otherwise normal heart, atrial fibrillation is caused by rapidly discharging triggers, or "hot spots." These hot spots are like abnormal pacemaker cells that fire so rapidly that the upper chambers of the heart quiver instead of beating efficiently. Radiofrequency energy directed to these hot spots through a catheter inserted in a vessel near the collarbone or leg may be used to destroy these hot spots, scarring the tissue so the erratic electrical signals are normalized. This corrects the arrhythmia without the need for medications or implantable devices. In some cases, other types of catheters that can freeze the heart tissue (cryotherapy) are used.
- Surgical maze procedure. The maze procedure is often done during an open-heart surgery. Using a scalpel, doctors create several precise incisions in the upper chambers of the heart to create a pattern of scar tissue. Because scar tissue doesn't carry electricity, it interferes with stray electrical impulses that cause atrial fibrillation. Radiofrequency or cryotherapy can also be used to create the scars, and there are several variations of the surgical maze technique. The procedure has a high success rate, but because it usually requires open-heart surgery, it's generally reserved for people who don't respond to other treatments or when it can be done during other necessary heart surgery, such as coronary artery bypass surgery or heart valve repair. Some people need a pacemaker implanted after the procedure. [9]

1.1.3 Challenges

Although radio-frequency ablations are becoming an increasingly common procedure, there are a number of technical difficulties that impede success. As a result the success rate is not as high, and the recurrence rate not as low, as would be desireable. In this thesis we present techniques developed to address one of the impediments to successful catheter-based ablation. In particular, such procedures the chest remains closed and all interventions are carried out using instruments placed at the distal end of catheters which are then inserted through the circulatory system into the heart chambers. This presents obvious challenges to the clinician; Operating inside a beating heart without direct vision of the effector or the effected tissue is extremely challenging.

The problem is compounded by the lack of proper three-dimensional (3D) visualization during the training, planning, and guidance stages of surgery, which can lead to improper patient selection, longer procedures, and increased risks to the patient. To address these issues, the overall project in which this work has taken place aims to develop interactive catheter and ablation guidance strategies for the treatment of AF through real-time, MRI based navigation systems that guide catheter placement and then monitor lesion formation. As part of this larger project, there is a need to generate a 4-D model of the heart so the surgeon has a reference within the anatomy of the heart which will help navigate with the catheter in the closed thoracic cavity.

1.2 The Heart

1.2.1 Anatomy of the heart

The human heart is located in the center of the thoracic cavity between the two lungs, lying on the thoracic face of the diaphragm. It provides a continuous blood circulation through the cardiac cycle and is one of the most vital organs in the human body.

It is divided into four chambers. The two upper chambers are called the left and right atria (LA and RA), and the two lower chambers are called the left and right ventricles (LV and RV). Physicians commonly refer to the right atrium and right ventricle together as the right heart and to the left atrium and ventricle as the left heart. [10]

1.2.2 Physiology of the Heart

The cardiac cycle is a process coordinated by a series of electrical impulses which are produced by specialized heart cells. They can be located within the sino-atrial node (SA) and the atrioventricular node (AV). These cells, also known as myocytes, initiate their own contraction without help of external nerves (with the exception of modifying the heart rate due to metabolic demand).

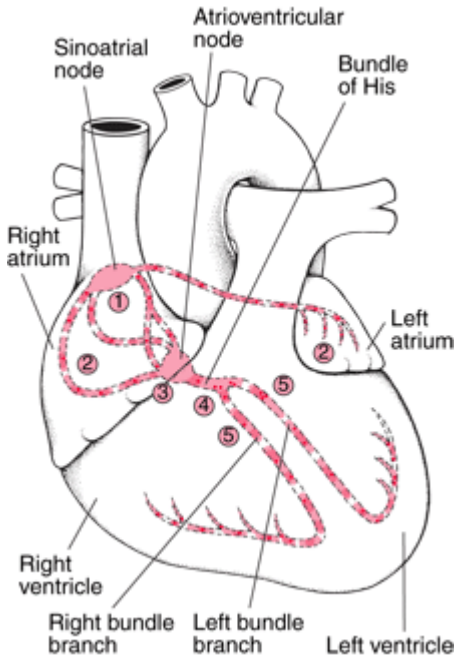

Figure 1.2: The heart's electrical pathway. Figure taken from [11].

Following the pink numbers in Figure 1.2, the action potential generated by the AV node spreads throughout the atria. As the wave of action potentials depolarizes the

atrial muscle, myocytes contract by a process termed excitation-contraction coupling. The AV node slows the impulse conduction, thereby allowing sufficient time for complete atrial depolarization and contraction (systole) prior to ventricular depolarization and contraction.

The impulse then enters the ventricle at the Bundle of Hiss and then follows the left and right bundle branches of Hiss along the interventricular septum. These fibers conduct the impulses at a very high speed. Bundle branches then divide into an extensive system of Purkinge fibers that conduct the impulse at high velocity through the ventricles. This results in rapid depolarization of ventricular myocites through both ventricles. [12]

Under normal circumstances, each cycle takes approximately one second, and the frequency of the cardiac cycle is called the heart rate. Every single normal 'beat' of the heart involves five major stages:

1. Late diastole: The semilunar valves close, the AV valves open and the whole heart is relaxed.

2. Atrial systole: The atria is contracting, AV valves open and blood flows from atrium to the ventricle.

3. Isovolumic ventricular contraction: The ventricles begin to contract, AV valves close, as well as the semilunar valves and there is no change in volume.

4. Ventricular ejection: Ventricles are empty, they are still contracting and the semilunar valves are open.

5. Isovolumic ventricular relaxation: Pressure decreases, no blood is entering the ventricles, ventricles stop contracting and begin to relax, semilunar valves are shut because blood in the aorta is pushing them shut.

1.3 The Lungs

Cardio-circulatory and respiratory systems are intimately related. The movement of one organ may disturb the other, such as during inspiration when there is an increase of the cardiac frequency because the heart can't expand and the diastolic phase gets shorter. Also, the opposite effect happens during expiration. For this reason, it is of great interest to review the anatomy and physiology of the lungs.

1.3.1 Anatomy of the Lungs

Humans have two lungs located in the chest on either side of the heart, with the left being divided into two lobes and the right into three lobes. Their main function is to transport oxygen from the atmosphere into the bloodstream, and to release carbon dioxide from the bloodstream into the atmosphere. This exchange of gases is accomplished in the mosaic of specialized cells that form millions of tiny, exceptionally thin-walled air sacs called alveoli.

Figure 1.3 shows the elements that constitute the lungs, which can be classified in two zones:

- Conductiong zone: Contains the trachea, the bronchi, the bronchioles, and the terminal bronchioles.
- Respiratory zone: Contains the respiratory bronchioles, the alveolar ducts, and the alveoli.

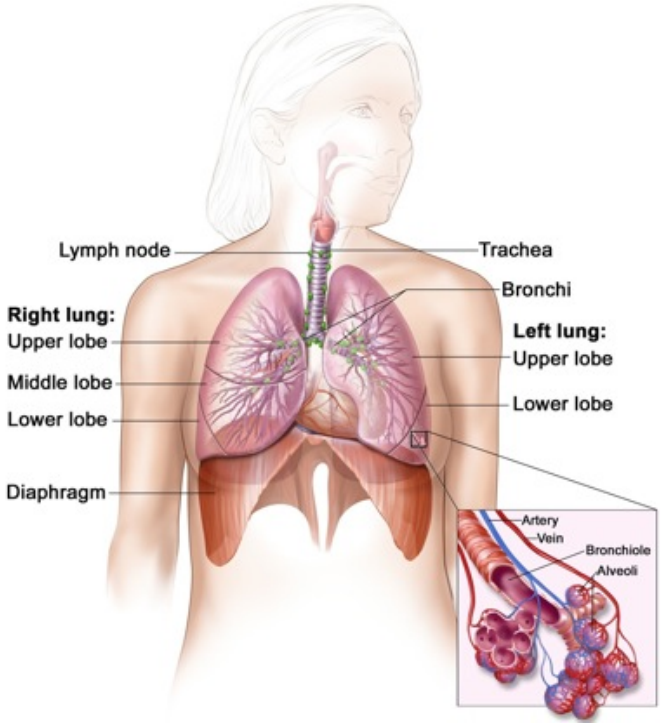

Figure 1.3: Anatomy of the lungs. Figure taken from [13].

The characteristics of a specific pair of lungs will directly affect the adquisition of a data set. Total lung capacity (TLC) includes inspiratory reserve volume, tidal volume, expiratory reserve volume, and residual volume. TLC depends on the person's age, height, weight, sex, and normally ranges between 4,000 and 6,000cm^3 (4 to 6l). For example, females tend to have a 20–25% lower capacity than males. Tall people tend to have a larger total lung capacity than shorter people. Smokers have a lower capacity than nonsmokers. Lung capacity is also affected by altitude. People who are born and live at sea level will have a smaller lung capacity than people who spend their lives at a high altitude. In addition to the total lung capacity, one can also measure the tidal volume, defined as the volume breathed in with an average breath, which is about 500cm^3. [14, 15]

1.3.2 Physiology of the Lungs

To fully understand the respiratory cycle, there are some important pressures that need to be described first:

- Atmospheric (barometric) pressure is the pressure exerted by weight of air in atmosphere on objects on earth's surface.
- Intra-alveolar pressure (sometimes called intrapulmonary pressure) is the pressure within the alveoli.
- Intrapleural pressure (also called intrathoracic pressure) is the pressure exerted outside the lungs within the thoracic cavity.

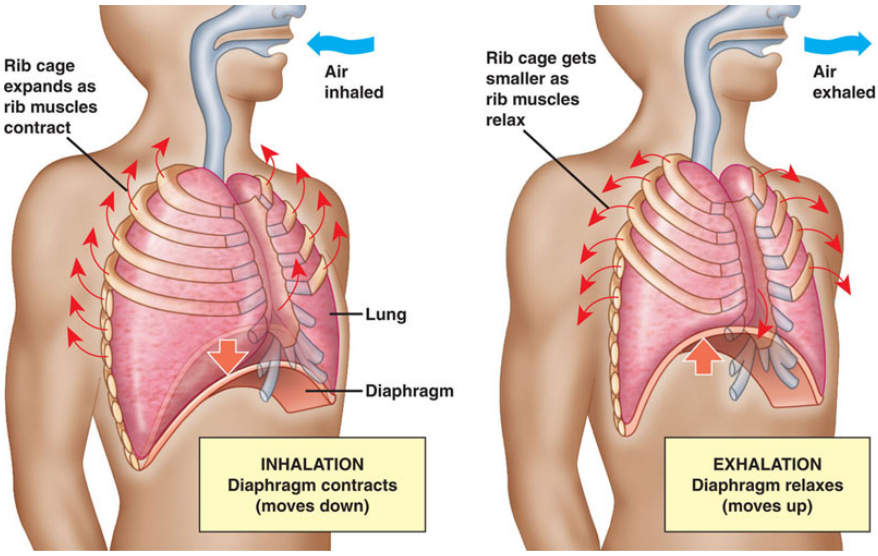

Figure 1.4: Respiratory cycle. Figure taken from [16].

Respiration works by changing the volume of the chest cavity, as seen in 1.4. Before the start of inspiration, respiratory muscles are relaxed, the intra-alveolar pressure is the same as the atmospheric pressure, and no air is flowing. At onset of inspiration, inspiratory muscles (primarily the diaphragm) contract, this results in enlargement of the thoracic cavity. As the thoracic cavity enlarges, the lungs are forced to expand to fill the larger cavity. Because the intra-alveolar pressure is less than atmospheric pressure, air follows its pressure gradient and flows into the lungs until no further gradient exists. Therefore, lung expansion is not caused by movement of air into the lungs. At the end of inspiration, the inspiratory muscles relax, the chest cavity returns to original size, and the lungs return to original size. Although at rest expiration is a passive process, during exercise it is an active process and expiratory muscles (primarily abdominal muscles) contract to decrease the size of the chest cavity during expiration.[17]

1.4 Previous Work

The respiratory motion of the heart has been studied for a long time, although it is still not completely understood. This section encompasses a brief explanation of the most important techniques used to analyse and compensate for the internal movement of the human organs due to respiration.

Around 30 years ago, Dougherty studied the effects of respiration on the electrocardiogram [18, 19]. He noted that the heart undergoes anatomic rotation in the frontal plan with respiration.

The next step in research of respiratory motion was taken by Bogren [20]. He provided the first quantitative study of the respiratory motion of the heart, observing that the superior-inferior motion at the valve planes was half as much as the superior-inferior motion of the diaphragm, which averaged 15mm during normal respiration.

Years later, Wang used 2D MRI at multiple breath-hold levels to conclude that the primary motion of the heart was translation in the superior-inferior direction, and that this movement at the level of the coronary ostia was between 0.6 and 0.7 times the displacement of the diaphragm. [21]

After that, a 3D rigid body analysis of the motion of the heart was presented by McLeish [22]. He obtained 3D MRI datasets of the whole heart at multiple breath-hold levels and registered them using an image intensity method. Rotations and translations were reported for different patients and volunteers, but 6mm of slice thickness and a 180ms temporal resolution were insufficient to isolate the effects of cardiac motion from the respiratory analysis.

More recently, a standard solution for motion compensation problems has been to track changes over short time intervals, using models based on assumptions about small deformations[23]. Other approaches impose rigid models of motion such as periodicity [24, 25]or fixed autoregressive models [26, 27]. Also, some approaches have been recently introduced using more flexible dynamic models for tumor tracking, [28, 29] which has respiratory but not contractile motion, or even for cardiac tracking with both types of motion during robotic surgery [30, 31].

Last but not least, Crosas has studied the respiratory motion of the heart in 2009, using ECG-gated and ungated data to extract information, in order to reproduce the respiratory motion of the heart.

1.5 Aims of the Project

As we have seen in the previous chapters, the aim of the overall project is to develop an interactive strategy for the treatment of AF through real-time, MRI based navigation systems. As a portion of this generic project, the specific research encompassed in this document was developed to generate a 4-D model of the heart as a reference for the surgeon when using catheter navigation. Work was divided in several stages which are listed below:

1. Segmenting the available data set. Section 3.2

2. Detecting position information in the available data set. Section 3.3

3. Finding a method to filter the points that define the heart in space and respiratory phase. Section 3.5
4. Finding a method to interpolate frames and slices that don't exist in the real data set. Section 3.5
5. Finding an average heart that moves only because of the respiratory cycle. Chapter 4
6. Finding a way to obtain sections of the heart through a plane. Section 4.3

Chapter 2

Backgroud Theory

2.1 Snake Algorithm for Contour Detection and Sementation

This section encompasses a brief review of the snake algorithm, which has been used to detect the boundary of the heart in MRI images. The exposition here follows that of [34].

2.1.1 Definiton

A snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. The snake algorithm is used in this work to segment the boundary of the heart in each frame, so a brief explanation is needed.

Representing the position of a snake s parametrically by $\mathbf{v}(s) = (x(s), y(s))$, its energy functional can be written as

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))\,ds = \int_0^1 E_{int}(\mathbf{v}(s)) + E_{ext}(\mathbf{v}(s))\,ds \tag{2.1}$$

where E_{int} represents the internal energy of the spline due to bending, and E_{ext} consists of both the image forces E_{img} and the external constraint forces E_{con} which can be expressed as

$$E_{ext} = \int_0^1 E_{img}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))\,ds$$

In order to simply the algorithm, the external constraint forces will be disregarded.

2.1.2 Image Energy

A crucial step in the snake model is finding a suitable term for the image energy. This is because the snake will try to position itself in areas of low energy. The snake should be attracted to edges in the image and when this is the case a suitable energy term is $E_{img} = -\parallel \nabla I((x,y) \parallel^2$ where I is the image function.

In order to remove noise from the image and increase the capture range of the snake, the image can be convolved with a Gaussian kernel before computing the gradients. This gives the following image energy term

$$E_{img} = - \parallel \nabla \left[G_\sigma (x,y) * I(x,y) \right] \parallel^2$$

where $G_\sigma (x,y)$ is a two dimensional Gaussian with standard deviation σ. When strong edges in the image are blurred by the Gaussian the corresponding gradient is also smoothed, which results in the snake coming under the influence of the gradient forces from a greater distance, hereby increasing the capture range of the snake.

2.1.3 Internal Energy

In addition to the forces given by the image energy, the snake is also under the influence of its own internal energy. The internal energy of the snake is defined as

$$E_{int} = \frac{1}{2} \left(\alpha (s) \parallel \mathbf{v_s} (s) \parallel^2 + \beta (s) \parallel \mathbf{v_{ss}} (s) \parallel^2 \right) \tag{2.2}$$

The first order term $\parallel \mathbf{v_s} (s) \parallel^2$ measures the elasticity of the snake, while the second order term $\parallel \mathbf{v_{ss}} (s) \parallel^2$ measures the curvature energy. The influence of each term is controlled by the coefficients $\alpha (s)$ and $\beta (s)$. The more the snake is stretched at a point $\mathbf{v}(s)$ the greater the magnitude of the first order term, whereas the magnitude of the second order term will be greater in places where the curvature is high. It should be noted that if the snake is not under the influence of any image energy and only moves to minimize its own internal energy, for a closed curve, it will take the form of a circle that keeps shrinking. For a open curve the snake will position itself to form a straight line that also shrinks. However, the internal forces of the snake are necessary in order to preserve desirable properties such as continuity and smoothness of the curve.

The first term in the internal energy of the snake is defined as the magnitude of the first derivative of the parametric curve. The first derivative of a parametric curve is given by

$$\mathbf{v}_s (s) = \begin{bmatrix} x_s (s) \\ y_s (s) \end{bmatrix}$$

This derivative is the tangent vector to the curve at any point, and the direction of the tangent vector is the direction that the curve has at the point. The magnitude/lenght of the tangent vector states the speed of the curve at the point, and the speed of a parametric curve indicates the distance covered on the curve when s is incremented in equal steps. Therefore when the internal energy of the snake is being minimized, the speed of the curve will be diminished in places where the speed is high. This means that the elasticity force helps keeping the curve from excessive deformation. The elasticity force also has the effect of making the curve shrink, even when it is not deformed. Thus, the shrinking effect allows the snake to shrink around the object whose edges are defined by the image energy. However it might be necessary to adjust the parameter $\alpha (s)$ in order to prevent the elasticity force from overcoming the image energy completely, as this would result in the snake disregarding the image energy and shrinking into a single point.

The magnitude of the second derivative of the parametric curve $\| \mathbf{v_{ss}}(s) \|$ is used to measure how much the snake curves at a point. The curvature k of a parametric curve is usually given by the second derivative of the curve with regards to the arc length l

$$\mathbf{v}_{ll}(s) = k\mathbf{N}$$

where $\mathbf{N}$ is the unit principal normal vector. The magnitude of the curvature equals

$$\| \mathbf{v}_{ll(s)} \| = \| k\mathbf{N} \| = | k | \| \mathbf{N} \| = | k | 1 = | k |$$

The parameter $\beta(s)$ that controls the influence of the curvature energy must be adjusted, so that the snake should is able to bend somewhat in places with high image energy while staying straight in places with low image energy.

2.1.4 Minimization

The minimization procedure is a $O(n)$ iterative technique using sparse matrix methods. Each iteration effectively takes implicit Euler steps with respect to the internal energy and explicit Euler steps with respect to the image and external constraint energy.

Let $E_{ext} = E_{image} + E_{con}$. When $\alpha(s) = \alpha$ and $\beta(s) = \beta$ are constants, minimizing the energy functional of 2.1 gives rise to the following two independent Euler equations:

$$\alpha x_{ss} + \beta x_{ssss} + \frac{\delta E_{ext}}{\delta x} = 0$$

$$\alpha y_{ss} + \beta y_{ssss} + \frac{\delta E_{ext}}{\delta y} = 0$$

When $\alpha(s)$ and $\beta(s)$ are not constant, it is simpler to go directly to a discrete formulation of the energy functional in equation 2.2. Then

$$E^*_{snake} = \sum_{i=1}^{n} E_{int}(i) + E_{ext(i)}$$

Approximating the derivatives with finite differences and converting to vector notation with $\mathbf{v}_i = (x_i, y_i) = (x(ih), y(ih))$, $E_{int}(i)$ can be expanded as

$$E_{int}(i) = \alpha_i | \mathbf{v}_i - \mathbf{v}_{i-1} |^2 / 2h^2 + \beta_i | \mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1} |^2 / 2h^4$$

where $\mathbf{v}(0) = \mathbf{v}(n)$.

Let $f_x(i) = \frac{\delta E_{ext}}{\delta x_i}$ and $f_y(i) = \frac{\delta E_{ext}}{\delta y_i}$, where the derivatives are approximated by a finite difference if they cannot be computed analytically. Now the corresponding Euler equations are

$$\begin{aligned}\alpha_i(\mathbf{v}_i - \mathbf{v}_{i-1}) - \alpha_{i+1}(\mathbf{v}_{i+1} - \mathbf{v}_i) + \beta_{i-1}[\mathbf{v}_{i-2} - 2\mathbf{v}_{i-1} + \mathbf{v}_i] \\ - 2\beta_i[\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}] + \beta_{i+1}[\mathbf{v}_i - 2\mathbf{v}_{i+1} + \mathbf{v}_{i+2}] + (f_x(i), f_y(i)) = 0\end{aligned}$$

The above Euler equations can be written in matrix form as

$$\mathbf{A}\mathbf{x} + \mathbf{f}_{\mathbf{x}}(\mathbf{x}, \mathbf{y}) = 0 \tag{2.3}$$

$$\mathbf{A}\mathbf{y} + \mathbf{f}_{\mathbf{y}}(\mathbf{x}, \mathbf{y}) = 0 \tag{2.4}$$

where A is a pentadiagonal banded matrix.

To solve equations 2.3 and 2.4, the right-hand sides of the equations are set equal to the product of a step size and the negative time derivatives of the left-hand sides. It must be taken into account that derivatives of the external forces used require changing **A** at each iteration, so faster iteration is achieved by simply assuming that f_x and f_y are constant during a time step. This yields an explicit Euler method with respect to the external forces. The internal forces, however, are completely specified by the banded matrix, so the time derivative can be evaluated at time t rather than time $t-1$ and therefore arrive at an implicit Euler step for the internal forces. The resulting equations are

$$\mathbf{A}\mathbf{x}_t + \mathbf{f}_{\mathbf{x}}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}) = -\gamma(\mathbf{x}_t - \mathbf{x}_{t-1}) \tag{2.5}$$

$$\mathbf{A}\mathbf{y}_t + \mathbf{f}_{\mathbf{y}}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}) = -\gamma(\mathbf{y}_t - \mathbf{y}_{t-1}) \tag{2.6}$$

where γ is a step size. At equilibrium, the time derivative vanishes and there is a solution of equations 2.3 and 2.4.

Equations 2.5 and 2.6 can be solved by matrix inversion:

$$\mathbf{x}_t = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\gamma\mathbf{x}_{t-1} - \mathbf{f}_{\mathbf{x}}(x_{t-1}, y_{t-1})) \tag{2.7}$$

$$\mathbf{y}_t = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\gamma\mathbf{y}_{t-1} - \mathbf{f}_{\mathbf{y}}(x_{t-1}, y_{t-1})) \tag{2.8}$$

The matrix $\mathbf{A} + \gamma\mathbf{I}$ is a pentadiagonal banded matrix, so its inverse can be calculated by *LU* decompositions in $O(n)$ time. Hence equations 2.7 and 2.8 provide a rapid solution to equations 2.5 and 2.6. The method is implicit with respect to the internal forces, so it can solve very rigid snakes with large step sizes. If the external forces become large, however, the explicit Euler steps of external forces will require much smaller step sizes.

2.2 Singular Value Decomposition

This section contains an overview of the Singular Value Decomposition method, as well as some of its most important properties used in this Thesis. For additional content, please refer to [35, 36, 37].

2.2.1 Definition

In linear algebra, the singular value decomposition (SVD) is an important factorization of a rectangular real or complex matrix, with many applications in signal processing and statistics.

Suppose M is an $m \times n$ matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization of the form

$$M = U\Sigma V^* \tag{2.9}$$

where U is an $m \times m$ unitary matrix over K, the matrix Σ is $m \times n$ diagonal matrix with nonnegative real numbers on the diagonal, and V^* denotes the conjugate transpose of V, an $n \times n$ unitary matrix over K. Such a factorization is called the singular-value decomposition of M. A common convention is to order the diagonal entries $\Sigma_{i,i}$ in descending order. In this case the diagonal matrix Σ is uniquely determined by M, though the matrices U and V are not, and the diagonal entries of Σ are known as the singular values of M. Also, if r is the rank of M, and thus $r \leqq min((m,n))$, the first r columns of U and the first r rows of V are unique.

2.2.2 Relation to Singular Values and Singular Vectors

A non-negative real number σ is a singular value for M if and only if there exist unit-length vectors u in K^m and v in K^n such that

$$Mv = \sigma u$$

and

$$M^*u = \sigma v$$

The vectors u and v are called left-singular and right-singular vectors for σ, respectively. In any singular value decomposition such as 2.9, the diagonal entries of Σ are equal to the singular values of M. The columns of U and V are, respectively, left- and right-singular vectors for the corresponding singular values. Consequently, an $m \times n$ matrix M has at least one and at most $p = \min(m,n)$ distinct singular values. Also, it is always possible to find a unitary basis for K^m consisting of left-singular vectors of M, or a unitary basis for K^n consisting of right-singular vectors of M.

2.2.3 Eigenvalue Decomposition

The SVD is very general in the sense that it can be applied to any $m \times n$ matrix, whereas eigenvalue decomposition can only be applied to certain classes of square matrices. Nevertheless, the two decompositions are related.

Given the SVD of M described in 2.9, two relations can be defined:

$$M^*M = V\Sigma^*U^*U\Sigma V^* = V(\Sigma^*\Sigma)V^*$$

$$MM^* = U\Sigma V^* V\Sigma^* U^* = U\left(\Sigma\Sigma^*\right)U^*$$

The right hand sides describe the eigenvalue decompositions of the left hand sides. Consequently, the squares of the non-zero singular values of M are equal to the non-zero eigenvalues of both M^*M and MM^*. Furthermore, the columns of U (left singular vectors) are eigenvectors of MM^* and the columns of V (right singular vectors) are eigenvectors of M^*M.

In the special case that M is a normal matrix, which by definition must be square, such as the correlation matrix, the spectral theorem says that it can be unitarily diagonalized using a basis of eigenvectors, so that it can be written $M = UDU^*$, for a unitary matrix U and a diagonal matrix D. When M is Hermitian positive semi-definite, the decomposition $M = UDU^*$ is also a singular value decomposition.

2.2.4 Relation to the Spectral Theorem

Let M be an m-by-n matrix with complex entries. M^*M is positive semi-definite and Hermitian. By the spectral theorem, there exists a unitary n-by-n matrix V such that

$$V^*M^*MV = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$$

where D is diagonal and positive definite. V can be appropriately partitioned so

$$\begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} M^*M \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} V_1^*M^*MV_1 & V_1^*M^*MV_2 \\ V_2^*M^*MV_1 & V_2^*M^*MV_2 \end{bmatrix} = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$$

Therefore

$$V_1^*M^*MV_1 = D$$

Define

$$U_1 = MV_1D^{-\frac{1}{2}}$$

Then

$$U_1D^{\frac{1}{2}}V_1^* = MV_1D^{-\frac{1}{2}}D^{\frac{1}{2}}V_1^* = M$$

which is almost the desired result, except that U_1 and V_1 are not unitary in general, but merely isometries. To finish the argument, these matrices must be filled out to become unitary. For example, U_2 can be chosen such that

$$U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$$

is unitary.
Define

$$\Sigma = \begin{bmatrix} \begin{bmatrix} D^{\frac{1}{2}} & 0 \\ 0 & 0 \end{bmatrix} \\ 0 \end{bmatrix}$$

where extra zero rows are added or removed to make the number of zero rows equal the number of columns of U_2. Then

$$\begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} D^{\frac{1}{2}} & 0 \\ 0 & 0 \end{bmatrix} \\ 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^* = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} D^{\frac{1}{2}} V_1^* \\ 0 \end{bmatrix} = U_1 D^{\frac{1}{2}} V_1^* = M$$

which is the desired result in 2.9.

The argument could begin with diagonalizing MM^* rather than M^*M, which shows that MM^* and M^*M have the same non-zero eigenvalues.

2.3 Fast Fourier Transform

This section embodies a short description of the most important concepts used in this thesis about the Fast Fourier Transformation. For additional content, please refer to [38, 39, 40, 41].

2.3.1 Definition

A Discrete Fourier Transform (DFT) decomposes a sequence of values into components at different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. A Fast Fourier Transform (FFT) is a way to compute the same result more quickly.

Let $x_0, \ldots, x_{N-1}$ be complex numbers. The DFT is defined by the formula

$$X_k = \sum_{0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1$$

Obtaining a DFT of N points using the definition takes $O\left(N^2\right)$ arithmetical operations, while an FFT can compute the same result in only $O(N \log N)$ operations. The difference in speed can be substantial, especially for long data sets where N may be in the thousands or millions. Computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to $\frac{N}{\log(N)}$.

2.3.2 Cooley-Tukey Algorithm

The most common FFT is the Cooley–Tukey algorithm. It is based on a divide and conquer solution, that recursively breaks down a DFT of any composite size $N = N_1 N_2$ into many smaller DFTs of sizes N_1 and N_2, along with $O(N)$ multiplications by complex roots of unity, traditionally called "twiddle factors". The most well-known use of this algorithm is to divide the transform into two pieces of size N/2 at each step, and is therefore limited to power-of-two sizes, but any factorization can be used in general.

2.3.3 Multidimensional FFT

The multidimensional DFT of a multidimensional array $x_{n_1,n_2,\cdots,n_d}$ that is a function of d discrete variables, where $n_l = 0,\cdots,N_l - 1$ for l in $1,2,\ldots,d$, is defined by

$$X_{k_1,k_2,\ldots,k_d} = \sum_{n_1=0}^{N_1-1} \left(\omega_{N_1}^{k_1 n_1} \sum_{n_2=0}^{N_2-1} \left(\omega_{N_2}^{k_2 n_2} \cdots \sum_{n_d=0}^{N_d-1} \omega_{N_d}^{k_d n_d} \cdot x_{n_1,n_2,\ldots,n_d} \right) \cdots \right) \tag{2.10}$$

where $\omega_{N_l} = e^{\left(\frac{-2\pi i}{N_l}\right)}$ and the d output indices run from $k_l = 0,1,\ldots,N_l - 1$.

Define $\mathbf{n} = (n_1,n_2,\ldots,n_d)$, $\mathbf{k} = (k_1,k_2,\ldots,k_d)$ as d-dimensional vectors of indices from 0 to $\mathbf{N}-1$, and $\mathbf{N}-1 = (N_1-1,N_2-1,\ldots,N_d-1)$, so 2.10 can be wirtten as the compact expression

$$X_k = \sum_{\mathbf{n}=0}^{\mathbf{N}-1} e^{-2\pi i \mathbf{k}\cdot\left(\frac{\mathbf{n}}{\mathbf{N}}\right)} x_{\mathbf{n}}$$

where $\frac{\mathbf{n}}{N} = \left(\frac{n_1}{N_1},\cdots,\frac{n_d}{N_d}\right)$ is performed element-wise.

In other words, this d-dimensional FFT is the composition of a sequence of d sets of one-dimensional DFTs, performed along one dimension at a time, in any order. This compositional viewpoint immediately provides the simplest and most common multidimensional FFT algorithm, known as the row-column algorithm. It is based on calculating a sequence of d one-dimensional FFTs, i.e. first transforming along the n1 dimension, then along the n2 dimension, and so on.

In the specific case of two dimensions, $x_{\mathbf{k}}$ can be viewed as an $n_1 \times n_2$ matrix, and this algorithm corresponds to first performing the FFT of all the rows and then of all the columns, or vice versa. In more than two dimensions, it is often advantageous for cache locality to group the dimensions recursively. For example, a three-dimensional FFT might first perform two-dimensional FFTs of each planar slice for each fixed n_1, and then perform the one-dimensional FFTs along the n_1 direction.

2.4 Associated Legendre Functions

This section encloses a brief introduction to the Associated Legendre Functions. For additional content, please refer to [42, 43].

2.4.1 Definition

The associated Legendre functions are the canonical solutions of the general Legendre equation

$$\left(1-x^2\right)y'' - 2xy' + \left(l\left[l+1\right] - \frac{m^2}{1-x^2}\right)y = 0$$

where the indices l and m, which in general are complex quantities, are referred to as the degree and order of the associated Legendre function, respectively. This equation

has nonzero solutions that are nonsingular on $[-1,1]$ only if l and m are integers with $0 \leq m \leq l$, or with trivially equivalent negative values.

The Legendre ordinary differential equation is frequently encountered in physics and other technical fields. In particular, it occurs when solving Laplace's equation, and related partial differential equations, in spherical coordinates. Associated Legendre functions play a vital role in the definition of spherical harmonics.

2.4.2 Degree and Order

Associated Legendre functions are denoted $P_l^m(x)$, where the superscript indicates the order, and not a power of P. Their most straightforward definition is in terms of derivatives of ordinary Legendre polynomials

$$P_l^m(x) = (-1)^m \left(1-x^2\right)^{\frac{m}{2}} \frac{d^m}{dx^m}\left(P_l(x)\right) \qquad m \geq 0$$

where if m is zero and l integer, the functions are identical to the Legendre polynomials.

Also, the range of m can be extended to $-l \leq m \leq l$ with the relation

$$P_l^{-m}(x) = (-1)^m \frac{(l-m)!}{(l+m)!} P_l^m(x)$$

2.5 Spherical Harmonics

This section contains a brief review of the spherical harmonics. For additional content, please refer to [44, 45, 46, 47].

2.5.1 Definition

Spherical harmonics are the angular portion of a set of solutions to Laplace's equation. Represented in a system of spherical coordinates, Laplace's spherical harmonics are a specific set of spherical harmonics that forms an orthogonal system.

2.5.2 Normalization

Several different normalizations are in common use for the Laplace spherical harmonic functions. These functions are generally defined as

$$Y_l^m(\theta,\varphi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta)\, e^{im\varphi}$$

which are orthonormal

$$\int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} Y_l^m Y_{l'}^{m'*} d\Omega = \delta_{ll'}\delta_{mm'}$$

where $\delta_{aa} = 1$, $\delta_{ab} = 0$ if $a \neq b$ and $d\Omega = \sin\theta d\varphi d\theta$.

2.5.3 Real Basis

A real basis of spherical harmonics is given by setting

$$Y_{lm} = \begin{cases} Y_l^0 & m = 0 \\ \frac{1}{\sqrt{2}}\left(Y_l^m + (-1)^m Y_l^{-m}\right) = \sqrt{2}N_{(l,m)}P_l^m(\cos\theta)\cos m\varphi & m > 0 \\ \frac{1}{i\sqrt{2}}\left(Y_l^{-m} - (-1)^m Y_l^m\right) = \sqrt{2}N_{(l,m)}P_l^{-m}(\cos\theta)\sin m\varphi & m < 0 \end{cases}$$

where $N_{(l,m)}$ denotes the normalization constant as a function of l and m. The real form requires only associated Legendre functions $P_l^{|m|}$ of non-negative $\mid m \mid$. The harmonics with $m > 0$ are said to be of cosine type, and those with $m < 0$ of sine type.

2.5.4 Visualization

Figure 2.1 shows how the Laplace spherical harmonics Y_l^m can be visualized by considering their "nodal lines", that is, the set of points on the sphere where Y_l^m vanishes.

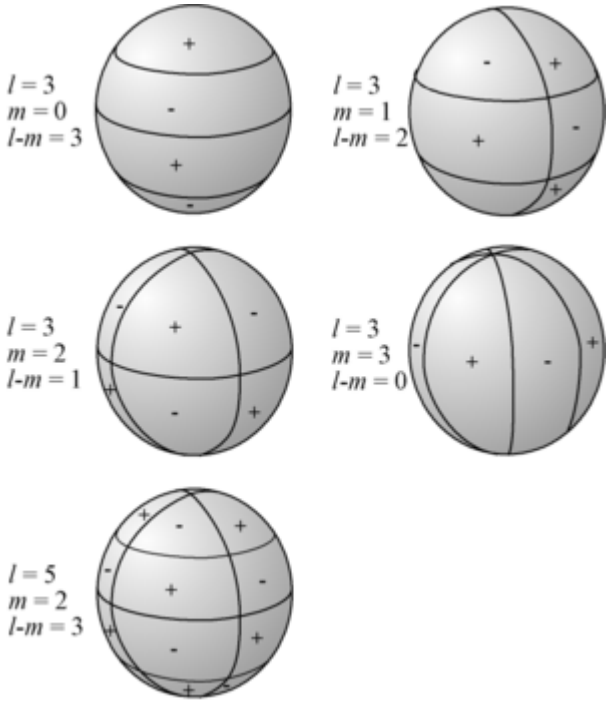

Figure 2.1: Nodal lines in the Laplace spherical harmonics

Nodal lines of Y_l^m are composed of circles, some of which are latitudes and others are longitudes. One can determine the number of nodal lines of each type by counting the number of zeros of Y_l^m in the latitudinal and longitudinal directions independently. For the latitudinal direction, the associated Legendre functions possess $l - \mid m \mid$ zeros, whereas for the longitudinal direction the trigonometric sin and cos functions possess $2 \mid m \mid$ zeros.

When the spherical harmonic order m is zero, the spherical harmonic functions do not depend upon longitude and are referred to as zonal. When $l = \mid m \mid$, there are no zero crossings in latitude and the functions are referred to as sectoral. For the other cases, the functions checker the sphere and they are referred to as tesseral.

2.6 Rotation matrix

This section includes a short description of some basic facts about rotation matrices. For additional content, please refer to [48].

In linear algebra, a rotation matrix is any matrix that acts as a rotation in Euclidean space. Though most applications involve rotations in 2 or 3 dimensions, rotation matrices can be defined for n-dimensional space. They are always square and are usually assumed to have real entries, though the definition makes sense for other scalar fields. The set of all n × n rotation matrices forms a group, known as the rotation group, or special orthogonal group.

2.6.1 Basic rotation

There are three basic rotation matrices in the three dimensions in Cartesian coordinates:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

R_x rotates the Y axis towards the Z axis, R_y rotates the Z axis towards the X axis, and R_z rotates the X axis towards the Y axis.

2.6.2 Right hand rule

The right-hand rule is a common mnemonic for understanding notation conventions for vectors in 3 dimensions. There are two orientation solutions when choosing three vectors that must be at right angles to each other, so there is a need to define which

solution is used. In other words, some kind of convention must be specified in these situations.

For example, there are situations in which an ordered operation must be performed on two vectors a and b that has a result which is a vector c perpendicular to both a and b, such as the vector cross product.

$$\overrightarrow{c} = \overrightarrow{a} \times \overrightarrow{b}$$

With the thumb, index pointed straight, and middle fingers at right angles to each other, the thumb points in the direction of c when the index represents a and the middle finger represents b.

2.6.3 General rotation

Every rotation in three dimensions is defined by:

- Axis: Direction that is left fixed by the rotation.
- Angle: Amount by which the rotation turns.

To find the axis, given a rotation matrix$R$, a vector $\mathbf{u}$ parallel to the rotation axis must satisfy

$$R\mathbf{u} = \mathbf{u}$$

since the rotation of $\mathbf{u}$ around the rotation axis must result in itself. Further, the equation may be rewritten

$$R\mathbf{u} = I\mathbf{u}$$

which shows that is $\mathbf{u}$ is the null space of R–I. That is, $\mathbf{u}$ is an eigenvector corresponding to the eigenvalue $\lambda = 1$.

To find the angle of rotation once the axis is a known parameter, a vector perpendicular to the axis must be selected. The angle of the rotation is the angle between $\mathbf{v}$ and $R\mathbf{v}$.

For some applications, it is helpful to be able to make a rotation with a given axis. Given a unit vector $u = (u_x, u_y, u_z)$, the matrix for a rotation by an angle of θ about an axis in the direction of $\mathbf{u}$ is:

$$R = \begin{bmatrix} u_x^2 + \left(1 - u_x^2\right)\cos\theta & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_x u_y(1-\cos\theta) - u_z\sin\theta & u_y^2 + \left(1 - u_y^2\right)\cos\theta & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_x u_z(1-\cos\theta) + u_y\sin\theta & u_y u_z(1-\cos\theta) - u_x\sin\theta & u_z^2 + \left(1 - u_z^2\right)\cos\theta \end{bmatrix}$$

This matrix can be more concisely written as

$$R = \mathbf{u} \otimes \mathbf{u} + \cos\theta(I - \mathbf{u} \otimes \mathbf{u}) + \sin\theta\,[\mathbf{u}]_x$$

where $[\mathbf{u}]_x$ is the skew symmetric form of $\mathbf{u}$, and $\otimes$ is the outer product.

If the 3D space is oriented in the usual way, the rotation will be counterclockwise for an observer placed so that the $\mathbf{u}$ axis goes in his direction, following the right-hand rule.

Chapter 3

Reconstruction of a Heart for each Respiratory Phase

3.1 Data Sets

This section briefly describes how this work is based on two different linked types of images, one that contains images of the real heart and another that contains relative position information.

3.1.1 Sagittal Data Set

The 4D MRI data set consists of sagittal images of a healthy human.

Concept	Value
Viewpoint	sagittal
Type of Image	MRI
Intra Slice Resolution	2mmx2mm
Inter Slice Resolution	8mm

Table 3.1: sagittal Data Features

Each image showing a 2D slice at a point in time is called a frame, and for every sagittal position, namely a slice, there is one set of frames. There are 16 slices of the heart, and every slice consists of 50 frames which have been captured at consecutive moments in time. 3.1 shows the first 20 frames for the fourth slice of the data set.

These simple concepts described above are very important to understand the organization of the data. Also, it is important to note that the MRI data provided for the work in this thesis are non-gated. In other words, no imaging constraints have been taken to avoid the influence of both respiratory and cardiac motions in the images, which increases the difficulty to obtain reasonable results from them.

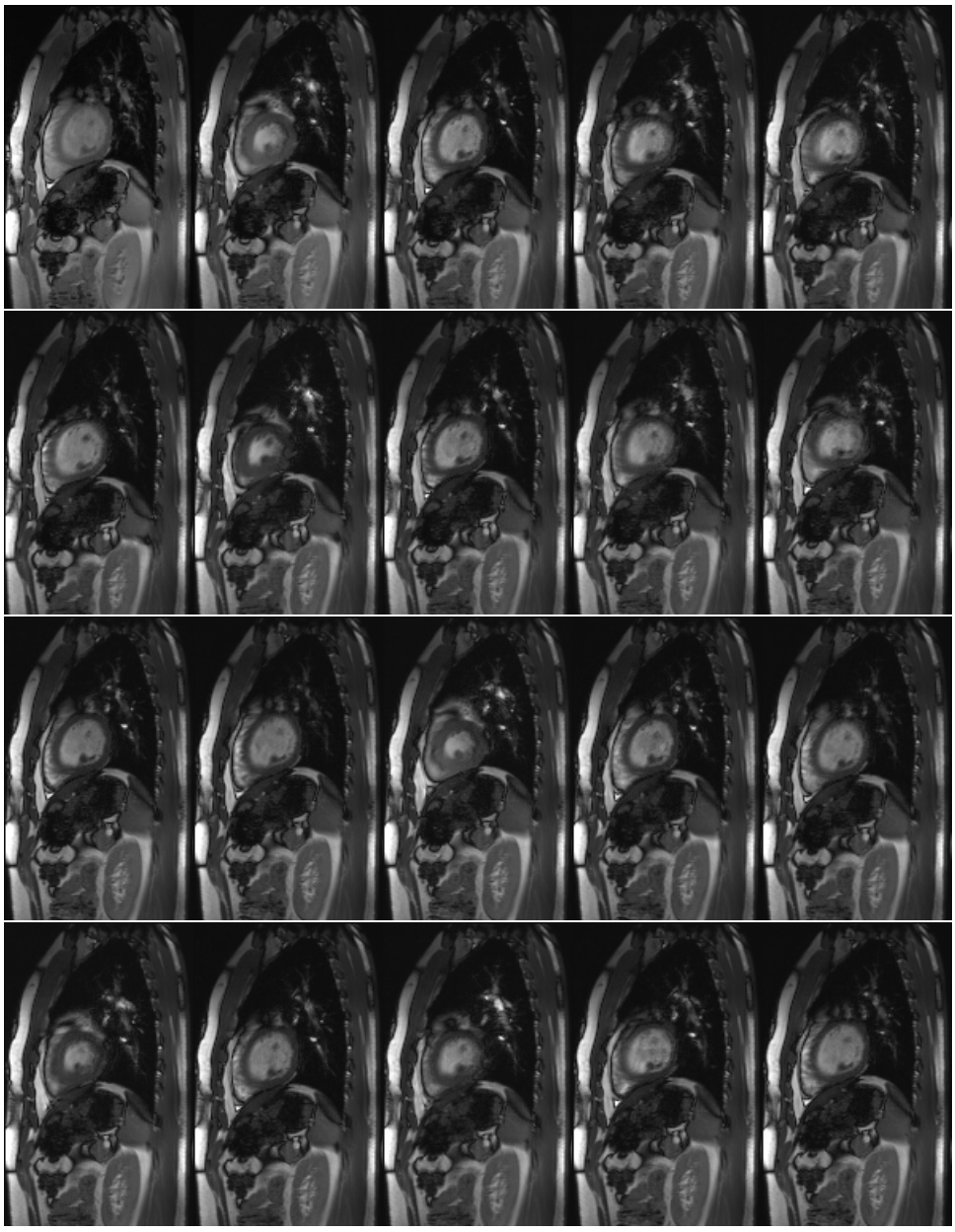

Figure 3.1: Frames 1 to 20 for slice in position #4

3.1.2 Navigator Data Set

There is no synchronization between frames of different slices, and in particular no synchronization in terms of respiratory phase. This means that the sample times of the frames of each slice begin and end arbitrarily with respect to respiratory phase. One of the aims of this project is to establish a new relationship between the slices. In order to accomplish this objective, a second data set was available, which consisted of a set of what are called navigator images.

Navigator images are created following the movement of an internal point of the human body, i.e. in this case the top of the liver. For every frame in the sagittal data set, the position of that specific point is saved and represented in a simple bar diagram, as shown in 3.2. Every set of navigator images contains 5 to 6 respiratory cycles.

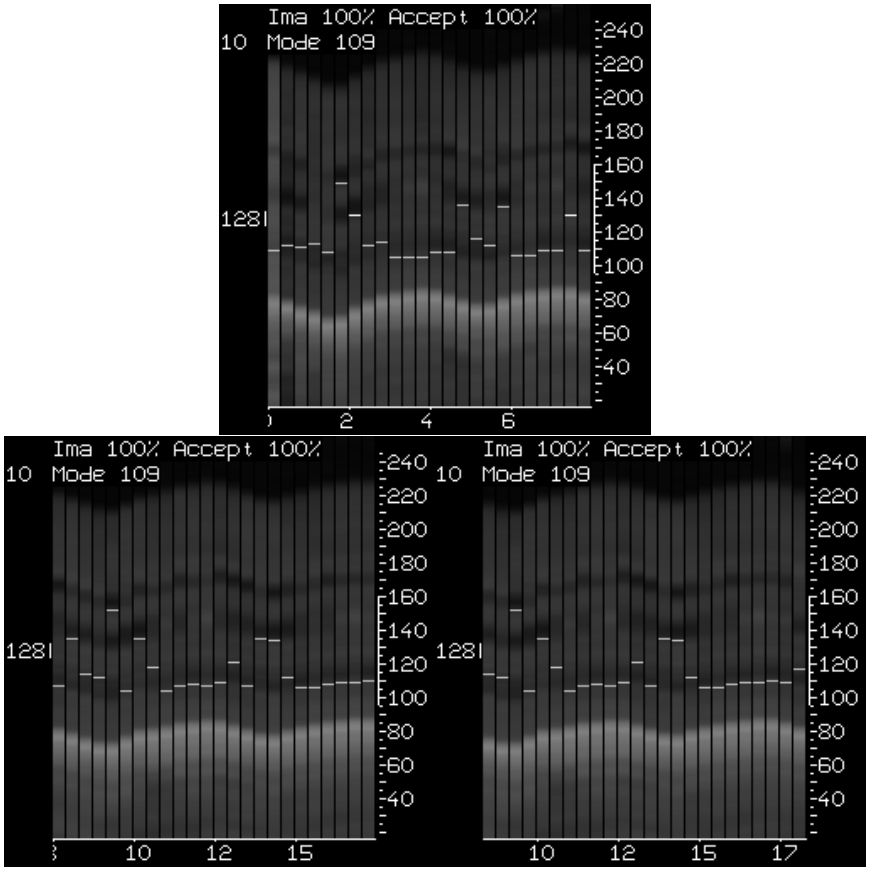

Figure 3.2: Navigator images for slice #4. Three images are needed for each slice because one navigator image contains postion information for 24 frames and our data set contains 50 frames for every slice. It can be noted that most of the information in the third image also appears in the second image.

3.2 Frame-by-frame segmentation

This section explains the work done to develop a correct segmentation of sagittal images. Please refer to B.1.1 for the corresponding code structure.

The objective is not to obtain the most refined and exact segmentation, especially given the fact that the heart is beating as well as moving with respiration, and therefore we are only looking for a reasonably easy to carry out and reasonably accurate method. The algorithm used was based on a snake algorithm, so the reader who happens not to be familiar with the theory of active contour snake algorithms is encouraged to go through the theory in 2.1 and 2.2before following any further.

In order to make the segmetation process easier, and because the images are noisy and full of arctifacts, an external software called *Seg3D* was used to create masks of every heart [49]. When the snake algorithm was executed, a mask on top of the heart prevented the snake from getting stuck in parts of the image which were not important. Please refer to A.1 for more information about the organization of image and mask files.

3.2.1 Snake Algorithm Implementation

Figure 3.3 shows how an arbitrary number of points was manually selected around the heart of the first frame for every slice.

Figure 3.3: Points selected by the user

The active contour moves in every iteration, under control of the snake algorithm described earlier, until it surrounds the heart, as seen in Figure 3.4. The algorithm stops when an arbitrary maximum number of iterations is reached, or when the movement of the points is smaller than a fixed treshold value. The length and the center of every curve are saved for latter use. Please refer to A.1 for more information about the final data matrix.

Figure 3.4: Boundary detected by the algorithm

3.2.2 Heart center

It will turn out to be very important to define a global center of the heart for latter operations. A center for the heart in each frame is found after segmentation, and the center of a slice will then be taken as the average of the centers of all its frames. Following the same idea, the average of the centers of all slices will be defined as the global center of the heart.

Two algorithms were developed to find the center of a frame, based on the use of an arclength dl:

- Linear dl:

    1. Get the set of points that create the boundary of the heart in one frame and put them in the rows of a matrix.

2. For each point, find the previous and the next point, which are located in the previous and following rows of the data matrix.
3. Calculate the length of the segments between each point and its previous and next points.
4. For each point, add both lengths.
5. Assign as dl of each point half of the corresponding value obtained in the previous step.
6. Get the contribution of each point by multiplying its x and y coorinates by its dl.
7. Add up all the contributions and divide by the sum of all dl.

- 2nd order dl:

1. Get the set of points that create the boundary of the heart in one frame and put them in the rows of a matrix.
2. For each point, find the previous and the next point, which are located in the previous and following rows of the data matrix.
3. Calculate the arclength of an interpolated second order polynomial that intersects with these three points.
4. Assign as dl of each point half of the corresponding value obtained in the previous step.
5. Get the contribution of each point by multiplying its x and y coorinates by its dl.
6. Add up all the contributions and divide by the sum of all dl. The sum of all dl is considered as the length of the curve and is saved for latter use.

The first method was used because it is faster, easier to implement and results are approximately the same due to the proximity between data points. Added to this, the second algorithm has to take into account special cases when the three points are arranged in a way in which they can not fit a second degree polynomial.

3.2.3 Base functions of the heart boundaries using the SVD

This objective of this subsection is to use the SVD decomposition of the data to represent all the boundaries as products of a reduced number of base functions and coefficients. At this point there is a curve that defines every boundary of the heart in every frame of each slice, which in general will produce a large number of curves. In our case, 800 such curves were produced.

The following operations were individually applied to each frame:

1. Subtract the frame center from all the points, so the curve is centered in the origin of coordinates.

2. Change from Cartesian to polar coordinates, so the curve is represented as a function of radius r and angle θ.

3. Interpolate a point every x degrees, so that points along the curve are equally spaced. In practice, we have used $x = 5º$.

4. Normalize all the radii from 0 to 1, dividing each radius by the largest radius over that frame. This value is saved for latter use.

The following operations were globally applied to each slice:

1. Find the average curve. It is found using a simple averaging method, thanks to the fact that all the curves are equally sampled, have a normalized radii and are expressed in polar coordinates. Every x degrees, the corresponding value of the average curve is the average of the values of all the curves at that angle.

2. Change to Cartesian coordinates, so the curves are represented as a function of x and y.

3. Subtract the average curve from all the curves, so there is no translational offset from the origin affecting the SVD calculation.

4. Multiply each curve by its length, obtained when calculating the center of the heart, in order to give the appropriate weight to every boundary.

5. Run the SVD on the data matrix saved after the segmentation. There is a huge amount of data for every slice, determined by the number of frames, and at the same time by the points in each frame. In practice, the data matrix on which the SVD is run has a size of 2x5000. The execution of the SVD on the data matrix should show that not all the saved data is needed to represent the boundaries of the heart for each slice, so a reduced data matrix could be used in the following sections. Before running the SVD all the points in the data matrix have been organized as follows:

$$M = \begin{bmatrix} x_{p_1 f_1} & x_{p_1 f_2} & \cdots & x_{p_1 f_P} \\ y_{p_1 f_1} & y_{p_1 f_2} & \cdots & y_{p_1 f_P} \\ x_{p_2 f_1} & x_{p_2 f_2} & \cdots & x_{p_2 f_P} \\ y_{p_2 f_1} & y_{p_2 f_2} & \cdots & y_{p_2 f_P} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p_N f_1} & x_{p_N f_2} & \cdots & x_{p_N f_P} \\ y_{p_N f_1} & y_{p_N f_2} & \cdots & y_{p_N f_P} \end{bmatrix}$$

where N is the number of points in a frame and P the number of frames in a slice, and $x_{p_r f_s}$ represents the x coordinate of the point r in frame s.

Figure 3.5 shows the the contribution of the base functions obtained after running the SVD on the data matrix. It can be noted that the contribution of the first 10 base functions gives 90% of the information needed. In other words, the combination of 10 curves is enough to represent all the frames of one slice, instead of the original 50 curves.

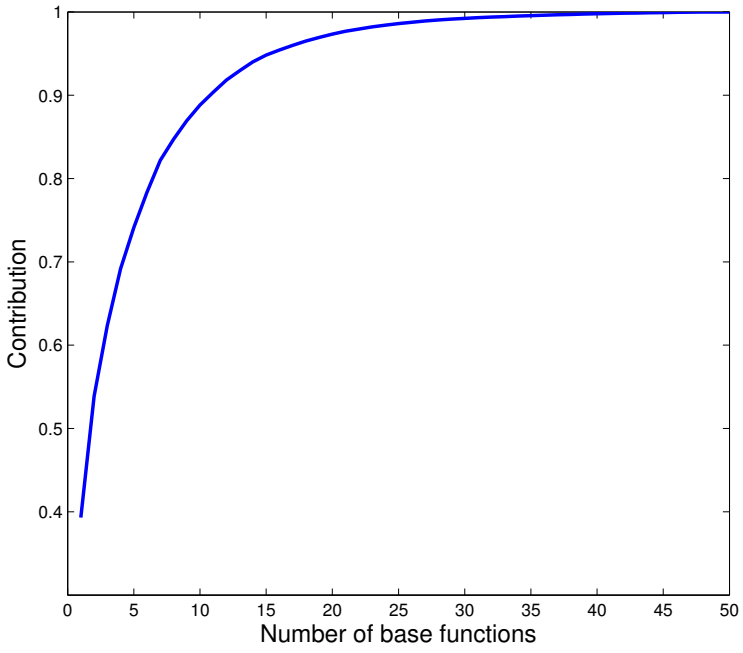

Figure 3.5: Contribution of the base functions obtained for one slice. It can be noted that the first 10 curves give 90% of the information needed, so their combination is enough to represent all the frames of one slice, instead of using 50 curves for one slice.

3.2.4 Inverse SVD

The objective of this subsection is to obtain a set of curves which are a filtered approximation of the original ones, using the set of base functions and coefficients found in the previous subsection. This will allow to work with a considerable smaller amount of data in the following sections.

- The following operations were globally applied to each slice:

1. Use the matrices that have resulted in the SVD execution to obtain the new points. These points are rearranged in a new structure, so a matrix is built for each frame. For $F_1 = \begin{bmatrix} x_{p_1 f_1} & x_{p_2 f_1} & \cdots & x_{p_N f_1} \\ y_{p_1 f_1} & y_{p_2 f_1} & \cdots & y_{p_N f_1} \end{bmatrix}$ where $x_{p_r f_s}$ represents the x coordinate of the point r in frame s. Then, every matrix is stored in an element of a cell array, which defines the data matrix for that specific slice $S_1 = \begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1P} \end{bmatrix}$, where S_q is the cell array for slice q and F_{qs} is the data matrix for frame s of slice q.

2. Divide each curve by its length, to remove weights given to the curves.

3. Add the average curve of each slice to all the curves from that slice, to recover their original shape.

- The following operations were individually applied to each frame:

1. Change to polar coordinates, so the curves are represented as a function of radius r and angle θ.
2. De-normalize all the radii from the interval 0 to 1, using the original radii saved when calculating the center of the heart.
3. Change to Cartesian coordinates, so the curves are represented as a function of x and y.
4. Add the heart center of the frame to all the points, so the curves are centered in the original center.

Both Figure 3.6 and Figure 3.7 show the results of this algorithm for two different frames of two different slices.

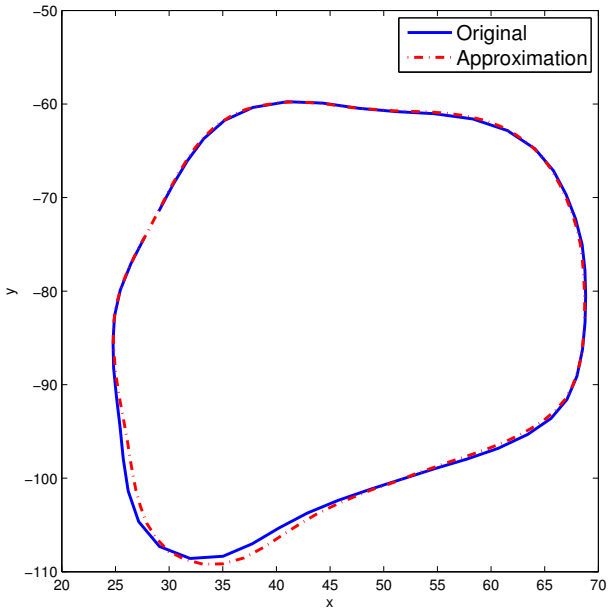

Figure 3.6: Comparison between an original curve and its approximation with base functions, obtained after running the SVD on the data matrix of one frame

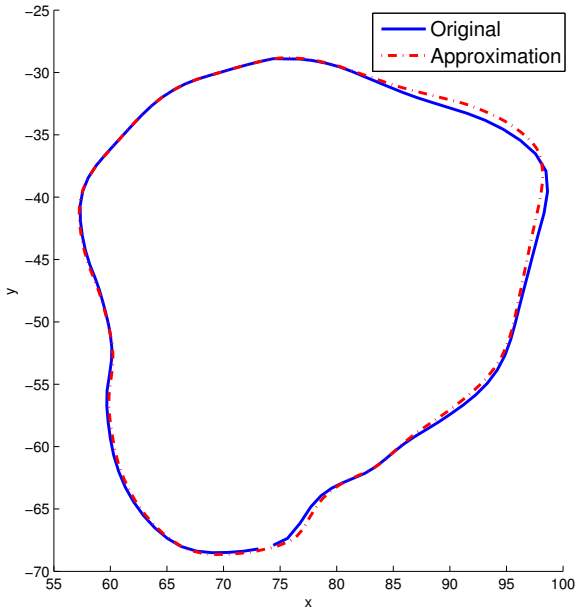

Figure 3.7: Comparison between an original curve and its approximation with base functions, obtained after running the SVD on the data matrix of one frame

3.3 Detection of the respiratory phase

This section explains how navigator images were used to detect a respiratory level for every frame of each slice. These images, as described above, show a series of levels, which are an indicator of the position of a specific point inside the body. Please refer to B.1.1 for the corresponding code structure.

There are 16 slices of the heart and there is a set of navigator images for each slice. A set of navigator images consists of 3 images, showing the levels from frame 1 to frame 50. Some levels appear in two images, so it is important to know which are the levels that must be detected and saved, as described in 3.1.2. Please refer to A.2 for more information about the organization of image files.

3.3.1 Level detection

The objective of the following algorithm is to detect and save postion information about the respiratory phase value for each frame, based on the navigator images. The structure of the algorithm is as follows:

1. Apply a Prewitt edge filter on the navigator images. As a result, the top of the level that needs to be detected turns out to be a darker zone in the image, due to

a higher change in the gradient. This effect can be observed in both Figure 3.8 and Figure 3.9

2. Detect the levels in the filtered images, manually setting a lowest and highest limit on the image, as shown in Figure 3.8 and Figure 3.9. A recursive search compares the values of the pixels in the set interval, and saves the position of the pixel whose value has the highest difference with the following pixel.

3. Normalize all levels of all slices between 0 and 1. Value 0 is assigned to the lowest level and 1 to the highest level.

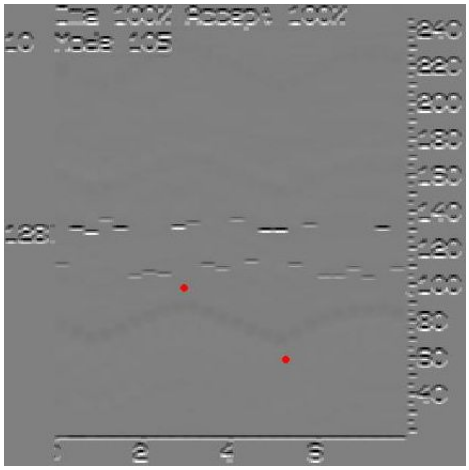

Figure 3.8: Selection of lowest and highest search values

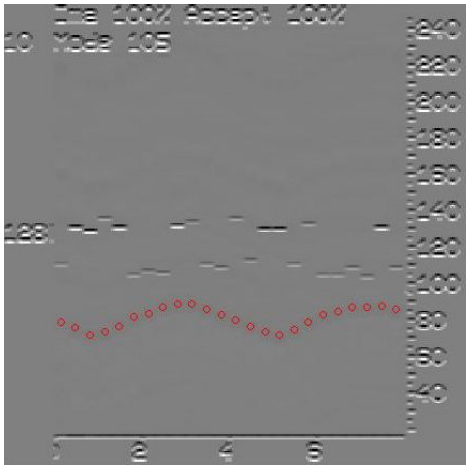

Figure 3.9: Detection of levels

3.4 Relationship between Segmentation and Phase Detection

This section describes how a relationship was established between the sagittal data set and the navigator data set. The contents that follow can be considered as an inflection point in our exposition, because they draw some conclusions which will be very important in the following chapters. Please refer to B.1.1 for the corresponding code structure.

Taking a look at both data sets at the same time, it can be seen that each level detected in the navigator has a correspondence in the sagittal image taken at the same time. In other words, the relative position of the heart in the sagittal images follows the level variations shown in the navigator images.

From now on, the analysis of the respiratory cycle is the main goal. It can be observed that, although there are levels which appear more than once in the same slice's set of navigator images, they take part in different parts of the respiratory cycle. That is, the same level can be sapled during inspiration or during expiration. For this reason, there is a need to modify the vector of levels. The objective is to maintain the same order, in order to avoid losing the relationship of each level with its corresponding sagittal image. At the same time, though, there has to be a distinction between those levels that appear during inspiration and those which appear during expiration.

The solution consists of normalizing the values between -1 and 1. Across each slice, level -1 is the maximum value for expiration and level 0 is the minimum, while level 0 is also the minimum value for inspiration and level 1 is the maximum. This is an easy way to differentiate levels, which brings up a debate on how to treat an interesting respiratory property: periodicity.

Until now, we have observed the respiratory cycle as a function of time, following an almost-periodic behavior. With the new arrangement of the levels, though, it is possible to visualize all the curves of a slice following the order of a respiratory cycle based on the value of the levels, instead of the order in time in which the images were taken. This is a first hint that it is necessary to neglect the time dimension and jump into respiratory-phase dimension. It seems a natural change now that levels are already ordered in a periodic fashion, and will turn out to be very useful in the following chapters.

3.4.1 Data analysis

Developing a change from time to respiratory phase (which we will denote as α in what follows) requires a deeper study of the periodic function of navigator levels in time. A second dimension is added to the data, which is the separation between levels, due to the fact that there are gaps along the horizontal axis between the phase points extracted from the navigator images.

As we have seen in 3.3, the magnitude of the phase is taken as the normalized height between 0 and 1 of the detected value from the navigator bar, while it is assigned a negative sign if it occurs during expiration or it has a positive sign if it occurs during inspiration. The sign is determined by looking at whether the first difference of the

detected points is positive or negative. In the unlikely event that the first derivative is zero, the sign is the same as that of the preceding frame. For representation purposes, points in expiration have a negative derivative and are plotted in red, while points in inspiration have a positive derivative and are plotted in blue. Those points which have a zero derivative are plotted in green, but for practical reasons they assume the derivative of the point located before them in the following algorithms.

We can then resort the frames from all respiratory cycles available for a given slice, which covers about 5-6 cycles for each frame as described in 3.1.2, and instead treat them as samples of a single cycle. We illustrate this process in Figure 3.10 and Figure 3.11. Figure 3.10 shows the normalized phase values as a function of horizontal position (pixel) in the navigator frame, with color used to denote expiration (red) or inspiration (blue), while Figure 3.11 shows the same values resorted into a single cycle. Note that in Figure 3.10 the vertical axis is phase, while the horizontal axis is, in effect, time, while in Figure 3.11 phase becomes the horizontal axis, i.e. phase is now the independent horizontal axis, and for convenience we emphasize the collapsing of several cycles into one by showing all points at the same height on the vertical axis.

To make things easier from now on, levels are organized in a row vector following the same order than their corresponding frames.

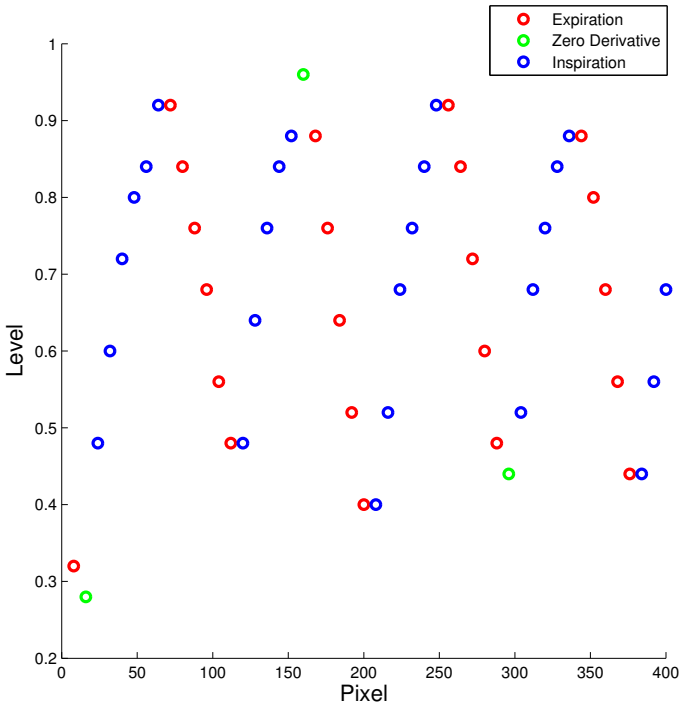

Figure 3.10: Normalized phase values as a function of horizontal position (pixel) in the navigator frame

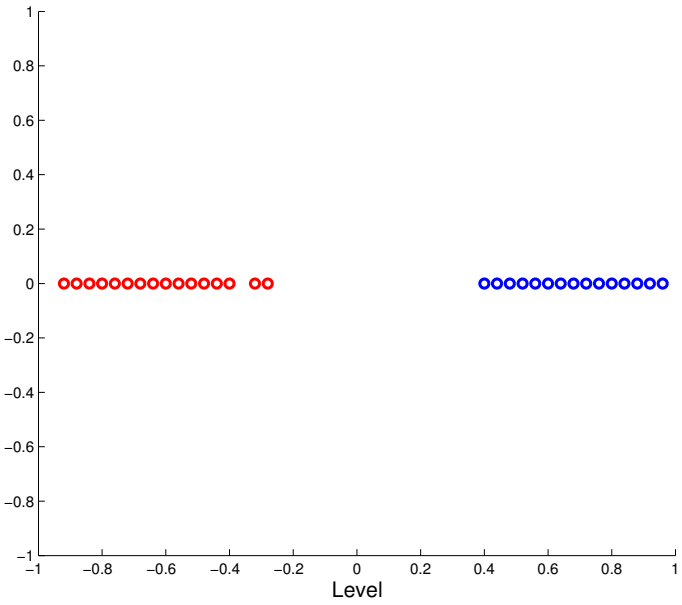

Figure 3.11: Normalized phase values resorted into a single cycle

3.5 Time and Spatial Interpolation

As stated in 1.5, one of our main objectives consisted in finding a method to interpolate frames and slices that don't exist in the real data set. This section goes through different techniques that were developed to achieve a good interpolation of the human heart. The first methods are simpler and straight to the point, while the latter methods are more complex and challenging. Filtering and interpolation algorithms are involved, so the reader is encouraged to go through the theory in 2.2, 2.3, 2.4 and 2.5 before following any further.

3.5.1 Level Interpolation in the Respiratory Function

To have a better representation of the almost sinusoidal character of the respiratory cycle, a denser grid of points can be interpolated based on the original levels. The first interpolation algorithm was based on a simple convex interpolation

$$p_{new} = d p_0 + (1-d) p_1$$

where d is the distance between two points p_0 and p_1. Please refer to B.2.1 for the corresponding code structure.

Figure 3.12 shows that due to the fact that respiration is not truly sinusoidal, in addition to the fact that we have noisy data and the beating of the heart, the obtained function was far from being a pure sinusoidal.

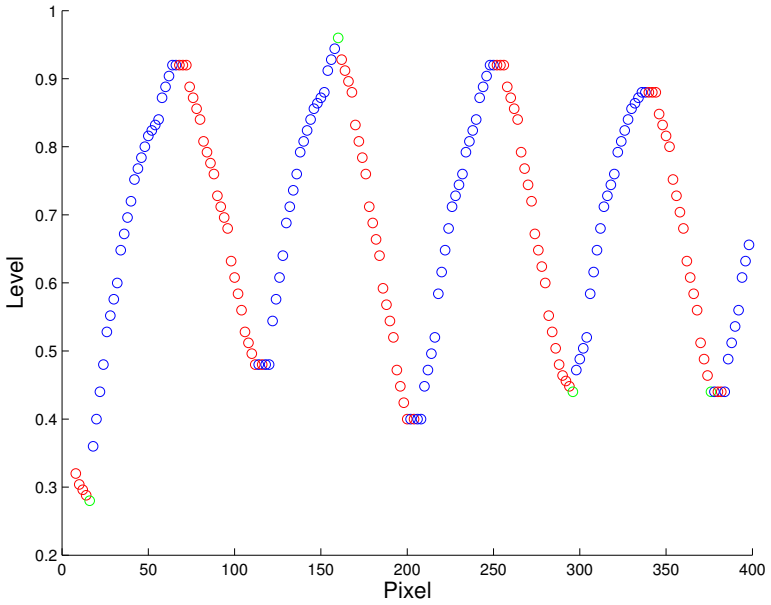

Figure 3.12: Denser grid of 5 new levels between the original values

3.5.2 Curve Fitting

Curve fitting was applied as the second interpolation algorithm and did not use any of the results obtained to the first method. It was also the first attempt to filter the level values, based on fitting the data into different kinds of functions and trying to determine which was the best way to obtain a smoother respiratory function. The main objective was not to obtain a final solution, but to get a first glimpse of the behavior of the respiratory levels. Please refer to B.2.2 for the corresponding code structure.

Our first approach was to use a polynomial to fit the data. Figure 3.13 shows that a second order polynomial was not able to follow the respiratory levels. A third order polynomial gave a better result, as seen in Figure 3.14.

Results for a third order polynomial fitting were misleading, in the sense that it followed very well the evolution of the points, without taking into account that the ideal result should be a smooth and periodic function. More complex fittings were used to obtain smoother results, such as a third order Fourier fitting shown in Figure 3.15. In the end, we decided that in case we decided to use this method the Fourier fitting was the most appropriate, but we saw that there was a need to develop a more complex method that could model the data better.

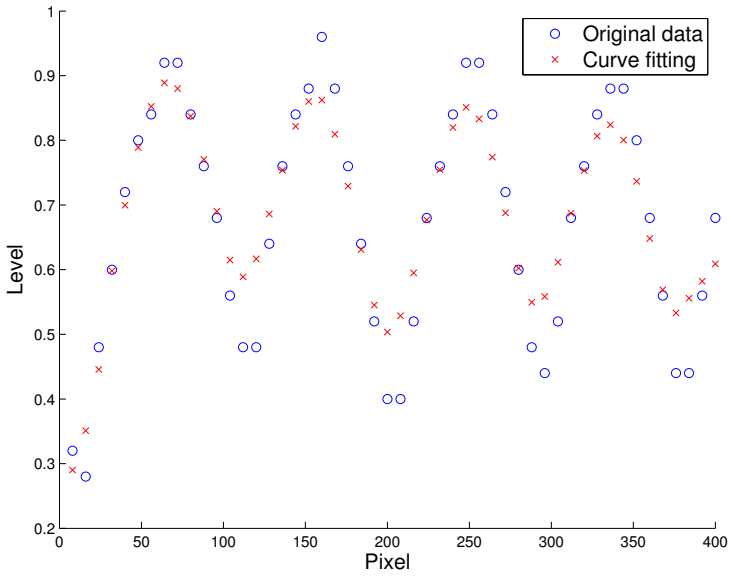

Figure 3.13: Second order polynomial fitting

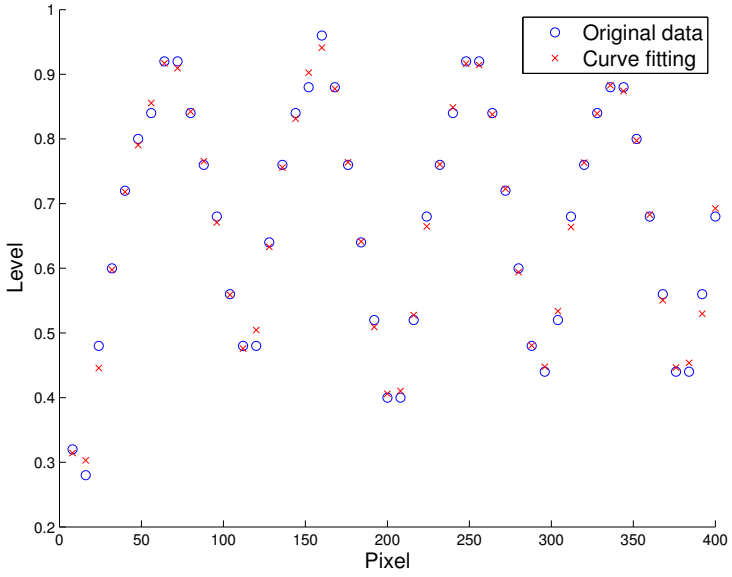

Figure 3.14: Third order polynomial fitting

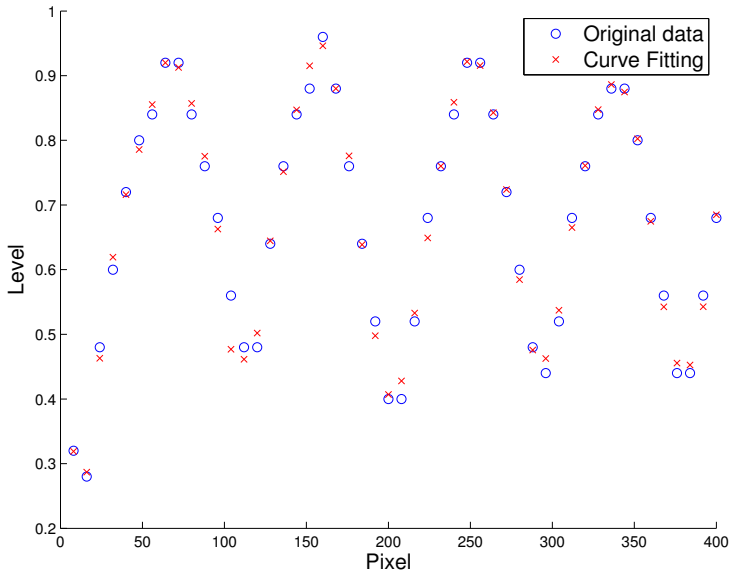

Figure 3.15: Third order Fourier fitting

3.5.3 Convex Interpolation with Curves

Until this point, we had interpolated over α and obtaining new values, but without any idea of how they would translate into a curve of the heart. Convex interpolation with curves was the first method developed to obtain a reasonable approximation of unknown respiratory levels, interpolating over the 3D spatial dimension. In other words, the points used in the interpolation algorithm were those which constituted the curve. For this reason, an important visualization change was introduced, because the interpolated heart curve associated to each new interpolated level could be plotted. Please refer to B.2.4 for the corresponding code structure.

Also, it can be noted that until the previous subsection the interpolated and real data sets were represented in 2D. The following subsections work on 3D data, which structure is detailed in A.3. Please refer to B.2.3 for the corresponding code structure.

The algorithm is based on a simple convex interpolation with single points of two curves

$$p_{new} = d p_0 + (1-d) p_1$$

where d is the distance between two points p_0 and p_1. In order to find p_{new}, weight is given to both points as a function of the distance, so if a point is closer to the new value it gets more weight than the other, and vice-versa. To obtain a new α, there is a need to find those existing values which are closest to the desired new value and associate them with their curves. Then, the interpolation can be done between every point p_0 of the first curve, and the corresponding point p_1 of the second curve.

Some special cases or exceptions were taken into consideration:

- If a level that is meant to be interpolated already exists, the algorithm doesn't run and shows the existing curve.
- If a level that is meant to be interpolated is at the end or the beginning of the respiratory cycle, i.e values close to -1 or close to 1, the algorithm would only have one value to calculate the interpolation. For this reason, it takes advantage of the periodicity of the respiratory cycle. That is, in these specific situation the following or previous level is used, even if it doesn't belong to the same part of the cycle. In other words, a value higher than the highest positive value, which is at the end of inspiration, uses the highest negative value which is at the beginning of expiration, and vice-versa.

There are three interpolation possibilities listed below. It is very important to correctly plot the results in order to obtain the desired information.

1. Figure 3.16 shows interpolation of a single unknown curve given a single new level, for a specific slice.
2. Figure 3.17 shows interpolation of an interval of unknown curves given an interval of new levels, for a specific slice.
3. Figure 3.18 shows interpolation of a single unknown curve given a single new level, for all slices. Only the interpolated curve is shown, lower and higher curves are not plotted to avoid too many curves in a 3D visualization.

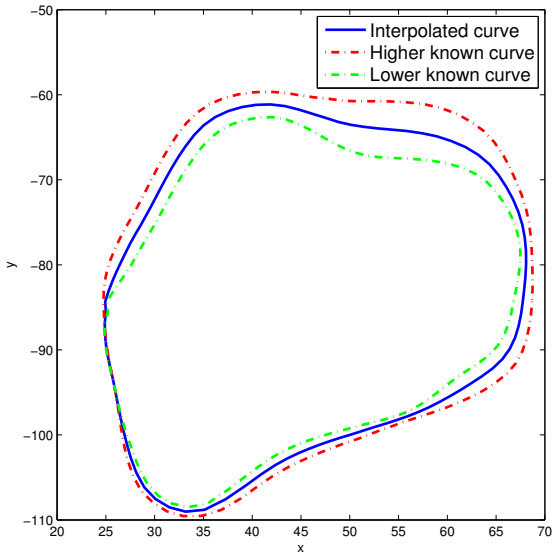

Figure 3.16: Interpolation of a new curve for a specific slice

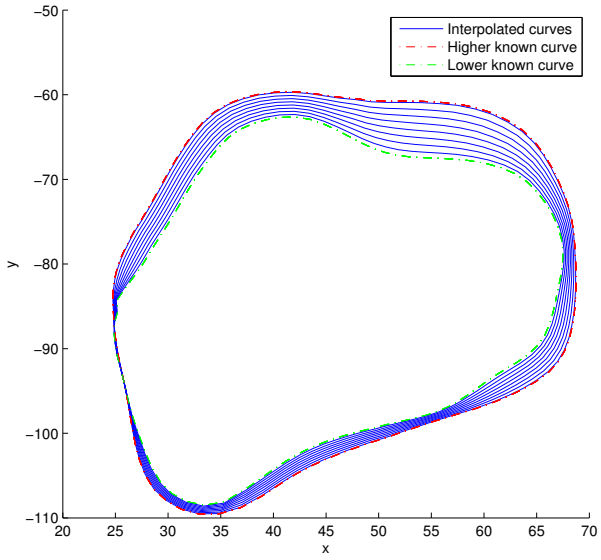

Figure 3.17: Interpolation of an interval of new curves for a specific slice

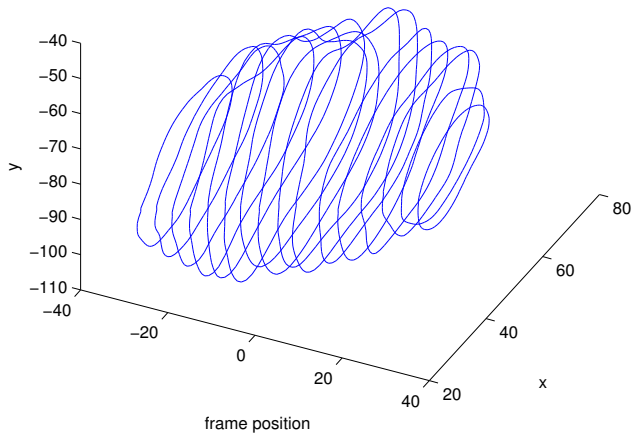

Figure 3.18: Interpolation of a single new curve for all slices

3.5.4 2D Fourier Filtering and Interpolation

At this point there was a need to step up in the complexity of interpolation methods and use a more elaborate procedure. Convex interpolation was a very robust and easy-to-impement method, but it only gave results based on a simple weighting of curves.

As an alternative, the method described in this subsection interpolated new points based on using a Fast Fourier Transformation on two dimensions at the same time. In other words, it interpolated new radii taking into account both the angle position and the position in respiratory phase. The algorithm was applied as follows. Please refer to B.2.5 for the corresponding code structure.

- Change the points of all the curves from cartesian coordinates to polar coordinates, so the curves are represented as a function of radius r and angle θ.
- Build a matrix A which contains base functions for the respiratory phase dimension α.

$$A = \begin{bmatrix} 1 & \sin\pi\alpha_1 & \cos\pi\alpha_1 & \sin 2\pi\alpha_1 & \cos 2\pi\alpha_1 & \cdots & \sin m\pi\alpha_1 & \cos m\pi\alpha_1 \\ 1 & \sin\pi\alpha_2 & \cos\pi\alpha_2 & \sin 2\pi\alpha_2 & \cos 2\pi\alpha_2 & \cdots & \sin m\pi\alpha_2 & \cos m\pi\alpha_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \sin\pi\alpha_n & \cos\pi\alpha_n & \sin 2\pi\alpha_n & \cos 2\pi\alpha_n & \cdots & \sin m\pi\alpha_n & \cos m\pi\alpha_n \end{bmatrix}$$

where α_j is the respiratory phase value for level j, n is the number of respiratory levels and m is the number of desired complex harmonics. In practice, this matrix is build with 21 respiratory levels and 1 or 2 complex harmonics, so it has a size of 21x3 or 21x5.

- Build a matrix Y whose rows contain the FFT of every curve as a function of θ. In practice, only a specific number of relevant harmonics p is stored.

$$Y = FFT(M) = \begin{pmatrix} FFT(c_1) \\ FFT(c_2) \\ \vdots \\ FFT(c_n) \end{pmatrix}$$

- Find a matrix X in the α dimension which, in combination with the base functions of the respiratory phase, results in the FFT obtained in the previous step. In other words, a relatioship is established between the α dimension and the θ dimension.

If

$$AX = Y$$

then

$$X = A^{+}Y$$

where A^{+} is the generalized inverse of A.

- Calculate the desired new curve $\mathbf{c_{new}}$ for a new α value. Filtering applies when values that already existed are used to obtain curves, while filtering and interpolation applies when unknown values are used to obtain new curves.

$$\mathbf{c}_{new} = \mathbf{a}X\mathrm{B}'$$

where

$$\mathbf{a} = \begin{bmatrix} 1 & \sin\pi\alpha_{new} & \cos\pi\alpha_{new} & \sin 2\pi\alpha_{new} & \cos 2\pi\alpha_{new} & \cdots & \sin m\pi\alpha_{new} \cos m\pi\alpha_{new} \end{bmatrix}$$

and

$$B = IFFT(I)$$

which contains the base functions for the Inverse Fast Fourier Transformation. There are two different ways to represent the interpolated curves:

1. Figure 3.19 shows how a value that already existed in the original data is filtered into a smoother curve.

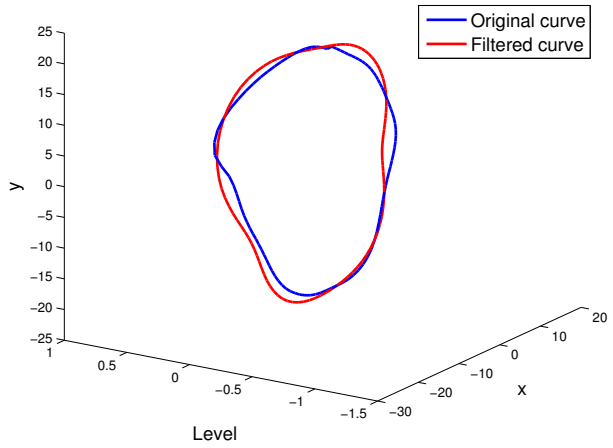

Figure 3.19: Original and filtered curves for a single level, obtained using $m = 1$ harmonics in α and $p = 25$ harmonics of the FFT. Each curve contains 100 angle samples

2. Figure3.20 shows an equally spaced grid of interpolated curves. Figure 3.21 and 3.22 show that if a broad interval of new levels is plotted, the combination of the contributions of beating and respiration can be clearly identified.

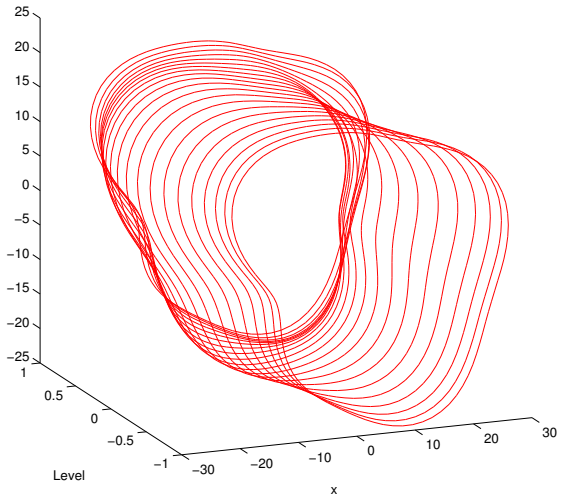

Figure 3.20: Interpolated curves for an equally spaced grid from -1 to 1

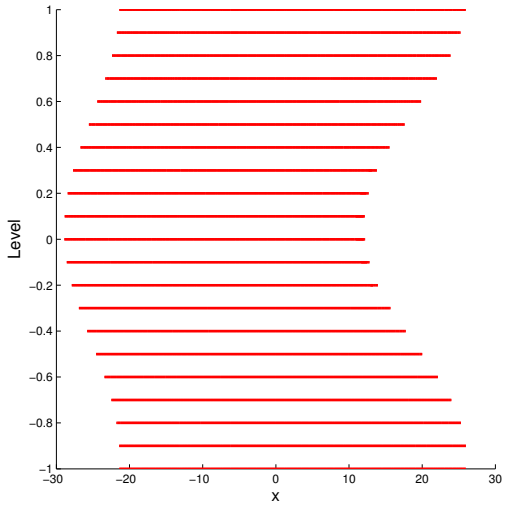

Figure 3.21: First view of the movement of the heart, which corresponds to Figure 3.20 observed from the top

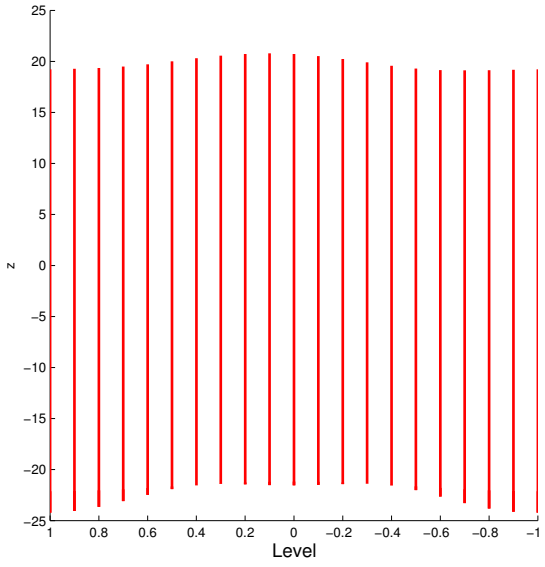

Figure 3.22: Second view of the movement of the heart, which corresponds to Figure 3.20 observed from the side

Compared to the previous methods which needed to run through the whole process for each interpolated curve, it can be observed that this method allows working with all the data of a slice at the same time. The use of FFT coefficients makes interpolation much faster and easy to calculate. In addition, filtering is implicit and the original curves are smoothed due to the limited number of coefficients and FFT harmonics.

3.5.5 3D Fourier Filtering and Interpolation

After trying to solve the interpolation problem in 2D, the natural tendency was trying to evolve the same idea into a 3D FFT environment. This new step gave the possibility of plotting a global heart for each new value of α. In other words, it allowed working with all the data of all slices at the same time. The algorithm is based on the steps used in the previous subsection using the 2D FFT, and it is given in detail as follows. Please refer to B.2.6 for the corresponding code structure.

- Change the points of all the curves from Cartesian coordinates to spherical coordinates, so the curves are represented as a function of radius r and two angles θ, φ.
- Build a matrix A which contains base functions for the respiratory phase dimension α.

$$A = \begin{bmatrix} 1 & \sin\pi\alpha_1 & \cos\pi\alpha_1 & \sin 2\pi\alpha_1 & \cos 2\pi\alpha_1 & \cdots & \sin m\pi\alpha_1 & \cos m\pi\alpha_1 \\ 1 & \sin\pi\alpha_2 & \cos\pi\alpha_2 & \sin 2\pi\alpha_2 & \cos 2\pi\alpha_2 & \cdots & \sin m\pi\alpha_2 & \cos m\pi\alpha_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \sin\pi\alpha_n & \cos\pi\alpha_n & \sin 2\pi\alpha_n & \cos 2\pi\alpha_n & \cdots & \sin m\pi\alpha_n & \cos m\pi\alpha_n \end{bmatrix}$$

where n is the number of respiratory levels and m is the number of desired complex harmonics.

- Build a matrix P which contains base functions for the first angle dimension φ.

$$P = \begin{bmatrix} 1 & \sin\pi\varphi_1 & \cos\pi\theta & \sin 2\pi\theta_1 & \cos 2\pi\theta_1 & \cdots & \sin p\pi\theta_1 & \cos p\pi\theta_1 \\ 1 & \sin\pi\theta_2 & \cos\pi\theta_2 & \sin 2\pi\theta_2 & \cos 2\pi\theta_2 & \cdots & \sin p\pi\theta_2 & \cos p\pi\theta_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \sin\pi\theta_q & \cos\pi\theta_q & \sin 2\pi\theta_q & \cos 2\pi\theta_q & \cdots & \sin p\pi\theta_q & \cos p\pi\theta_q \end{bmatrix}$$

where q is the number of values available and p is the number of desired complex harmonics in the φ dimension.

- Build a matrix T which contains base functions for the second angle dimension θ.

$$T = \begin{bmatrix} 1 & \sin\pi\theta_1 & \cos\pi\theta_1 & \sin 2\pi\theta_1 & \cos 2\pi\theta_1 & \cdots & \sin t\pi\theta_1 & \cos mt\theta_1 \\ 1 & \sin\pi\theta_2 & \cos\pi\theta_2 & \sin 2\pi\theta_2 & \cos 2\pi\theta_2 & \cdots & \sin t\pi\theta_2 & \cos t\pi\theta_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \sin\pi\theta_q & \cos\pi\theta_q & \sin 2\pi\theta_q & \cos 2\pi\theta_q & \cdots & \sin t\pi\theta_q & \cos t\pi\theta_q \end{bmatrix}$$

where q is the number of values available and t is the number of desired complex harmonics in the θ dimension. The value q is the same for P and T because when data points are changed to spherical coordinates, we use the same ammount of values in θ and φ.

- Build a matrix M which combines all base functions contained in the three previous matrices, as detailed below, having in mind that each θ has a corresponding φ. At the same time, each α has a corresponding set of θ and φ values, depending on how many points define each heart.

The following steps were taken in order to obtain interpolation coefficients, which allowed us to produced a filtered and interpolated 3D heart for a specific α value. Because of the huge dimensions of the data matrix M, we took advantage of the SVD properties given in 2.2 to perform calculations using the correlation matrix C in the smaller dimension (α).

- Calculate SVD of the correlation matrix

$$C = MM^*$$

where

$$M = USV^*$$

and

$$SVD(C) = U_C S_C V_C^* = US^2 V_C^*$$

- Save matrix U of singular vectors and matrix S of squared singular values

$$S = S_C^{\frac{1}{2}}$$

- Obtain V_M matrix of singular vectors using the previous results. Based on the definition

$$M = USV^*$$

then

$$V^* = S^{-1}U^*M$$

- Obtain the coefficients that will produce the filtered and interpolated data, which means that

$$\mathbf{r}' = M^*\mathbf{a}$$

then

$$\mathbf{a}' = \mathbf{r}VS^{-1}U^*$$

- Use an equally spaced grid of points, such as in Figure 3.23, to represent the interpolated points.

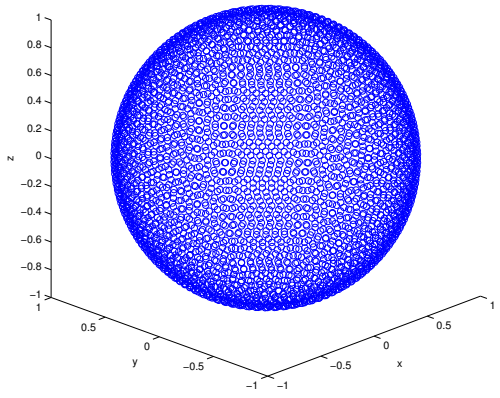

Figure 3.23: Equally spaced grid of points

Figure 3.24 shows that the results were a little bit disappointing, because the 3D heart didn't look close enough to a human heart and there were some convergence issues in the top and bottom parts. For this reason, we thought there was a need to find a better solution for the interpolation of a 4D model of the heart.

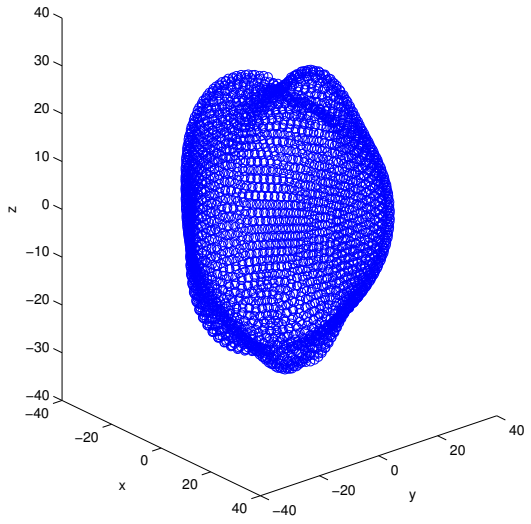

Figure 3.24: Interpolated heart for $\alpha = 0.5$, $n = 21$, $m = 2$, $p = 6$ and $t = 6$

3.5.6 Spherical Harmonics

The last method applied was based on the use of Legendre associated functions and spherical harmonics and was used to represent the heart and develop methods described the following chapters. It might seem that this method adds a new degree of complexity to the problem but, as a matter of fact, it turned out to be a better and faster solution than 3D Fourier filtering and interpolation. Please refer to B.2.7 for the corresponding code structure. The algorithm is as follows:

- Build a matrix A which contains base functions for the respiratory phase dimension α.

$$A = \begin{bmatrix} 1 & \sin\pi\alpha_1 & \cos\pi\alpha_1 & \sin 2\pi\alpha_1 & \cos 2\pi\alpha_1 & \cdots & \sin m\pi\alpha_1 & \cos m\pi\alpha_1 \\ 1 & \sin\pi\alpha_2 & \cos\pi\alpha_2 & \sin 2\pi\alpha_2 & \cos 2\pi\alpha_2 & \cdots & \sin m\pi\alpha_2 & \cos m\pi\alpha_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \sin\pi\alpha_n & \cos\pi\alpha_n & \sin 2\pi\alpha_n & \cos 2\pi\alpha_n & \cdots & \sin m\pi\alpha_n & \cos m\pi\alpha_n \end{bmatrix}$$

where n is the number of respiratory levels and m is the number of desired complex harmonics.

- Build a matrix S whose rows contain base functions in spherical harmonics, until a specified degree d and order l. These base functions represent the two remaining dimensions θ and φ.
- Build a matrix M which combines all base functions contained in the two previous matrices.
- Obtain the interpolation coefficients following the same procedure used in 3.5.5.
- Use an equally spaced grid of points, such as in Figure 3.23, to represent the interpolated points.

Figures 3.25 and 3.26 show the results for an interpolated heart using spherical harmonics. The small bulbs on top of the heart represent the beginning of the aorta and pulmonar arteries, as well as the end of the superior vena cava. They were included in the initial segmentation, so the interpolation model reproduced them too.

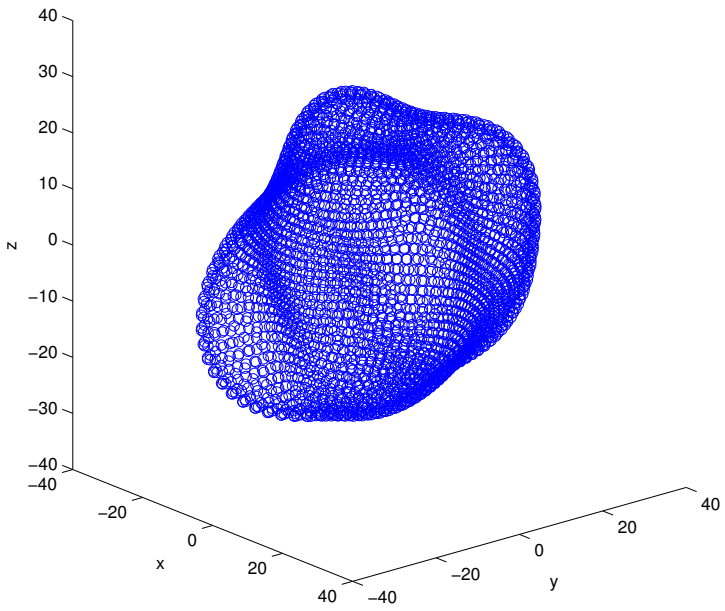

Figure 3.25: First view of the interpolated heart for $\alpha = 0.5$, $n = 21$, $m = 2$, $d = 6$ and $l = 6$

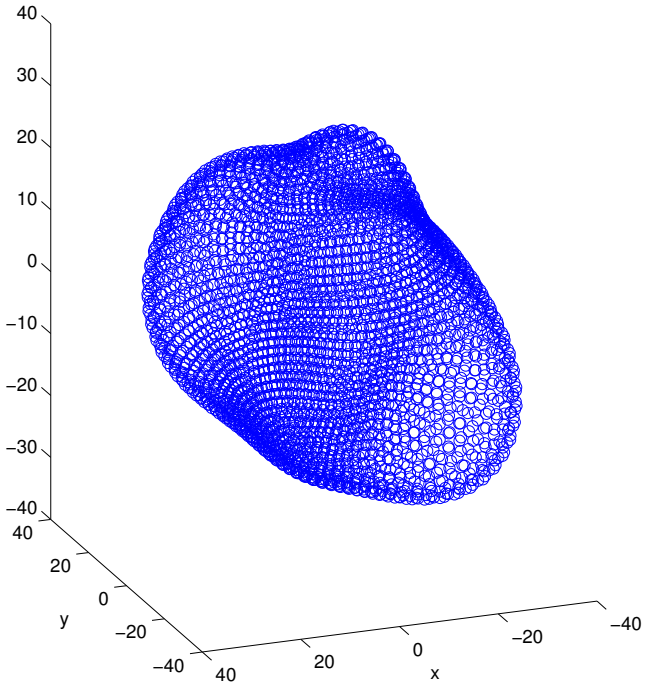

Figure 3.26: Second view of the interpolated heart for $\alpha = 0.5$, $n = 21$, $m = 2$, $d = 6$ and $l = 6$

Chapter 4

Construction and Sectioning of an Average Heart

This chapter explains how an average heart was obtained to supress any movement that not related to respiratory motion. The method used was based on the interpolated set of hearts that results from the method with spherical harmonics described in the previous chapter. Operations with rotation matrices were involved in the process, so the reader is encouraged to go through the theory in 2.6 before following any further. Please refer to B.3.1 for the corresponding code structure.

4.1 Data rotation

Before building an average heart it is important to reference all the interpolated hearts the same way. For this reason, the center of mass was subtracted from every heart, and the SVD was calculated to obtain three base vectors that define the principal directions of variation of the heart shapes. The centers of mass, singular vectors and singular values for each heart of one full respiratory cycle can be visualized. Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5 and Figure 4.6 give a sense of the heart movement and provied an easier understanding of the motion.

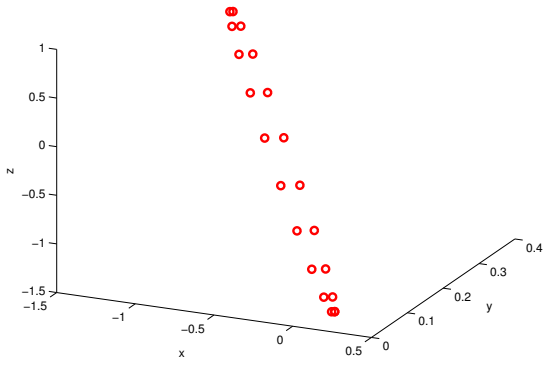

Figure 4.1: Values of the centers of mass for each heart of one full respiratory cycle

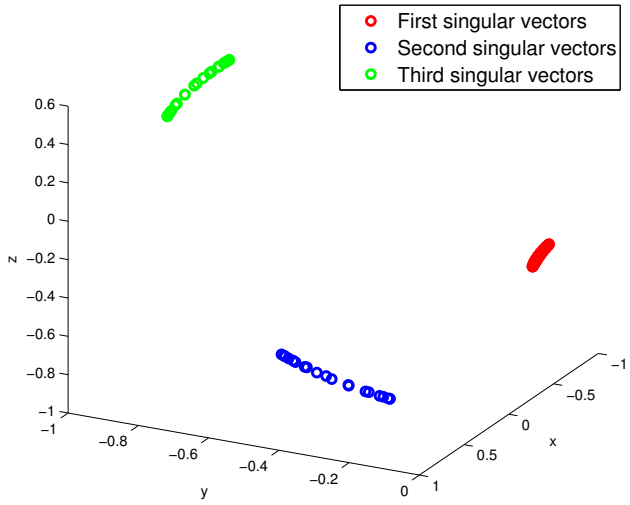

Figure 4.2: Location of the singular vectors for each heart of one full respiratory cycle

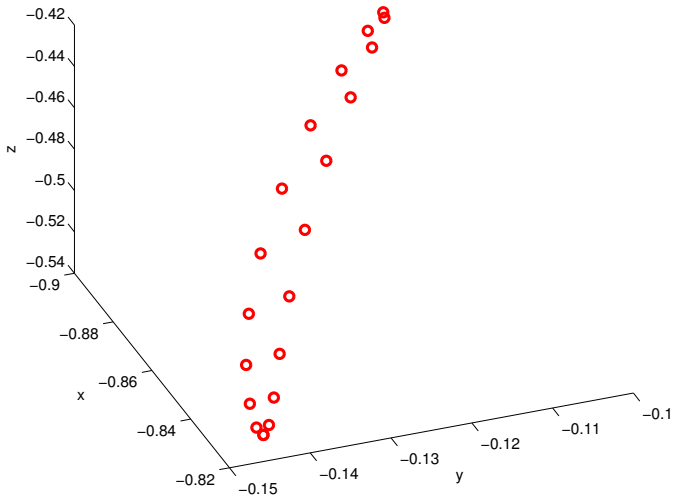

Figure 4.3: First singular vector for each heart of one full respiratory cycle, contained in the $-X, -Y, -Z$ quadrant

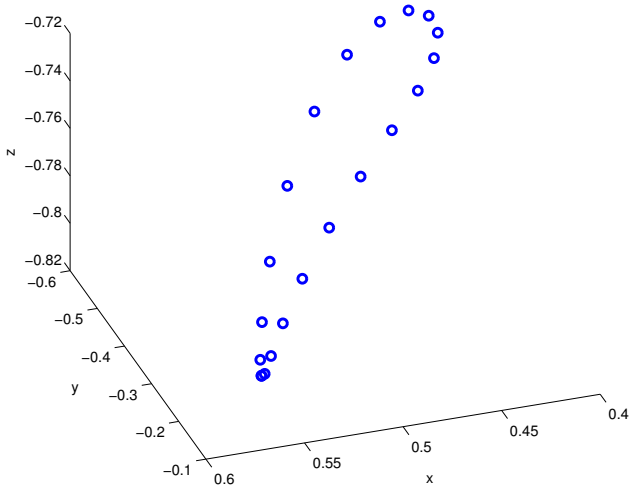

Figure 4.4: Second singular vector for each heart of one full respiratory cycle, contained in the $X, -Y, -Z$ quadrant

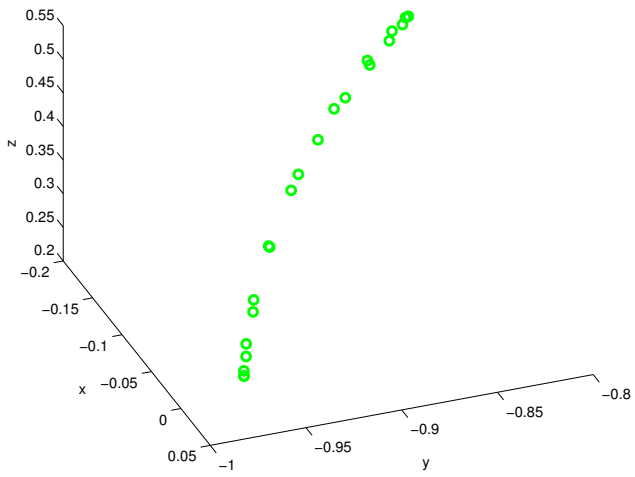

Figure 4.5: Third singular vector for each heart of one full respiratory cycle, contained in the $-X, -Y, Z$ quadrant

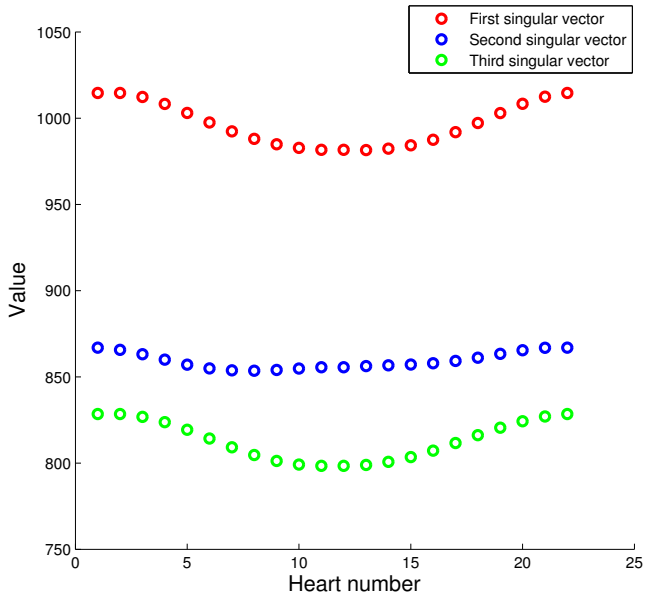

Figure 4.6: Values of the singular values for each heart of one full respiratory cycle

Once every heart was centered in the desired position, two rotations were applied. In every case:

1. The first and most important singular vector was aligned with a known typical XYZ axis. Matrix R_1 is obtained.

2. The second singular vector was rotated in a consistent way to one of the remaining two axes. Matrix R_2 is obtained.

3. The third was then automatically obtained, following the right-hand rule.

For example, the first singular vector can be moved to the $-Z$ axis, because it involves the minimum amount of rotation, as shown in 4.8. Then the second vector is rotated to the Y axis, for the same reason. In consequence, the third singular vector moves itself to stay aligned to the X axis, as seen in 4.8.

Rotating the data was performed by creating a global rotation matrix, the result of the multiplication of the two matrices R_1 and R_2 previously obtained, and then applying this rotation to every data matrix.

$$N = MR_1R_2 = MR$$

where M is a data matrix in the original coordinate system, N is the same data in the new coordinate system and R_1, R_2 are the rotation matrices.

Any information involved in each rotation, such as rotation angles and the center of mass, was saved for future use.

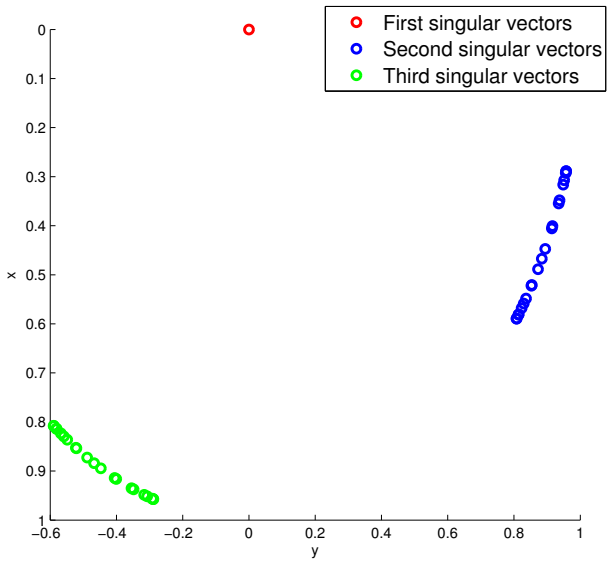

Figure 4.7: Result of the first rotation, where the first singular vector is alligned with the $-Z$ axis

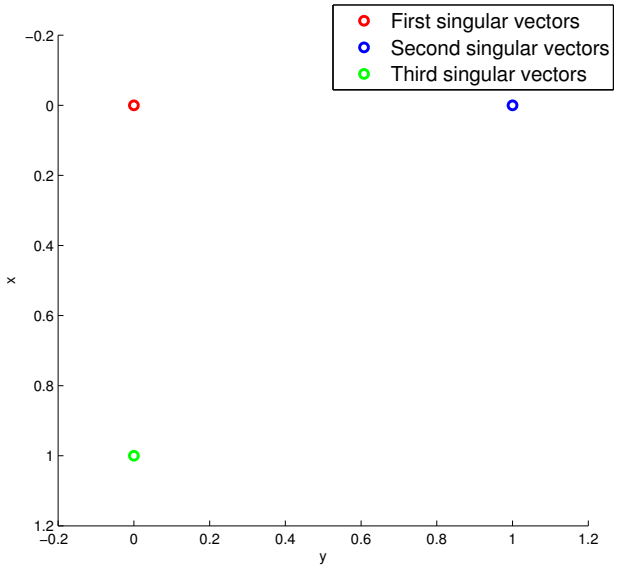

Figure 4.8: Result of the second rotation, where the second singular vector is alligned with the Y axis and the third singular vector is automatically alligned with the X axis

4.2 Average Heart

The average heart shown in Figure 4.9 was obtained in a very straightforward way after rotation, as the average of each point in every heart. The combination of rotating all hearts to the same position and then averaging them had a direct implication in the movements of the heart, because at least some of the variation due to beating was suppressed. Please refer to B.3.2 for the corresponding code structure.

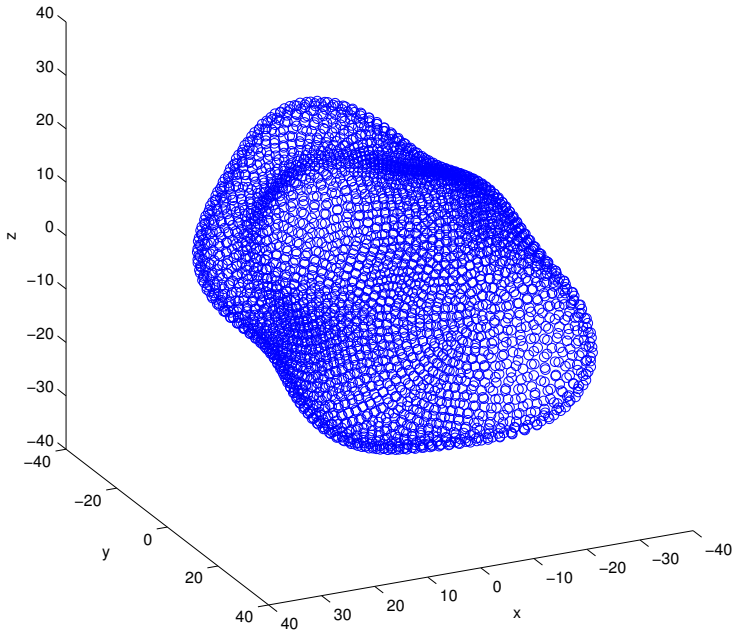

Figure 4.9: Average heart before undoing rotations

The next step consisted of undoing the rotations, although this time using the average heart, so the result was a set of hearts with the same shape but in different respiratory phases, such as the heart seen in Figure 4.10.

As a consequence, the only remaining visible movement was that which was caused by respiration, to the best of our ability to achieve this given the available data.

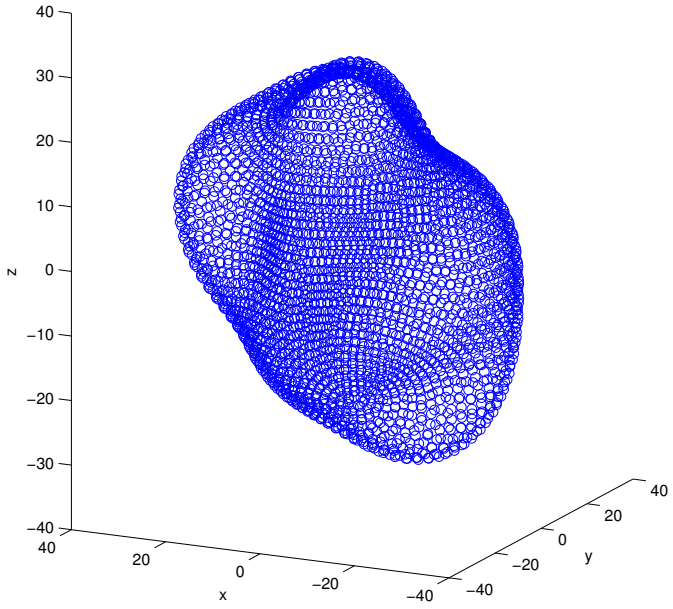

Figure 4.10: Average heart after undoing rotation for $\alpha = 0.5$

4.3 Heart Sectioning

This section explains a method developed in order to cut a heart with a plane and obtain the curve which is a result of their intersection. It is possible to use any plane in any interpolated heart but, in each case, the dimensions of the heart must be taken into account. The algorithm is as follows. Please refer to B.3.3 for the corresponding code structure.

- A vector $\overrightarrow{v_0}$ is selected, which goes from the origin to a point, and defines a plane normal to it at the tip of the vector. We note that any plane in $\mathbb{R}^3$ can be defined in this manner.
- A fixed grid of points is chosen in the φ dimension. To find the corresponding θ values of the points that are on the given plane, an exhaustive search over that grid is performed. The algorithm searches, for each φ, for a value $\theta \in [0, \pi]$ so that,

$$\langle \overrightarrow{v_p}, \overrightarrow{v_0} \rangle = \| \overrightarrow{v_0} \|$$

where $\overrightarrow{v_p}$ is the vector that goes from the origin to that point. In practice, this equals equation is defined as having the difference be below a threshold, which is set by the decimal precision of MATLAB.

- A set of points is obtained, which are on the plane or within a defined distance of it, as shown in Figure 4.11.

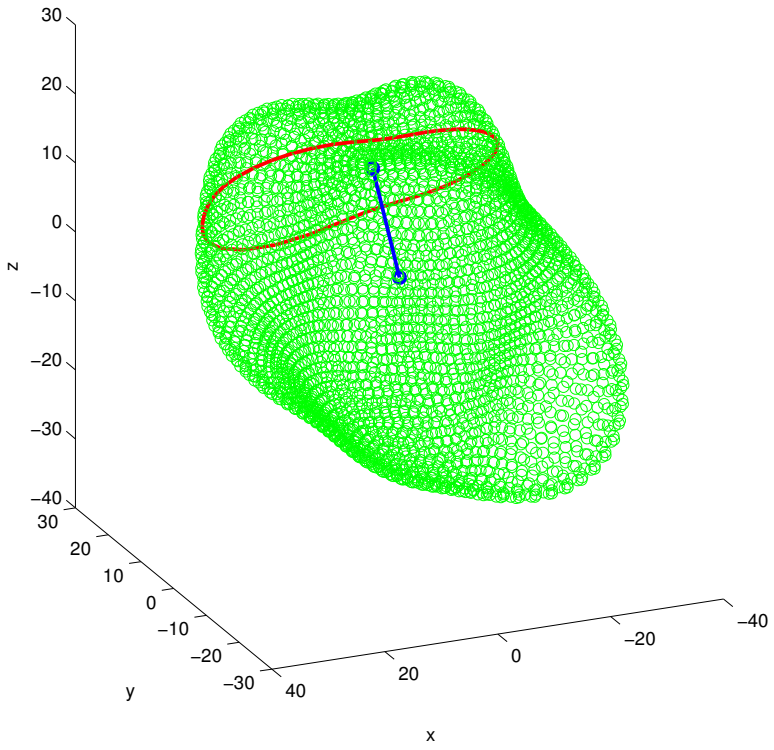

Figure 4.11: Curve that is "almost" on the plane

- In order to be sure that all points are on the plane, an algorithm was developed using basic vector geometry. To obtain the equation of the plane and the exact coordinates of each point on it, every point is projected on the given plane. This can be done because the distance between the plane and those points that are within a defined distance of it is very small. The result of this algorithm can be seen in Figure 4.12 and Figure 4.13

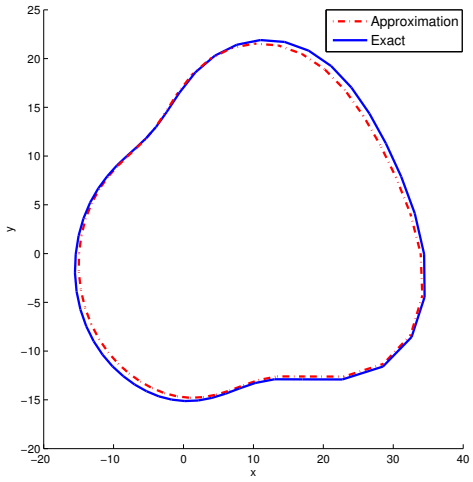

Figure 4.12: Almost-on-plane points and exact plane points

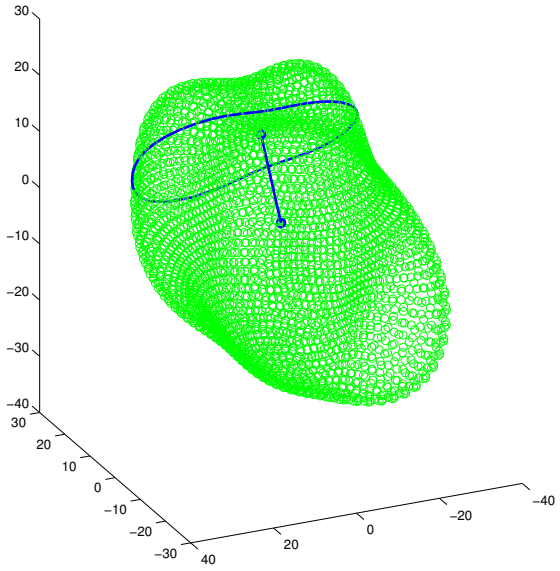

Figure 4.13: Curve that is exactly on the plane

Chapter 5

Discussion and Future Work

This thesis has presented a study of the respiratory motion of the heart, as obtained from asynchronous sets of MR images of 2D slices, in order to obtain a model of the heart and to parameterize its movement. Several goals have been successfully achieved, and are listed below:

- Frame-by-frame segmentation of a set of sagittal MRI images.
- Detection of the respiratory phase in a set of Navigator images.
- Construction of a model heart for each respiratory phase using spherical harmonics.
- Construction of an average heart that moves primarily due to respiratory motion.
- Sectioning of the average heart at any respiratory phase along an arbitrary plane.

The parameterization of the heart and the respiratory motion was achieved with the procedures described in the preceding chapters.

The first chapter contains an introduction to the problem of atrial fibrillation, as well as a brief anatomical and physiological description of the human organs involved in this research. Reference has been made to previous techniques that have contributed to the development of this thesis.

The second chapter contains an explanation of the main theoretical concepts that were used throughout the following chapters.

The third chapter describes how we achieved the first three main goals itemized above. Two linked data sets have been analyzed to obtain information from them, such as a heart boundary and a respiratory phase. Afterwards, this information was assembled and integrated to create a heart for a chosen set of regularly-sampled respiratory phase values. Appropriate time and spatial interpolation was a challenge, so different approaches have been attempted, analyzed, and presented here. Finally, the use of spherical harmonics was found to be very helpful to accomplish the representation of a 4D heart.

The fourth chapter describes the procedure used for rotation of the heart at each pahse point to obtain an average representation of the organ. Any motion other than such caused by the respiratory movement has been removed to the extent possible. It also explains the sectioning of the average heart in different planes at any desired respiratory phase.

In conlusion, by the end of this project we have provided a set of parameters that define each respiratory level. That is, a plane that cuts the heart in a specific fashion and at a specific respiratory phase will have a vector of numbers that will define it.

The first challenge for future research should be to reverse the process of sampling the heart by a given plane at a given respiratory phase. In other words, given a curve in space and time, such as might be obtained by segmenting an MR image taken at an arbitary angle and phase, it may be possible to extract the relevant information and assign it to a specific respiratory phase. It is not clear to what degree this result will be unique, but it would be of great value in the atrial fibrillation ablation procedures of interest in the larger project and thus an attempt would be worthwhile.

The second challenge for future research would be to combine our results with those presented in the thesis of Gerard Pons [50], in order to obtain a model of the human heart which would include not only the respiratory motion, but also the movement due to heart beating.

Bibliography

[1] Klabunde, Richard (2005). “Cardiovascular Physiology Concepts”. Lippincott Williams & Wilkins. ISBN 978-0781750301.

[2] Blackshear JL, Odell JA (February 1996). "Appendage obliteration to reduce stroke in cardiac surgical patients with atrial fibrillation". Ann. Thorac. Surg. 61 (2): 755–9. doi:10.1016/0003-4975(95)00887-X. PMID 8572814.

[3] Nakagami H, Yamamoto K, Ikeda U, Mitsuhashi T, Goto T, Shimada K (1998). "Mitral regurgitation reduces the risk of stroke in patients with nonrheumatic atrial fibrillation". American Heart Journal 136 (3): 528–32. PMID 9736148.

[4] Al-Saady, N. M., O. A. Abel, A. J. Camm (1999). "Left atrial appendage: structure, function, and role in thromboembolism". Heart 82 (5): 547–55. PMID 1760793

[5] http://heartstrong.wordpress.com

[6] http://www.americanheart.org/presenter.jhtml?identifier=4451

[7] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC101077/

[8] Fuster V, Rydén LE, Cannom DS, et al. (2006). "ACC/AHA/ESC 2006 Guidelines for the Management of Patients with Atrial Fibrillation: a report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines and the European Society of Cardiology Committee for Practice Guidelines (Writing Committee to Revise the 2001 Guidelines for the Management of Patients With Atrial Fibrillation): developed in collaboration with the European Heart Rhythm Association and the Heart Rhythm Society". Circulation 114 (7): e257–354. doi:10.1161/CIRCULATIONAHA.106.177292. PMID 16908781.

[9] http://www.mayoclinic.com/health/atrial-fibrillation/DS00291

[10] Brendan Phibbs (2007). “The human heart: a basic guide to heart disease”. Lippincott Williams & Wilkins.

[11] http://milestoneclinic.com

[12] Guyton, A.C. & Hall, J.E. (2006). “Textbook of Medical Physiology (11th ed.)”. Elsevier Saunder. ISBN 0-7216-0240-1.

[13] http://www.bami.us

[14] Weinberger SE (2004). "Principles of Pulmonary Medicine (4th ed. ed.)". Saunders. ISBN 0-7216-9548-5.

[15] Maton, Anthea; Jean Hopkins, Charles William McLaughlin, Susan Johnson, Maryanna Quon Warner, David LaHart, Jill D. Wright (1993). 2Human Biology and Health". Englewood Cliffs, New Jersey, USA: Prentice Hall. ISBN 0-13-981176-1.

[16] http://www.bodysoundsi.com

[17] Dee Unglaub Silverthorn. "Human Physiology". University of Texas.

[18] J.D. Dougherty (1970). "The relation of respiratory changes in the horizontal QRS and T-wave axes to movement of the thoracic electrodes". J. Electrocardiology.

[19] J.D. Doughert (1970). "Change in the frontal QRS axis with changes in the anatomic positions of the heart". J. Electrocardiology.

[20] H.G. Borgen, B.M.T. Lantz, R.R. Miller and D.T. Mason (1977). "Effect of respiration on cardiac motion determined by cineangiography". Acta Radiologica Diagnosis.

[21] Y. Wang, S.J. Rieder and R.L. Ehman (1995). "Respiratory motion of the heart: kinematics and the implications for the spatial resolution in coronary imaging. Man. Reson. Med.".

[22] K. McLeish, D.L.G. Hill, D.Atkinson, J.M. Blackall and R. Razavi (2002). "A study of the motion and deformation of the heart due to respiration". In Proc. Intl. Soc. Mag. Reson. Med.

[23] Ramkrishnan Narayanan, Jeffrey A. Fessler, Hyunjin Park, and Charles R. Meyer (2005). "Diffeomorphic nonlinear transformations: A local parametric approach for image registration". In IPMI, pages 174–185.

[24] Anthony E. Lujan, James M. Balter, and Randall K. Ten Haken (2003). "A method for incorporating organ motion due to breathing into 3d dose calculations in the liver: Sensitivity to variations in motion". Medical Physics, 30(10):2643–2649.

[25] T Neicu, H Shirato, Y Seppenwoolde, and S B Jiang (2003). "Synchronized moving aperture radiation therapy (smart): average tumour trajectory for lung patients". Physics in Medicine and Biology, 48(5):587–598.

[26] Gregory C Sharp, Steve B Jiang, Shinichi Shimizu, and Hiroki Shirato (2004). "Prediction of respiratory tumour motion for real-time image-guided radiotherapy". Physics in Medicine and Biology, 49(3):425–440.

[27] Huanmei Wu, Gregory C Sharp, Betty Salzberg, David Kaeli, Hiroki Shirato, and Steve B Jiang (2004). "A finite state model for respiratory motion analysis in image guided radiation therapy". Physics in Medicine and Biology, 49(23):5357–5372.

[28] D Ruan, J A Fessler, J M Balter, R I Berbeco, S Nishioka, and H Shirato (2008). “Inference of hysteretic respiratory tumor motion from external surrogates: a state augmentation approach”. Physics in Medicine and Biology, 53(11):2923–2936.

[29] D Ruan, J A Fessler, J M Balter, and P J Keall (2009). “Real-time profiling of respiratory motion: baseline drift, frequency variation and fundamental pattern change”. Physics in Medicine and Biology, 54(15):4777–4792.

[30] Rogerio Richa, Antonio P. L. Bo, and Philippe Poignet (2008). “Motion prediction for tracking the beating heart”. In Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE, pages 3261–3264.

[31] W. Bachta, P. Renaud, L. Cuvillon, E. Laroche, A. Forgione, and J. Gangloff (2009). “Motion prediction for computerassisted beating heart surgery”. Biomedical Engineering, IEEE Transactions on, 56(11):2551–2563.

[32] G. Pons Moll, G. Crosas Cano, G. Tadmor, R.S. MacLeod, B. Rosenhahn, and D.H. Brooks (2009). “Parametric Modeling of the Beating Heart with Respiratory Motion Extracted from MR Images”. Computers in Cardiology, Park City, Utah.

[33] G. Pons Moll, G. Tadmor, R.S. MacLeod, B. Rosenhahn, and D.H. Brooks (2009). “4D Cardiac Segmentation of the Epicardium and Left Ventricle”. World Congress 2009 on Medical Physics and Biomedical Engineering, Munich, Germany.

[34] M. Kass, A. Witkin, and D. Terzopoulos (1988). “Snakes: Active contour models. International Journal of Computer Vision”, 1(4):321-331,1988.

[35] Horn, Roger A. and Johnson, Charles R (1985). "Matrix Analysis". Cambridge University Press. ISBN 0-521-38632-2.

[36] Strang G (1998). "Introduction to Linear Algebra 3rd ed.”, Wellesley-Cambridge Press. ISBN 0-9614088-5-5.

[37] Golub, Gene H.; Kahan, William (1965), "Calculating the singular values and pseudo-inverse of a matrix", Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis 2 (2): 205–224, doi:10.1137/0702016.

[38] Brigham, E.O. (2002), The Fast Fourier Transform, New York: Prentice-Hall

[39] P. Duhamel and M. Vetterli (1990). “Fast Fourier transforms: a tutorial review and a state of the art”.

[40] James C. Schatzman (1996). “Accuracy of the discrete Fourier transform and the fast Fourier transform”.

[41] Cooley, James W., and John W. Tukey (1965). "An algorithm for the machine calculation of complex Fourier series".

[42] Ivanov, A.B. (2001). "Legendre function". Hazewinkel, Michiel, Encyclopaedia of Mathematics, Springer, ISBN 978-1556080104.

[43] Abramowitz, Milton; Stegun, Irene A., eds. (1965). "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables". ISBN 978-0486612720.

[44] Solomentsev, E.D. (2001). "Spherical harmonics". Hazewinkel, Michiel, Encyclopaedia of Mathematics, Springer, ISBN 978-1556080104.

[45] MacRobert, T.M. (1967). "Spherical harmonics: An elementary Treatise on Harmonic Functions, with Applications". Pergamon Press.

[46] Eremenko, Alexandre; Jakobson, Dmitry; Nadirashvili, Nikolai (2007). "On nodal sets and nodal domains on S^2 and $\mathbb{R}^2$". Université de Grenoble. Annales de l'Institut Fourier 57 (7): 2345–2360, MR2394544, ISSN 0373-0956.

[47] Watson, G. N.; Whittaker, E. T. (1927). "A Course of Modern Analysis". Cambridge University Press.

[48] Murray, Glenn (2005). "Rotation About an Arbitrary Axis in 3 Dimensions".

[49] http://www.sci.utah.edu/cibc/

[50] Pons, G. (2008). "4D Cardiac MRI Segmentation and Surface Reconstruction". Northeastern University.

Appendix A

Image, Mask and Data Organization

A.1 Frame-by-frame Segmentation

A.1.1 Image Organization

The set of images is organized in numerical folders, to make it easier.

# of Slice	Folder	First Image	Last Image
1	9seg	IM-0009-0001.dcm	IM-0009-0050.dcm
2	11seg	IM-0011-0001.dcm	IM-0011-0050.dcm
...	...	...	...
16	39seg	IM-0039-0001.dcm	IM-0039-0050.dcm

Table A.1: Organization of sagittal images

A.1.2 Mask Organization

The set of masks is organized in a *.mat* file for every slice, which contains a mask for each frame and is located inside its corresponding image folder.

# of Slice	Folder	Filename
1	9seg	seg9.mat
2	11seg	seg11.mat
...	...	...
16	39seg	seg39.mat

Table A.2: Organization of masks

A.1.3 Final Data organization

The set of all the curves is organized in cell arrays. A cell array contains one cell position for each slice and at the same time, every slice is a cell array containing each frame in a different cell position. These last cell elements contain a boundary for a frame and are saved as a 2xNP matrix, where NP corresponds to the number of points with x and y Cartesian coordinates.

A.2 Detection of the Respiratory Phase

A.2.1 Image Organization

The navigator images are organized in numerical folders, to make it easier.

# of Slice	Folder	First Image	Last Image
1	8bars	IM-0008-0000.dcm	IM-0008-0002.dcm
2	10bars	IM-0010-0000.dcm	IM-0010-0002.dcm
...	...	...	...
16	38bars	IM-0038-0000.dcm	IM-0039-0002.dcm

Table A.3: Organization of navigator images

A.2.2 Data Organization

Once all levels are detected, they are organized following a cell structure similar to the structure used in the sagittal data set. A cell array contains one cell position for each slice, which at the same time contains all the levels of a slice in a 1xNumber-of-levels vector.

A.3 Change to 3D Data

As seen in Table 3.1 on page 23, the inter slice resolution is 8mm, and the inter slice resolution is 2mm x 2mm. As a consequence, the third dimension is sampled following the fact that slices will be 4 pixels apart from each other.

There is also a visualization problem that needs to be taken care of. Matlab commands such as *imshow* or *plot* use different references to plot the axes, so data could be incorrectly visualized. When using 3D data this issue is even more important. For this reason, specific functions have been developed to accomplish this change.

Appendix B

MATLAB Code

This appendix encompasses the structure of all the MATLAB functions created during the development of this thesis, in order to show which are the routines involved in each chapter.

B.1 Reconstruction of a Heart for each Respiratory Phase

B.1.1 Frame-by-frame Segmentation

operateSlices.m: Given the path of the location of the images, the file type, and the location of the manual segmentation masks, runs an algorithm to detect the boundaries of the heart for every frame of one slice. Then finds a reduced number of base functions, with which it is possible to accurately represent a boundary in every frame.

- strSliceDir.m: Given the path where the sagittal images are, saves the names of every image in each folder to an element of a cell array.
- strMask.m: Given the path where the masks of the images are, saves the names of every image in each folder to an element of a cell array.
- operateSnakes.m: Given the path of the location of the images, the file type, and the location of the manual segmentation masks, runs an algorithm to detect the boundaries of the heart for every frame of one slice.
  - readImages.m: Given the path of the location of the images and the file type, finds all the images with that file type and saves their names.
  - useMasks.m: Given a set of masks and a set of images, gets the mask data, rotates it, changes it to double format and filters it with a gaussian filter. Then reads the images, changes them to double and adds each mask to each image.
  - clickPoints.m: Given an image, lets you select the center and points of an initial contour to develop the segmentation algorithm.
  - KassExecPts.m: Execution of the Kass et al. snake algorithm.
    - KassSnake.m: Implementation of the Kass et al. snake algorithm.
    - getImgEnrg.m: Calculates the image energy.

        * snakeResample.m: Resamples the snake curve. Resampling is done by inserting points in parts of the snake where the snake control points are far apart compared to the average distance. At the same time removing points in parts of the snake where they are close together.
            · getAvgDist.m: Calculates the average distance between all the points in the snake.
            · getModulo.m: Calculates the index arithmetic using modulo, this is necessary when the snake forms a closed curve.
    - findCenterEasy.m: Given a curve, drawn with a specific number of points, calculates its center and length. Arclength assigned to each point is obtained using distance between points.

- forwardSVD.m: Given the center and length of a set of curves, the set of curves, and the size of the step between points, calculates the SVD of the curves to obtain a specific number of base functions, in order to represent the whole set of curves.
    - subtractCenter.m: Subtraction of the center of a curve to all the points.
    - polarCoord.m: Given a set of points that create a curve, changes them from cartesian coordinates to polar coordinates.
    - interpSnake.m: Given a set of points in polar coordinates that create a curve, uses splines to interpolate a point with a step of a given number of degrees.
    - normalizeR.m: Given a set of points from a curve, divides every point by the maximum radius, in order to apply a normalization.
    - findAvSpline.m: Given a set of splines, obtains the average spline calculating the average of each point.
    - subtractMeanSpline.m: Subtraction of the mean curve to a set of curves.
    - ljMultiply.m: Given a set of splines and its lengths, multiplies each spline by its length.
    - svdMethod.m: Given a set of splines, runs the SVD algorithm to find base functions that define all the different splines and selects the space dimension based on the user input parameter.

- backwardSVD.m: Given the SVD parameters obtained before, the number of base functions, the maximum radius, the center and lengths of the set of curves, the average curves and the number of curves, obtains a set of aproximated curves.
    - invSvdMethod.m: Given a set of parameters that are the result of a previous execution of the SVD method, inverts the process to obtain a matrix which contains new curves, based on a specific number of base functions taken from the SVD.
    - ljDivide.m: Given a set of splines and its lengths, divides each spline by its length.

- addMeanSpline.m: Addition of the mean curve to a set of curves.
- polarCoord.m: Given a set of points that create a curve, changes from cartesian coordinates to polar coordinates.
- deNormR.m: Given a set of points from a curve, multiplies every point by a normalization radius parameter, in order to revert the normalization.
- cartCoord.m: Given a set of points that create a curve, changes from polar coordinates to cartesian coordinates.
- addCenter.m: Addition of the center of a curve to its points.

B.1.2 Detection of the Respiratory Phase

normalizeAll.m: Given the path of the location of the navigator images and the file type, finds the navigator levels for each bar associated to each frame of that slice, and then normalizes them between 0 and 1.

- readImages.m: Given the path of the location of the images and the file type, finds all the images with that file type and saves their names.
- setLowHigh.m: Given a navigator image, lets the user select an interval which contains the navigator level that the user wants to detect.
- readRespLevels.m: Given a navigator image and the low and high ends of an interval, finds the value and the point at the top of each level of the navigator.

B.1.3 Relationship between Segmentation and Phase Detection

getVectLev.m: Given a set of levels organized in cell arrays, puts them in a row vector with the appropriate respiration sign.

- derivCheck.m: Given a set of points indicating the levels of a series of navigator images, assigns a value depending on the derivative of the curve of points at that specific point.
- levelLine.m: Given a set of navigator levels, plots them between -1 and 1, where -1 to 0 is expiration and 0 to 1 is inspiration.

B.2 Time and Spatial Interpolation

B.2.1 Level Interpolation

convInterp.m: Given a set of points indicating the levels of a series of navigator images, interpolates a denser time grid between points.

B.2.2 Curve Fitting

fitLevelsErr.m: Given a set of points indicating the levels of a series of navigator images, fits every point with a specific type of function, considering a total number of given neighbors. Level values are updated at the same time than the execution.

B.2.3 Change to 3D Data

add3DimAll.m: Given all the curves corresponding to every frame of every slice in XY and a Z value, adds the third dimension to the curves.

- add3DimNot.m: Given all the curves corresponding to every frame of a slice in XY and a Z value, adds the third dimension to the curves.

reOrgCoord.m: Given a set of points that create a curve, changes the order of the coordinates.

B.2.4 Convex Interpolation

linIntAll.m: Given a set of splines, a set of navigator levels, the number of frames per slice and an interval of values, interpolates the curves associated with this interval.

- intLevelTestNew.m: Given a set of splines, a set of navigator levels and an interval of values, interpolates the curves associated with this interval.
- intLevelNew.m: Given a set of splines, a set of navigator levels and a value, interpolates the curve associated with this value.

B.2.5 2D Fourier Filtering and Interpolation

interpAllAlpha.m: Given a set of curves for all slices, the center of the % curves, a corrsponding set of navigator levels, an interval % of alpha values, the distance between points in a curve, % the number of harmonics for the fft, the number of % harmonics that the user wants to keep and the plotting color % and pattern, obtains a set of interpolated curves.

- interpIntAlpha.m: Given a set of curves for a specific slice, the center of the curves, the corresponding navigator levels, an interval of alpha values, the distance between points in a curve, the number of harmonics for the fft, the number of harmonics that the user wants to keep and the plotting color and pattern, obtains a set of interpolated curves.
- fftMethod.m
  - getVectLev.m: Given a set of levels organized in cell arrays, puts them in a row vector with the appropriate respiration sign.
  - allSpline2Pol.m: Given a set of curves in cartesian coordinates, subtracts the center and changes the curves from cartesian coordinates to polar coordinates.
  - buildSC.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements 1, $\cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level.

  - dftCurves.m: Given a set of curves and a number of harmonics, obtains the FFT of the curves and saves the specified ammount of harmonics.
  - dftCoeffs.m: Given two matrices M and Y, calculates C.

- newCurveThTest.m: Given an interval for which there is no curves, the FFT coefficients of all the curves, and the step that separates every point interpolates a new curve for every value of the interval.

compareCurves3D.m: Given a set of original curves and the FFT coefficients that are used to obtained their FFT, shows a 3D plot comparing each original curve for every navigator level to the new curves obtained by fourier filtering.

- originalCurveTest.m: Given the set of original curves, the center of the slice, the first and last curves to plot and the color to plot, subtracts the center to every curve and plots them in 2D.
  - subtractCenter.m: Subtraction of the center of a curve to its points.
- newCurveThTest.m: Given an interval for which there is no curves, the FFT coefficients of all the curves, and the step that separates every point, interpolates a new curve for every value of the interval.
  - newCurveTh.m: Given a level for which there is no curve, the FFT coefficients of all the curves and the step that separates every point, interpolates a new curve for this level.
    * newPointTh.m: Given a level for which there is no curve, the FFT coefficients of all the curves, the index indicating the phase and a matrix containing the FFT base fucntions, interpolates a new point of the new curve.
      · buildSC.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements $1, \cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level.
  - cartCoord.m: Given a set of points that create a curve, changes them from polar coordinates to cartesian coordinates.

plot3D.m: Given a set of curves and the color.

B.2.6 3D Fourier Filtering and Interpolation

operateModelF.m: Given a set of curves of the heart and a global center, the set of corresponding navigator levels, the desired number of harmonics in θ and φ, the number of desired harmonics in α and its parameters, obtains a set of coeficients which are meant to be used in an interpolation method.

- allSpline2Sph.m: Given a set of curves in cartesian coordinates, subtracts the center and changes the curves from cartesian coordinates to spherical coordinates.

  - subtractCenter3.m: Subtraction of the center of a curve to its points.
  - sphCoord.m: Given a set of points that create a curve, changes them from cartesian coordinates to spherical coordinates.

- getVect.m: Given a set of curves in spherical coorinates, gets three vectors containing each coordinate.
- basePHopt2.m: Given a set of values for θ and φ, creates an output matrix which contains rows with the elements 1, $\cos(n\pi val)$, $\sin(n\pi val)$, ... where n goes from 1 to the total number of desired harmonics.
- getVectLev.m: Given a set of levels organized in cell arrays, puts them in a row vector with the appropriate respiration sign.
- baseArep2.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements 1, $\cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level. Rows are repeated considering the number of data points in a heart, so that it is possible to create combined base functions afterwards.
- base2M.m: Given two matrices which contain base fuunctions, calculates new base functions as cross products between all of them.
- svdMatrix2.m: Given a matrix, calculates the SVD of its correlation matrix.
- getBigV.m: Given SVD parameters and a coefficients matrix, obtains the corresponding V matrix.
- calcBA.m: Given SVD matrices and the real data, obtains the interpolation coefficients.

plot4DF.m

- basePHopt2.m: Given a set of values for θ and φ, creates an output matrix which contains rows with the elements 1, $\cos(n\pi val)$, $\sin(n\pi val)$, ... where n goes from 1 to the total number of desired harmonics.
- baseArep2.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements 1, $\cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level. Rows are repeated considering the number of data points in a heart, so that it is possible to create combined base functions afterwards.
- base2M.m: Given two matrices which contain base fuunctions, calculates new base functions as cross products between all of them.
- finalData.m: Given a data matrix and a vector of coefficients, obtains the interpolated data.
- thphr2xyz.m: Given a set of curves in spherical coordinates, changes the curves from spherical coordinates to cartesian coordinates.

B.2.7 Spherical Harmonics

operateModel.m: Given a set of curves of the heart and a global center, the % set of corresponding navigator levels, the degree and order of the desired spherical harmonic expansion, the number of % desired harmonics in alpha and its parameters, obtains a set of coeficients which are meant to be used in an interpolation method.

- allSpline2Sph.m: Given a set of curves in cartesian coordinates, subtracts the center and changes the curves from cartesian coordinates to spherical coordinates.
  - subtractCenter3.m: Subtraction of the center of a curve to its points.
  - sphCoord.m: Given a set of points that create a curve, changes them from cartesian coordinates to spherical coordinates.
- getVect.m: Given a set of curves in spherical coorinates, gets three vectors containing each coordinate.
- baseSphHarm2.m: Given a grid of points in θ and φ, a global center, the degree and order of the desired spherical harmonic expansion, obtains the spherical harmonics' base functions.
- getVectLev.m: Given a set of levels organized in cell arrays, puts them in a row vector with the appropriate respiration sign.
- baseArep2.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements 1, $\cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level. Rows are repeated considering the number of data points in a heart, so that it is possible to create combined base functions afterwards.
- base2M.m: Given two matrices which contain base fuunctions, calculates new base functions as cross products between all of them.
- svdMatrix2.m: Given a matrix, calculates the SVD of its correlation matrix.
- getBigV.m: Given SVD parameters and a coefficients matrix, obtains the corresponding V matrix.
- calcBA.m: Given SVD matrices and the real data, obtains the interpolation coefficients.

eqThPh.m: Given a number of points to fill the maximum longitudinal % circle around a sphere, creates an equally spaced grid of % points in spherical coordinates, using the sin function as % a regulation for the number of points depending on the % value of the angle theta.

interpHearts4D.m: Given an equally spaced grid in spherical coordinates, the % desired interpolation degree and order for spherical % harmonics, a set of interpolation coefficients, an interval % of desired levels and the parameters to create the % harmonics in those alpha levels, obtains a set of % interpolated hearts for each respiratory level of the % interval.

- baseSphHarm2.m: Given a grid of points in θ and φ, a global center, the degree and order of the desired spherical harmonic expansion, obtains the spherical harmonics' base functions.
- baseArep2.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements 1, $\cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level. Rows are repeated considering the number of data points in a heart, so that it is possible to create combined base functions afterwards.
- base2M.m: Given two matrices which contain base fuunctions, calculates new base functions as cross products between all of them.
- finalData.m: Given a data matrix and a vector of coefficients, obtains the interpolated data.
- thphr2xyz.m: Given a set of curves in spherical coordinates, changes the curves from spherical coordinates to cartesian coordinates.
- plotHearts.m: Given a set of hearts, plots them like a video.

B.3 Construction of an Average Heart

B.3.1 Data Rotation

operateHearts.m: Given a set of coefficients, a number of points for the biggest circle around the sphere and a plot command, interpolates an average heart that moves following the respiratory phase.

- eqThPh.m: Given a number of points to fill the maximum longitudinal circle around a sphere, creates an equally spaced grid of points in spherical coordinates, using the sin function as a regulation for the number of points depending on the value of the angle θ.
- interpHearts4D.m: Given an equally spaced grid in spherical coordinates, the desired interpolation degree and order for spherical harmonics, a set of interpolation coefficients, an interval of desired levels and the parameters to create the harmonics in those alpha levels, obtains a set of interpolated hearts for each respiratory level of the interval.
  - baseSphHarm2.m: Given a grid of points in θ and φ, a global center, the degree and order of the desired spherical harmonic expansion, obtains the spherical harmonics' base functions.
  - baseArep2.m: Given a set of navigator levels for a slice, creates an output matrix which contains rows with the elements 1, $\cos(n\pi\alpha)$, $\sin(n\pi\alpha)$, ... where n goes from 1 to the total number of desired harmonics and α is the navigator level. Rows are repeated considering the number of data points in a heart, so that it is possible to create combined base functions afterwards.

  - base2M.m: Given two matrices which contain base fuunctions, calculates new base functions as cross products between all of them.
  - finalData.m: Given a data matrix and a vector of coefficients, obtains the interpolated data.
  - thphr2xyz.m: Given a set of curves in spherical coordinates, changes the curves from spherical coordinates to cartesian coordinates.
  - plotHearts.m: Given a set of hearts, plots them like a video.

- dataRotation.m: Given a set of hearts, finds a basis of three vectors that describes each heart and applies a rotation to fit the standard XYZ axes, following the right hand rule.

  - findBase3U.m: Given a set of hearts expressed by points, which are the columns of a matrix, subtracts the center of mass and finds a base of three vectors for each heart. These vectors create an ortonormal basis, where the first vector represents the direction of maximum variation of the heart, the second represents the second direction of maximum variation and so on.
    * centerHearts.m: Given a set of hearts, finds the center of mass for each heart and subtracts it.
      · findCoM.m: Given a heart, finds its center of mass.
      · subtractCoM.m: Given a heart, subtracts its center of mass to each point.
    * svdAll3D.m: Given a set of hearts expressed by points, which are the columns of a matrix, uses SVD to find a base of three vectors. These vectors create an ortonormal basis, where the first vector represents the direction of maximum variation of the heart, the second represents the second direction of maximum variation and so on.
      · svdHeart3D.m: Given a heart expressed by points, which are the columns of a matrix, uses SVD to find a base of three vectors. These vectors create an ortonormal basis, where the first vector represents the direction of maximum variation of the heart, the second represents the second direction of maximum variation and so on.
  - rotMall.m: Given three base vectors for each heart, obtains a rotation matrix to rotate every heart to the common XYZ axis, following the right hand rule. It also obtains the rotation angles contained in each rotation.
    * firstRotMall.m: Given three base vectors for each heart, obtains a rotation matrix to rotate the first vector of each heart to the desired XYZ axis.
      · firstRotM.m: Given three base vectors of a heart, obtains a rotation matrix to rotate the first vector of the heart to the desired XYZ axis. Rotation is divided into two steps, obtaining two different angles.

        * secRotMall.m: Given three base vectors for each heart, obtains a rotation matrix to rotate the second vector of each heart to the desired XYZ axis. If the first vector was already on a common XYZ axis, the third vector will automatically fall on the remaining XYZ axis, following the right hand rule.
            · secRotM.m: Given three base vectors of a heart, obtains a rotation matrix to rotate the second vector of the heart to the desired XYZ axis.
    – rotHearts.m: Given a set of hearts heart and two rotation matrices for each respiratory level, rotates each heart to fit the common XYZ axes, following the right hand rule.

B.3.2 Average Heart

operateHearts.m: Given a set of coefficients, a number of points for the biggest circle around the sphere and a plot command, interpolates an average heart that moves following the respiratory phase.

- findAvHeart.m: Given a set of hearts, finds the average heart. A point of the average heart is obtained averaging the addition of the same point of each heart.
- avheartInvRot.m: Given an average heart and two rotation matrices for each respiratory level, rotates the average heart back to the corresponding orientation for each level.
- addCoMall.m: Given a set of hearts, adds the center of mass to each heart.
    – addCoM.m: Given a heart, adds its center of mass to each point.
- plotHearts.m: Given a set of hearts, plots them like a video.

B.3.3 Heart Sectioning

heart2SphCoef.m: Given a heart in cartesian coordinates, the degree and order of the desired spherical harmonic expansion, obtains a set of coeficients which are meant to be used in an interpolation method.

- sphCoord.m: Given a set of points that create a curve, changes them from cartesian coordinates to spherical coordinates.
- baseSphHarm2.m: Given a grid of points in θ and φ, a global center, the degree and order of the desired spherical harmonic expansion, obtains the spherical harmonics' base functions.
- svdMatrix2.m: Given a matrix, calculates the SVD of its correlation matrix.
- getBigV.m: Given SVD parameters and a coefficients matrix, obtains the corresponding V matrix.

- calcBA.m: Given SVD matrices and the real data, obtains the interpolation coefficients..

searchPlanePoints.m: Given a point in spherical coordinates, the degree and order of the desired spherical harmonic expansion, the interpolation coefficients for that specific heart and the number of points to use, goes through an iterative method to obtain the interpolated value of that specific heart when it is crossed by the plane which is normal to the vector that goes from the origin to the input point p.

- findMag2.m: Given an angle φ in spherical coordinates, a normal vector and its magnitude, the degree and order of the desired spherical harmonics expansion, the interpolation coefficients for a specific heart, obtains the closest possible point to the point of the heart that sits on the plane which is normal to the input normal vector.
  - interpPointHeart.m: Given a point in spherical coordinates, the degree and order of the desired spherical harmonic expansion, and the interpolation coefficients for that specific heart, obtains the interpolated value of that specific heart at the point given.
    - baseSphHarm2.m: Given a grid of points in θ and φ, a global center, the degree and order of the desired spherical harmonic expansion, obtains the spherical harmonics' base functions.
    - finalData.m: Given a data matrix and a vector of coefficients, obtains the interpolated data.
  - thphr2xyz.m: Given a set of curves in spherical coordinates, changes the curves from spherical coordinates to cartesian coordinates.
- projectOnPlane.m: Given a set of points in cartesian coordinates that are almost on a plane and the normal vector to that plane, calculates the points that are exactly on the plane.