NTNU
Innovation and Creativity

Modelling fibre orientation of the left ventricular human heart wall

Knut Vidar Løvøy Siem

Master of Science in Computer Science
Submission date: June 2007
Supervisor: Ketil Bø, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The muscle fibres of the left ventricular human heart wall are oriented at different angles throughout the wall. In this thesis, an attempt will be made to obtain and represent these fibers using techniques from image processing and visualization. This can, for a study of heart dynamics, later be used to simulate heart beats by implementing muscle fibre contraction and volumetric constraints. The available data are MR data supplied by Medisinsk teknisk forskningssenter. The thesis builds on work done in a pre-project.

Assignment given: 15. January 2007
Supervisor: Ketil Bø, IDI

Abstract

The purpose of this thesis is to obtain and represent the orientation of the muscle fibres in the left ventricular wall of the human heart. The orientation of these fibres vary continuously through the wall. This report features an introduction to the human heart and medical imaging techniques. Attention is gradually drawn to concepts in computer science, and how they can help us get a "clearer picture" of the internals of, perhaps, the most important organ in the human body.

A highly detailed Magnetic Resonance Imaging data set of the left ventricle cavity is used as a base for the analysis with 3-D morphological transformations. Also, a 3-D extension of the Hough transformation is developed. This does not seem to have been done before. An attempt is made to obtain the general trend of the trabeculae carneae, as it is believed that this is the orientation of the inner-most muscle fibres of the heart wall.

Suggestions for further work include refinement of the proposed 3-D Hough transformation to yield lines that can be used as guides for parametric curves. Also a brief introduction to Diffusion Tensor Magnetic Resonance Imaging is given.

Preface

This master thesis has been written for the Knowledge-Based Systems group at the Department of Computer and Information Science at the Norwegian University of Science and Technology.

I would like to thank my supervisor, Ketil Bø (IDI), for guidance both with this thesis, and the preceding project. Bjarne Bergheim (ISB) was so kind to provide me with a high-resolution data set of the left ventricle, and initial guidance on cardiac anatomy. I want to credit Harald Hanche-Olsen (IMF) for helping me with 3-D geometry at a critical point in my work, and Ole Christian Eidheim (IDI) for conversations on mathematical morphology.

I would also like to thank Unn Aursøy, Johannes Odland and other students for lengthy discussions on various topics. Finally, I want to express my sincere gratitude to my fellow student Ståle Lyngaas, for his countless hours of help, discussion and ever-present interest in every aspect of this work.

Trondheim, June 19, 2007

Contents

List of Figures

Chapter 1

Introduction

A helical fibre configuration is the base for many studies of the heart, and in particular left ventricular, dynamics. The discoveries surrounding the ventricular myocardial band made by Dr. Torrent-Guasp in the 1950s have had significant impact on our understanding of the heart [1, 2]. Still, there are unanswered questions, in particular with regards to the link between the heart's structure and function. This thesis is the continuation of a pre-project [3] where approaches to model the structure of the fibres of the left ventricular wall were presented.

A new technique has been developed to obtain highly detailed data of the left ventricle (LV) cavity, in particular the trabeculae carneae muscular columns. These structures lack a documented functional purpose, which makes them all the more interesting. The study of their function do, however, exceed the strict scope of this thesis. Regarding anatomical regions of the heart, the LV is, as stated, the subject of the study. Dialogue with Bjarne Bergheim (Department of Circulation and Medical Imaging (ISB), NTNU) has revealed that this is a reasonable simplification, given the strength and independence of the LV.

1.1 Problem definition

Problem: *how to obtain and represent the varying muscle fibre orientation in the left ventricular human heart wall*

To explain the problem more thoroughly, I will first give a brief explanation of the terms in the problem statement.

The human heart is a powerful muscular organ situated in the chest [4]. It is the primal pump of our circulatory system. An average heart has a throughput of 7200 litres of blood per day [5].

The heart wall is the active muscle tissue of the heart. This thesis focuses on just one of the heart cavities and the surrounding muscle tissue. Therefore, in this thesis, the term wall extends to the tissue separating the cavities.

The left ventricle is one of four cavities in the heart, two atria atop two ventricles [4]. The wall enclosing the LV is significantly thicker than other parts of the heart wall, making the LV relatively independent with regards to muscle dynamics. The LV receives newly oxygen-enriched blood from the lungs via the left atrium and pumps it out to the rest of the body.

Muscle fibres are the principal components of muscles, that is contractile tissue. They serve their purpose of producing force and causing motion [6] through contraction and relaxation. Note that the fibres themselves cannot stretch back out — collagen tissue makes this possible. There are three types of muscles: skeletal, cardiac and smooth. This thesis focuses on cardiac muscle fibres.

Muscle fibre orientation relates to the angle at which the muscle fibres lie in the heart wall. The fibres are, however, all parallel to the heart wall. As a result of a complex growth process this angle varies throughout the wall but is relatively constant (locally) at any given depth [7].

Obtaining a muscle fibre orientation will involve various techniques from image processing combined with a priori anatomical knowledge.

The representation of muscle fibres should be one that supports visualization in 3-D space. Later, the fibres can be set in motion to simulate heart dynamics.

1.2 Goals and limitations

The goal of this project is to obtain and represent the orientation of the inner-most layer of muscle fibres in the LV of the human heart. Due to the strong inter-diciplinary characteristics of this work, implementation is based on prototyping. It is desired to test some pre-processing operations to obtain the fibre orientation, and then find a simple spatial representation for the muscle fibres themselves.

1.3 Outline

Chapter 2 gives background information on the anatomy of the heart, its formation and its function.

Chapter 3 levels the reader with recent material on the subject of fibre orientation, and heart and muscle dynamics.

Chapter 4 introduces relevant methods and procedures from the realm of image processing and visualization.

Chapter 5 evaluates the solution alternatives, and elaborates on the chosen solution.

Chapter 6 gives details on the implementation of the prototype.

Chapter 7 presents results and an accompanying discussion

Chapter 8 concludes the thesis.

Chapter 9 gives guidelines for future work.

Chapter 2

Medical background

This chapter will present the medical background relevant to the project. First an anatomical description of the heart will be given. This will be followed by an introduction to heart formation and the cardiac cycle. Most of this material was presented in the pre-project [3], but as it applies equally to this work it is repeated.

2.1 Anatomy of the heart

The heart is a muscular organ located between the lungs in the chest, protected by the rib cage and enclosed in a sac known as the pericardium [4]. It weighs approximately 300g and beats roughly 100 000 times in 24 hours [8]. The heart is the organ that drives blood through our double circulatory system [9].

The heart wall has a three-layered structure [10]. On the outside is the *epicardium*, a thin transparent layer which produces a lubricating fluid that allows the heart to move with minimal friction when it beats. The *myocardium* is the actual muscle layer in the heart wall, consisting of cardiac muscle tissue. These muscle fibres are "involuntary, striated and branched" [10]. The innermost layer of the heart wall is the *endocardium*, a thin layer of epithelium also found in the blood vessels. The thickness of the wall varies over the surface of the heart. More specifically it is the thickness of the myocardium that varies. The wall of the right ventricle (RV) is thicker than that of the atria, and the wall of the LV is even thicker than that of RV. The fibres in the wall are not uniformly oriented throughout the wall. Instead their orientation varies continuously; some studies report a total fibre angle variation of 140°[7]. The reason for such an intricate composition is the complex growth process (as will be described in Section 2.2). This composition yields homogenous stress and strain workload of the muscle fibres [11].

The heart is subdivided into two halves by walls known as the interatrial and interventricular septa [10]. Each half controls one part of our circulatory system. The halves are both subdivided into two cavities, an atrium atop a ventricle, by the atrioventricular valves; on the left side, the mitral valve and on the right, the tricuspid valve. [4]. This yields a total of four cavities, namely the left atrium, the right atrium, the left ventricle (LV) and the right ventricle (RV). This can be seen in the illustration in Figure 2.1 [12].

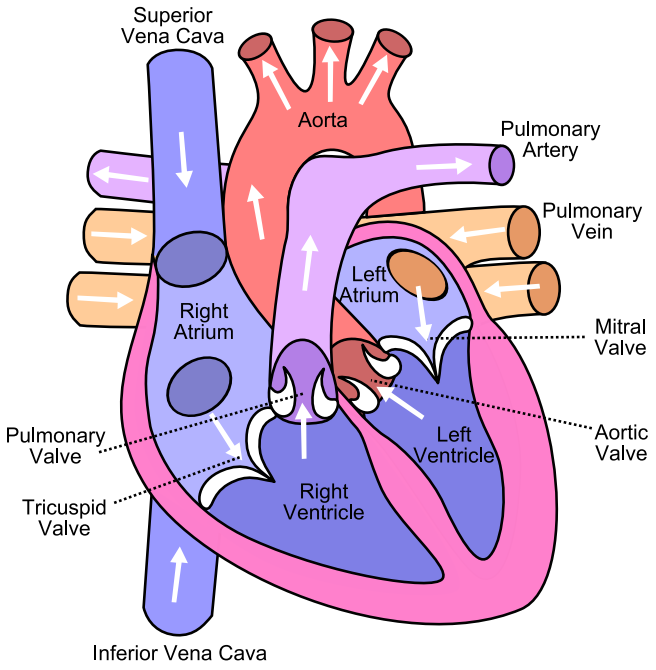

Figure 2.1: Diagram of the human heart (Eric Pierce)

Inside the ventricles is a system of irregular muscular columns, the *trabeculae carnae* [4]. Some take the form of ridges in the wall; others have only their ends attached to the wall. A third kind, musculi papillares, connect with the chordae tendineae, which in turn are attached to the cusps of the tricuspid and bicuspid valves. The trabeculae carneae on the ventricular wall are denser and more numerous in the LV than in the RV.

2.2 Heart formation

The heart is the first organ to form in the fetal body. Initially it is a system of aortae and veins, but after 18-20 days, two primitive aortae fuse together, receiving also two veins, forming a single tube as can be seen in Figure 2.2 [13, 4]. Furthermore, this tube starts bending on itself, and over the next eight days it becomes superficially similar to an adult heart. This process is illustrated with Figures 2.3, 2.4 and 2.5. Around day 22 [13], the tube

starts rhythmically contracting, an activity it will sustain for the rest of the person's lifetime. In the wall of the heart there now exists "an intricate trabecular network of mesodermal tissue" from which later the trabeculae carneae structures are formed [4].

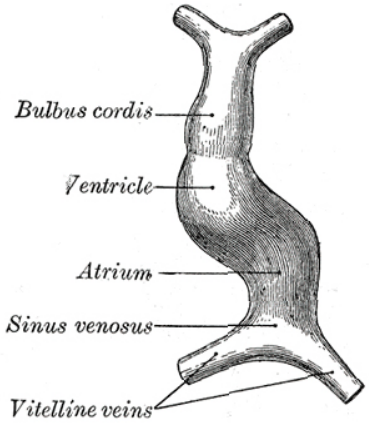

Figure 2.2: Tubular heart before looping [4]

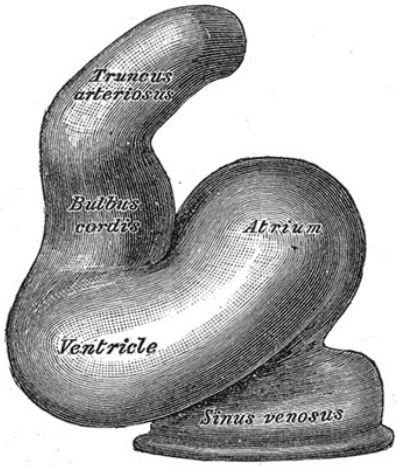

Figure 2.3: Heart of human embryo of about fourteen days [4]

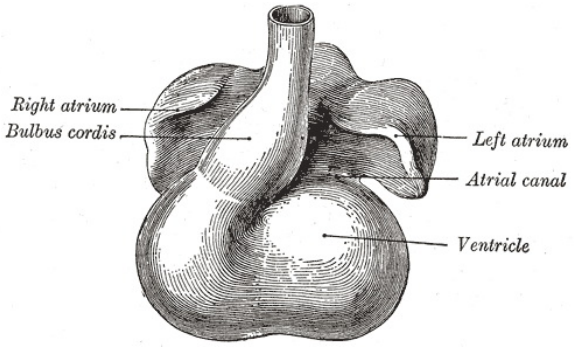

Figure 2.4: Heart after looping showing atrial expansion [4]

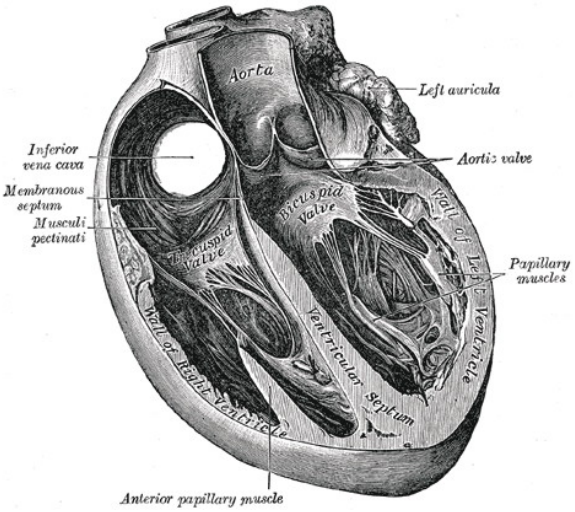

Figure 2.5: Coronal section of a fully developed heart [4]

2.3 Heart function

The heart's purpose is to supply all parts of the body with oxygenated blood. To complete this task it powers double circulatory system, distributing blood and replenishing it with oxygen. The cardiac cycle is what keeps these systems, and therefore the whole body, running.

Pulmonary circulation is the oxygen-replenishing part of the system. Oxygen-poor blood from the right part of the heart is taken to the lungs for oxygen reuptake, and back into the left part of the heart [14, 15]. Systemic circulation is the distribution of oxygenated blood from the left part of the heart, out to the rest of the body, and back to the right part of the heart [16]. The wall of the LV is three to six times thicker than that of the RV because of the work associated with producing sufficient pressure for systemic circulation [12].

Every heartbeat consists of a connected series of events. These events can be categorized into three main phases, atrial systole, ventricular systole and complete cardiac diastole [4]. The terms systole and diastole relate to the contraction and relaxation/dilation of the heart respectively. During atrial systole, accumulated blood in the atria is pushed through the valves and into the ventricles. During ventricular systole, blood in the ventricles is pushed out into the blood vessels. The subsequent phase is a period of complete diastole in the entire heart.

Chapter 3

Related literature

This chapter will present material related to imaging techniques, muscle fibre representation and approximation. Parts of the following were presented in the pre-project [3].

3.1 Imaging techniques

This section introduces various medical imaging techniques, with a special focus on cardiac imaging. The technologies presented are ultrasound, Computerized Tomography (CT) and Magnetic Resonance Imaging (MRI).

3.1.1 Ultrasound

Ultrasound is an imaging technique based on sound waves. Sound waves are waves of compression and decompression of the transmitting medium in the propagation direction, that is they are longitudinal waves. Their velocity is constant given a uniform transmitting medium and a constant temperature. The prefix ultra- refers to the fact that the frequency is higher than audible sound. The higher end of the audible sound frequency spectrum lie from 15 to 20kHz [17]. When ultrasound is used in medical applications, the frequency range of operation is usually 1-12MHz. Figure 3.1 shows an image from an obstetric ultrasound examination.

An ultrasound machine both sends the sound waves, and receives the echoes through a transducer probe [18]. The basis for the transducer probe's function is the piezoelectric, or pressure electricity, effect, discovered by Pierre and Jacques Curie in 1880. The principle is

Figure 3.1: Obstetric ultrasound examination (Sam Pullara, San Francisco, CA, USA)

that quartz (piezoelectric) crystals, produce sound waves as a result of vibration, that is rapid shape change, when an electrical current is applied to them [19]. Conversely they produce an electrical current when hit by sound or pressure waves. A layer of gel on the surface of the subject is applied to avoid the disturbances caused by air between the transducer probe and the subject. The Central Processing Unit (CPU) controls the whole operation, the sending of waves, the registration of received echoes and image generation from these echoes. The machine additionally has facilities such as a display and input control, disk storage and a printer. Scan parameters such as frequency, duration and scan mode can be modified with extra input controls.

3.1.2 Computed tomography

Computerized Axial Tomography (CAT) or CT for short is an imaging technique based on X-rays. The process involves a ring of detectors and a concentric X-ray source rotating, with the object sliding longitudinally in the center [20]. A CT examination usually consists of several images or slices taken as the object is moved. Put together, all these slices form a three-dimensional model of the objects internal structure. Figure 3.2 shows a CT head scan.

One of the strengths of CT imaging compared to MRI is its unique ability to register vastly different tissue types with good contrast [20]. A CT head scan contains information about soft tissue, bone structures and blood vessels. Which tissue type to examine is chosen by the physician after the scan. Newer scanners scan continuously in a spiral motion. This reduces the scan time and thereby greatly increases patient comfort. In these state of the art multi-slice CT scanners, the ring revolves around the object at 120rpm reading four slices per

Figure 3.2: Transversal CT slice of head (Andrew Ciscel)

revolution, a significant data acquisition jump from the earlier 60rpm single slice scanners. This further reduces scan time, enabling the scanning of larger body regions within the time constraints of a breath hold. This eliminates artefacts caused by motion, reducing the need for post-processing.

3.1.3 Magnetic resonance imaging

MRI is an imaging technique based on the principles of Nuclear Magnetic Resonance (NMR). The word nuclear was dropped because of its negative connotations [21]. The NMR phenomenon was discovered independently by Felix Bloch and Edward Purcell in 1946, but it was not until the 1970s that the NMR phenomenon was applied to medical imaging, initially by Raymond Damadian. Like ultrasound, MRI is a non-invasive method [22]. It does, however, produce images of much higher quality, both with respect to noise and contrast. Additionally, an MRI machine can scan arbitrarily oriented planes.

MRI is a very complex subject. What follows is a brief explanation of the physics behind the technology. Each nucleus has the fundamental property spin [21]. When a strong magnetic field is imposed on the region of interest, these spins align with the imposed magnetic field. Most of the spins cancel each other out because of difference of direction, but an excess spin along the applied field results in a tiny net magnetization in that direction. The alignment is not exact; the nucleons precess, or wobble, about the applied magnetic field with a motion similar to that of the top of a spinning top. If radio waves with frequency equal to the

resonance frequency[1] is applied, then this energy can be absorbed by the nucleons and cause a transition to a higher spin energy level [23]. When the Radio Frequency (RF) is turned off the system returns to its original state, aligned with the magnetic field. During this time the nucleons exert RF waves. These are registered and various techniques can be used to create an image from this data.

An MRI machine has many components of which the most prominent is a main magnet [24]. Other components include a radio transmitter, one or more transceiver coils and receivers [25] and magnetic field gradient coils. Also necessary is one or more computers to handle the data and to control the examination process. During examination, the patient lies on a table that can slide into the bore of the magnet. The much weaker magnetic field controlled by the gradient magnets enable modification of the magnetic field on a very local level [24]. The way an MRI machine forms an image is by taking readings from the nuclei of the atoms in the various parts of matter in the body. The registered data is a wide range of frequencies [25]. These data are converted to a digital form in process called analogue-to-digital conversion. An inverse Fourier transform is used to obtain the original image from the frequency data. Figure 3.3 shows an example of an MRI image.

Figure 3.3: Sagittal MRI image of head

The strength of MRI is its ability to examine soft tissue of similar density with greater detail than CT [26]. While CT and conventional X-ray imaging renders dense structures such as bones with great detail, MRI has its main contrast span in the range of soft tissue. Its resolution is similar to that of CT. Also, MRI can depict blood vessels without the use of contrast enhancers. The heart, the vascular system and the thoracic region in general, are but some of the anatomical regions of which MRI is becoming an increasingly popular examination technique. Another benefit of MRI is that it does not utilize X-rays. The

[1]This frequency is also known as the Larmor, precession or wobble frequency [23].

concept of Functional MR (fMR) introduces the possibility of examining not only anatomical structure, but also the various parts' function. This includes the activity of the heart wall as well as neural activity in the brain.

Diffusion Tensor Magnetic Resonance Imaging (DT-MRI) is a new imaging technique that allows for non-invasive, in-vivo studies of anatomical structures. Its potential for obtaining fibre directions extends from its capabilities to register how water diffuses in tissue [27]. Elongated structures such as muscle fibres and neurons have microstructures that constrain water diffusion. Neurons are covered by a myelin sheath that prefers water diffusion along the axon direction. This kind of restricted diffusion labels the matter anisotropic. [28]. The diffusivity information is extracted for each image point and stored in a tensor, later to reveal highly detailed anisotropic diffusion effects [29]. This is done by considering the eigensystem of each tensor, namely three orthogonal eigenvectors and three eigenvalues [27]. The principal eigenvector indicates the direction of fastest diffusion. This is an important feature when it comes to reconstructing fibre-like structures. Visualization of DT-MRI data is difficult due to its multivariate nature [27]. To represent the diffusion effects, glyphs are used. Glyphs are small parameterized volumes that represent data with their shape, color, orientation etc. Ellipsoids, with their three axes and radii are suitable for representing diffusion data. Another common glyph type is superquadrics, shown in Figure 3.4. One of

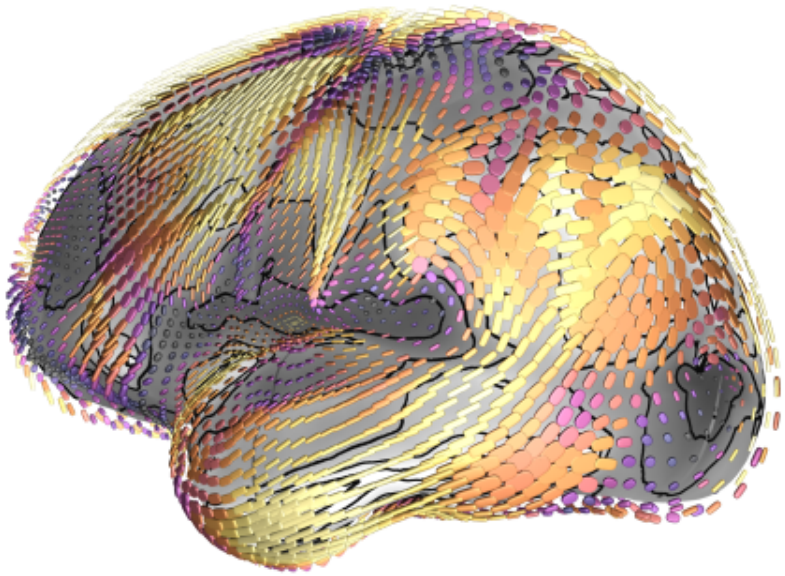

Figure 3.4: Tensor glyphs (Dr. Gordon Kindlmann, [29])

the challenges of DT-MRI is to account for the diffusion effects when fibres diverge, cross or kiss. A physical understanding of these events may result in a model that allows for better and more meaningful visualization [27].

At ISB, a new imaging technology, using MRI, allows high resolution (<0.1mm) imaging of

the LV endocardial surface. The details of this technology has not yet been published. For this reason, no details on this technology is discussed here. It has, however, provided the input dataset for this thesis.

3.2 Fibre orientation approximation

Instead of obtaining fibre orientation from cardiac imaging techniques such as DT-MRI, approximations can be made. One approach is to use fibre stress equations along with some constraints to estimate a muscle fibre orientation in a rotationally symmetric ventricle [7]. With this approach, the following assumptions are made; the myocardium is a soft incompressible material, and the fibres only support stress in the longitudinal direction. Regarding the stree, this may only be true in the systolic phase. These constraints require an approximation of the fibre orientation in the ventricle wall; a linear (Equation 3.1) and an S-shaped (Equation 3.2) is provided [7]:

$$\alpha = -\mu w + \alpha_m \tag{3.1}$$

$$\alpha = -\hat{\mu} \arcsin\left(\frac{2w}{h}\right) + \hat{\alpha}_m \tag{3.2}$$

Here, α is the fibre direction as a function of the position, w, in the wall. μ is a scaling factor and α_m is the angle offset in the middle of the wall. Locally, α is relatively constant. These approximations do not account for a transversal angle in addition to the helical angle. The transversal angle is most prominent at the apex where the inner fibres of the endocardium continue to form the outer fibres of the epicardium [7]. A model that also encompass this aspect is likely to produce a more realistic orientation approximation.

3.3 Modelling fibre dynamics

Given a suitable representation it is possible to set a muscle fibre model in motion by mechanical contraction as is suggested in [30]. Specifically it is suggested that each fibre is constructed of both contractile and elastic elements in parallel. Since a muscle fibre itself is unable to return to its original form after contraction, an elastic element is also included to mimic the collagen tissue of the heart that makes this happen. Inner ventricular pressure and volume graphs are available to control the simulated cardiac cycle.

Chapter 4

Methods and procedures

This chapter will present material from computer science on the subject of image processing and visualization. The methods and procedures presented here will be organized by subject, as the processing pipeline is highly experimental, and the techniques are not confined to some predefined task.

4.1 Mathematical morphology

Mathematical morphology can be thought of as a general image processing framework with composite procedures that focus on shape, and not pixel intensities [31]. Advanced procedures are built from two simple operations: erosion and dilation. The basics of mathematical morphology were described in the pre-project [3] of this thesis. In this section, only the relevant, composite transformations are described. Address Appendix A for a more introductory reading on mathematical morphology.

4.1.1 Top-hat transformations

The top-hat transform under opening, also called the white top-hat transform, is the difference between an image, A, and its opening, given a structuring element, B:

$$WTH_B(A) = A \backslash (A \circ B) \tag{4.1}$$

This transformation will find small bright or narrow features in an image [32]. Conversely, to detect dark features, the closing can be subtracted. This transformation is known as the

black top-hat transform:

$$BTH_B(A) = A\backslash(A \bullet B) \tag{4.2}$$

The self-complementary top-hat is the sum of the black and white top-hats. Simplified, this is the difference between the closing and the opening:

$$SCTH_B(A) = (A \bullet B)\backslash(A \circ B) \tag{4.3}$$

The top-hat transformations are typically used for unevenly lit greyscale images [31] when a simple threshold will not suffice [32].

4.1.2 Hit-or-miss transformation

The hit-or-miss transformation proves useful in shape detection and can be used for *thinning* and *thickening*. It addresses points of a completely defined neighbourhood. This is done by using two-part structuring element, one defining background and the other foreground. These structuring elements have a shared origo and do not overlap. The transform is the intersection of two erosions, one on the foreground with the foreground structuring element, B_1 and vice versa with B_2 [31]:

$$A \circledast B = \{z|(B_1)_z \subseteq A, (B_2)_z \subseteq A^{\mathcal{C}}\} \tag{4.4}$$

4.1.3 Skeletonization

A skeleton is a minimal representation of a shape [31]. There are two common ways of obtaining the skeleton of a shape, by thinning and, by calculating the distance transformation. Presented here, is morphological thinning. The hit-or-miss transformation can be used to find skeletons. This is done with the thinning transform, which is defined as follows [31]:

$$A \bigcirc B = A\backslash(A \circledast B) \tag{4.5}$$

where origo of the composite B is in B_1 of Equation 4.4. This can be reiterated a number of times, a process called sequential thinning, as is defined in Equation 4.6.

$$A \underline{\circ} B = (\dots((A \bigcirc B) \bigcirc B) \bigcirc \dots) \bigcirc B \tag{4.6}$$

Sequential thinning can be applied with all proper and improper 90° rotations of the structuring elements in Figure 4.1 [33]. Improper rotation does not preserve the handedness of the object, resulting in mirroring. This is desired as the set of structuring elements must match every possible constellation. Background cells are coloured grey, the foreground cell red, and don't-care cells green. The don't-care cells can be either foreground or background cells. While preserving its homotopy, hence the name homeotopic thinning, the sequence can

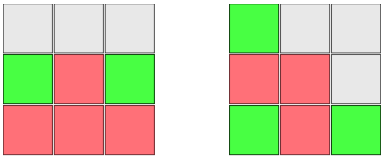

Figure 4.1: Structuring elements for 2-D skeletonization

be reiterated until only the medial axis, or skeleton, remains. The obtained skeleton often has small spurs that, for practical purposes, are not that interesting. These are sequentially pruned away using a particular set of structuring elements in Equation 4.6. This set of structuring elements are all 90°, proper and improper, rotations of Figure 4.2.

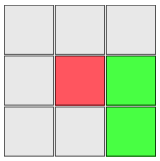

Figure 4.2: Structuring element for 2-D skeleton pruning

4.2 The Hough transformation

The Hough transformation was proposed as an approach to line detection by Hough in 1962 [34]. It has later been extended to detect arbitrary shapes [35]. An infinite number of lines can pass through a point, i, in space by varying the slope, a, and intercept, b, parameters:

$$y_i = ax_i + b \tag{4.7}$$

These lines can be represented in the parameter space by writing this equation as $b = -ax_i + y_i$, resulting in a single line for the point i. The same holds for a second point, j. The two points lie on the same line if the parameters are equal, meaning that the line connecting the two points is defined by the a and b parameter values where the lines intersect in parameter space. However, using Equation 4.7 to represent a line produces a problem; "the slope approaches infinity as the line approaches the vertical" [34]. To remedy this problem, the *normal representation* of a line is used:

$$x\cos\theta + y\sin\theta = \rho \tag{4.8}$$

This representation is similar to polar coordinates with regards to parameters, but where a polar coordinate pair defines a point, P, normal line representation defines a line, L. The only difference is the implicit knowledge that L is perpendicular to the line from origo (O) to P.

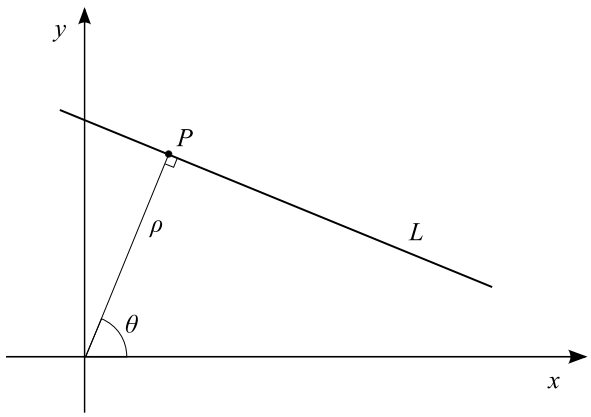

Figure 4.3: Normal representation of 2-D line

The Hough transform of an image is a frequency plot in the parameter space. Usually, each foreground point of the image is subjected to an iteration over the angle range, each with a calculation of the normal distance. With this scheme, every time a pair is found, a cell in an accumulator array, a discretization of the parameter space, is incremented. Note that other approaches exist, in which a continuous parameter space is used. In essence, the parameter pair of a long line segment will be supported by a larger number of points than that of a short segment. This brute-force approach results in quite a bit of noise in the accumulator array. This noise can conceal frequency peaks, especially peaks representing short line segments. To counter background noise, a technique known as *backmapping* can be employed. The procedure adds to the normal Hough transformation procedure, a second pass. In this second pass, instead of calculating the normal distance, the first accumulator is searched along the known angle to obtain the normal distance with best support. In the second accumulator array, only this normal distance is registered for the known angle for the given point. This means that with the voting scheme manifested in the second accumulator array, an image point can only vote for one line, namely the one it is part of that is best supported.

Chapter 5

Solutions

The data set made available by Bjarne Bergheim (ISB) for this thesis work is an implicit, solid volume, with structures such as the trabeculae carneae intact. Since nothing is known of the internals of the wall, such as with DT-MRI data, and the outside of the wall is not imaged, one must rely on the inner wall and the details of the trabeculae carneae. Although on the inner wall, these features do extend a bit into the wall, and it is this region of transition into solid wall that is the most interesting.

The strategy is to utilize the quality of localized focus on shape, of the morphological operations to isolate inner wall detail. To satisfy the goal of approximating muscle fibre orientation, an attempt is made of obtaining a first layer of muscle fibres based on the direction of the trabeculae carneae. With a first layer in place, defining the shape of the ventricular cavity, the equations presented in section 3.2 can be employed to build the heart wall, layer by layer, with a gradual change in fibre orientation. As stated in section 3.2, this model does not account for the transversal angle of the fibres in the wall.

The pre-project [3] revealed a helical pattern on the inner wall of the LV cavity data set. This was obtained with the black top-hat transformation of Equation 4.2. This transformed image is the starting point for the operations described in the following sections.

5.1 Pre-processing

The concept of muscle fibres calls for a thin, or minimal, representation. With this in mind, a technique that stands out is skeletonization. As described in section 4.1, a skeleton is a minimal representation of a shape. For 3-D data, several types of skeletons can be obtained. For the purpose of representing muscle fibres, a medial axis skeleton is the most interesting.

The obtained skeleton is likely to be very complex and noisy, due to the complexity of the wall detail isolated with the top-hat transformation. To remedy this, the process of pruning is suggested. Pruning will gradually remove endpoints until the data set converges. It is believed that the general orientation trend of the wall surface is expressed in the many skeleton stems of the now small, scattered volumes. Repeated pruning should be able to reveal these stems. Figure 5.1 shows the proposed 3-D extension of Figure 4.2. The single, red foreground cell is the candidate for removal.

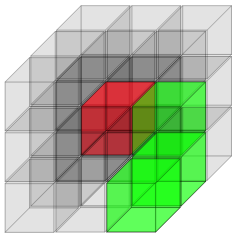

Figure 5.1: Structuring element for 3-D skeleton pruning

To reduce noise in the skeleton, both opening and closing operations were employed before the skeletonizing process was run. They were, however, quickly dismissed from the pipeline because they altered the basic structure of the skeleton by an unacceptable degree. Pseudo-code for this algorithm is given in Algorithm 1. The set of structuring elements, S, is that of Figure 5.1, rotated in all possible directions with a multiple of 90°, both properly and improperly.

Algorithm 1 PRUNE3D

Require: $n > 0$
Require: Binary input image, I
  $I_P \leftarrow I$
  S : the set of structuring elements
  for $i \leftarrow 0$ to n **do**
    for all $s \in S$ **do**
      $I_P \leftarrow I_P \cap \neg(I_P \circledast s)$
    end for
  end for
  return I_P

5.2 Representation

A first step on the way to muscle fibre representation is to locate fibre segments and approximate them as line segments. It is believed that The Hough transformation will be able to detect the general trend of the inner-wall detail. To perform the Hough transformation with

visual feedback, the data set was loaded into DynamicImager [36] [1], and a network built. It was discovered that DynamicImager successfully computes the Hough transform on 2-D images, but the transform of 3-D images seem to be erroneous; the result obtained from the 3-D image was also 3-D. How this is seemingly wrong becomes apparent when one tries to reconstruct the lines in the three-dimensional image space, using only the three available scalar values from the chosen point in the Hough space. The Hough transform is used mostly on two-dimensional images, and extending it to three dimensions, while still detecting lines, is not trivial [37], nor does it seem to have been done and documented. Below, a line-detecting extension of the Hough transform to 3-D is presented, but first some mathematical theory is given, followed by a discussion of the challenges surrounding such an extension.

In any given affine space, a linear equation defines a *hyperplane* [38]. A hyperplane is a codimension-1 subspace [39] that divides an n-dimensional space into two half-spaces [40]. In 2-D, a linear equation has the form $ax + by = c$ or more commonly $y = ax + b$. This is the equation for a line, or hyperplane, in two-dimensional space. In general, a subspace of dimension d in n-dimensional space can be defined by the intersection of $n - d$ hyperplanes. A line is a 1-D space, meaning that in 2-D, a line is the intersection of $2 - 1 = 1$ hyperplanes, which happens to be the line itself. In 3-D, a hyperplane is an ordinary plane. Thus, a line is the intersection of $3 - 1 = 2$ planes. A plane can be defined by three parameters, a, b, c, in a Cartesian coordinate system resulting in a six-parameter definition of a line. There are alternatives to representing a line with some number of hyperplanes, as the following paragraph will explore.

A line-detecting Hough transform in 2-D is in essence detecting hyperplanes. Therefore, a natural extension of this transform to 3-D would be a plane-detecting transform. In a Hough setting, the number of unknown parameters reflect the dimensionality of the accumulator array [41]. This calls for a minimal line representation in 3-D. The naive representation of two intersecting planes would yield an accumulator array filled with redundancies. This is due to the fact that a large number of pairs of planes can produce the same line. In the continuous case, this number is infinite. This alone would break the voting scheme of the Hough transform. A six-dimensional accumulator array would also increase problem complexity with regards to performance and memory usage. Additionally, there is the issue of infinite slope with a distance representation, which was remedied by normal representation in 2-D. Cylindrical and spherical coordinates are extensions of polar coordinates to three dimensions [42], and it is therefore reasonable to assume that they can solve the same problem. The relation to normal representation remains intact, differing only in that the coordinate tuple defines a plane instead of a line. As noted, angular representation is preferred over distance representation due to the problem of infinite slope. This makes a spherical coordinate system more ideal than a cylindrical system. A spherical coordinate system on the following form

[1] DynamicImager [36] is a high-level image processing programming utility that lets the user build a graphical network of operations that data flow in and out of, without doing any coding.

is therefore chosen.

$$x = \rho \sin\phi \cos\psi$$
$$y = \rho \sin\phi \sin\psi$$
$$z = \rho \cos\phi$$

A line can be represented by a tangent plane and an angle at which the line lies in the plane. The tangent plane intersection point can be defined by three spherical coordinates, ϕ, ψ, ρ. The angle, ω, of the line in the plane can be defined by the unit vectors, $\mathbf{u}$ and $\mathbf{v}$, of the tangent plane. Let $P : (p_x, p_y, p_z)$ be any point in the first octant of 3-D space, and let ϕ

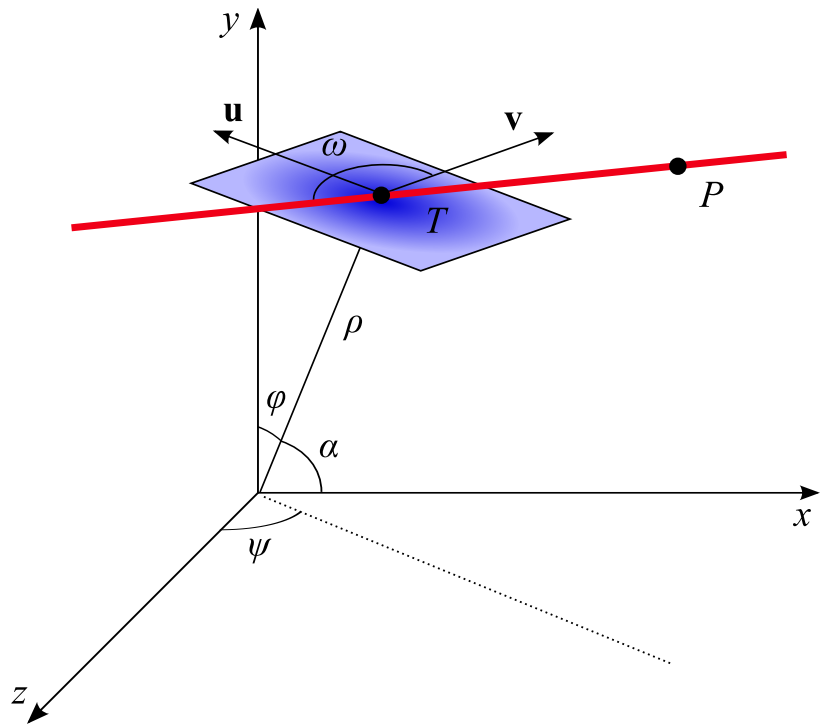

Figure 5.2: Normal representation of 3-D line

and ψ be known orientation angles as illustrated in Figure 5.2. The radius ρ can then be obtained, defining a point, $T : (\phi, \psi, \rho)$, such that $\mathbf{OT} \perp \mathbf{TP}$, that is $\mathbf{OT} \cdot \mathbf{TP} = 0$. This yields:

$$\rho = \frac{p_x \sin\phi cos\psi + p_y \sin\phi \sin\psi + p_z \cos\phi}{\sin^2\phi cos^2\psi + \sin^2\phi \sin^2\psi + \cos^2\phi} \tag{5.1}$$

The tangent plane is defined by T and two unit vectors $\mathbf{u}$ and $\mathbf{v}$. $\mathbf{u}$ is obtained by decreasing ϕ by a known, small angle θ and increasing ρ by a factor of $1/\cos\theta$:

$$\rho' = \frac{\rho}{\cos\theta}$$

$$\phi' = \phi - \theta$$

The equations above give a new point, U. The unit vector $\mathbf{u}$ is the vector from T to U:

$$\mathbf{u} = \begin{pmatrix} \frac{\rho}{\cos\theta}\sin(\phi-\theta)\cos\psi & - & \rho\sin\phi\cos\psi \\ \frac{\rho}{\cos\theta}\sin(\phi-\theta)\sin\psi & - & \rho\sin\phi\sin\psi \\ \frac{\rho}{\cos\theta}\cos(\phi-\theta) & - & \rho\cos\phi \end{pmatrix} \tag{5.2}$$

The other unit vector, $\mathbf{v}$, can be obtained from the cross product of $\mathbf{u}$ and $\mathbf{OT}$:

$$\mathbf{v} = \mathbf{u} \times \mathbf{OT} \tag{5.3}$$

The angle, ω, at which the line from P to T lies in the plane, relative to the unit vector $\mathbf{v}$, can now be obtained from the dot products $\mathbf{TP} \cdot \mathbf{u}$ and $\mathbf{TP} \cdot \mathbf{v}$. All the parameters defining the line from P to T are now known, and the following integers can be obtained and used, as indices in the accumulator array:

$$\psi_{index} = \left\lfloor \frac{\psi}{\Delta\psi} \right\rfloor, \quad \phi_{index} = \left\lfloor \frac{\phi}{\Delta\phi} \right\rfloor, \quad \rho_{index} = \left\lfloor \frac{\rho}{\Delta\rho} \right\rfloor, \quad \omega_{index} = \left\lfloor \frac{\omega}{\Delta\omega} \right\rfloor$$

A special case occurs when P lies on the line that passes through O and T. In this situation, it is not possible to obtain *one* line, as all lines in the tangent plane are candidates. In this case, all the accumulator cells for the given (ψ, ϕ, ρ) triple is incremented.

The ranges of the four parameter values deserve some attention. It is imperative that the these four values uniquely define a line. If the zenith angle, ϕ, and the azimuth angle, ψ, have ranges $(0, \pi)$ and $(0, 2\pi)$ or vice versa, then a negative normal distance will impose redundant votes on the accumulator array. In this implementation, the angles ψ, ϕ and ω are given a range $(0, \pi)$. ω is simply in this range because a wider range would introduce redundancies. The range of ω is dictated by the intolerable fact that vectors in different directions along the same line would cast two different votes. As the normal distance, ρ, is dependent on the dimensions of the input image, giving it a wider range would only tie the accumulator properties closer to the input image. It is believed that angular precision is beneficial over distance precision.

Backmapping as described in section 4.2 is naturally extended to account for the 3-D line representation by incrementing both orientation angles ψ and ϕ, while obtaining ρ and ω.

Algorithm 2 HOUGH3D

Require: $n_\psi, n_\phi, n_\rho, n_\omega > 1$
Require: Binary input image, $\mathbf{I}$, with diagonal, $d_\mathbf{I}$, centred at O
  Clear accumulator, $\mathbf{A}$, with dimensions $(n_\psi, n_\phi, n_\rho, n_\omega)$
  $\Delta\psi \leftarrow \pi/(n_\psi - 1)$
  $\Delta\phi \leftarrow \pi/(n_\phi - 1)$
  $\Delta\rho \leftarrow d_\mathbf{I}/(n_\rho - 1)$
  $\Delta\omega \leftarrow \pi/(n_\omega - 1)$
  $\rho_{min} \leftarrow -d_\mathbf{I}/2$
  for all $P : (p_x, p_y, p_z) > 0 \in \mathbf{I}$ **do**
    for $i_\psi \leftarrow 0$ to $n_\psi - 1$ **do**
      for $i_\phi \leftarrow 0$ to $n_\phi - 1$ **do**
        $\psi \leftarrow i_\psi \Delta\psi$
        $\phi \leftarrow i_\phi \Delta\phi$
        Calculate ρ
        Calculate ω
        $i_\omega \leftarrow \lfloor \omega/\Delta\omega \rfloor$
        $\mathbf{A}(i_\psi, i_\phi, i_\rho, i_\omega) \leftarrow \mathbf{A}(i_\psi, i_\phi, i_\rho, i_\omega) + 1$
      end for
    end for
  end for
  return $\mathbf{A}$

Chapter 6

Implementation

This chapter focuses on how the solution presented in chapter 5 was implemented.

6.1 Pre-processing

After having obtained the black top-hat transform of the data set with the MATLAB prototype implemented in the pre-project [3], the morphological pre-processing is extended to simplify the data set. This is done by computing the skeleton. Because of the challenges associated with obtaining a 3-D skeleton, far fewer implementations exist than for 2-D skeletons. In this situation, the desired skeleton is of the medial axis type. One such implementation is available in Digital Topology [43], an extension to the Insight Toolkit [44]. This implementation employs concepts such as connectivity and topological numbers that allows for simple homeotopic thinning [43]. This thinning process is reiterated until the result is the same as the input. Example source code was provided with Digital Topology. This can be seen in Figure 6.1. The weights of the chamfer distance transformation were set to $(24, 34, 24)$.

Even after skeletonization, it is not expected that the data set will expose the general fibre orientation trend of the inner wall. For this, the input to the skeletonizer is too complex. To obtain a reduced skeleton, 3-D pruning is employed. To perform 3-D pruning, the simple algorithm of Algorithm 1 was prototyped in MATLAB. Parts of the MATLAB source code is available in Figure 6.2, with structuring element generation omitted. The data is read using a function `rawRead()`, written by Torgil Rise. The pruning procedure is binary, so a call to the function `logical()` is necessary to convert the input to binary form. The structuring elements are generated and stored in the `strel` cell array. This generation has been omitted for brevity, but is described in section 5.1. The procedure copies the input data to `work` and successively updates its content with the thinned version, for each structuring element

Skeletonization using Digital Topology

```
1  #include <iostream>
2
3  #include <itkImageFileReader.h>
4  #include <itkImageFileWriter.h>
5  #include <itkImage.h>
6  #include <itkAnalyzeImageIO.h>
7  #include <itkJPEGImageIO.h>
8  #include <itkPNGImageIO.h>
9  #include "itkRescaleIntensityImageFilter.h"
10
11 #include "itkSkeletonizeImageFilter.h"
12 #include "itkChamferDistanceTransformImageFilter.h"
13
14 int main(int argc, char** argv)
15 {
16     try
17     {
18         unsigned int const Dimension = 3;
19         typedef itk::Image<unsigned char, Dimension> ImageType;
20         itk::ImageFileReader<ImageType>::Pointer reader = itk::ImageFileReader<
               ImageType>::New();
21         reader->SetFileName(argv[1]);
22         reader->Update();
23         std::clog << "Image read" << std::endl;
24
25         typedef itk::SkeletonizeImageFilter<ImageType, itk::Connectivity<Dimension,
               0> > Skeletonizer;
26
27         typedef itk::ChamferDistanceTransformImageFilter<ImageType, Skeletonizer::
               OrderingImageType> DistanceMapFilterType;
28         DistanceMapFilterType::Pointer distanceMapFilter = DistanceMapFilterType::
               New();
29         unsigned int weights[] = { 24, 34, 24 };
30         distanceMapFilter->SetDistanceFromObject(false);
31         distanceMapFilter->SetWeights(weights, weights+3);
32         distanceMapFilter->SetInput(reader->GetOutput());
33         distanceMapFilter->Update();
34         std::clog << "Distance map generated" << std::endl;
35
36         Skeletonizer::Pointer skeletonizer = Skeletonizer::New();
37         skeletonizer->SetInput(reader->GetOutput());
38         skeletonizer->SetOrderingImage(distanceMapFilter->GetOutput());
39         skeletonizer->Update();
40         ImageType::Pointer skeleton = skeletonizer->GetOutput();
41         std::clog << "Skeleton generated" << std::endl;
42
43         itk::ImageFileWriter<ImageType>::Pointer writer = itk::ImageFileWriter<
               ImageType>::New();
44         writer->SetFileName(argv[2]);
45         writer->SetInput(skeleton);
46         writer->Update();
47
48         std::cout << "PASSED" << "\n";
49     }
50     catch(std::exception & e)
51     {
52         std::cerr << "FAILED : " << e.what() << std::endl;
53         return EXIT_FAILURE;
54     }
55
56     return EXIT_SUCCESS;
57 }
```

Figure 6.1: Skeletonization using Digital Topology

in `strel`. The number of iterations is controlled by the `iterations` parameter. The output is finally converted to one byte per voxel, and written out as binary data.

3-D pruning MATLAB prototype

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Prototype file, demonstrating pruning on voxel grid skeleton
% Knut Vidar Siem, Spring 2007
%
%
  clear all;
% Parameters:
%
  inputFileName = 'skeleton.img';
  inputDims = [256 517 256]; % Length of input, dimension 1
  iterations  = 3; % number of sequential thinning iterations
  outputFileName = 'pruned.img';
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Read data
disp '--- Pruning prototype ---';
data = rawRead(inputFileName, inputDims(1),inputDims(2),inputDims(3));
disp 'Read data.';
data = logical(data);
disp 'Data is now binary.';

%% Create structuring elements
% ... store structuring elements in 'strel' cell array

%% Prune data set
numStrels = size(strels, 1);
work = data;
for i=1:iterations
    for strelIndex = 1:numStrels
       work = work & ~bwhitmiss(work, strels{strelIndex});
       strelIndex
    end
    i
end
disp 'Finished pruning';

%% Write output file
output = work;
if islogical(output)
    output = output*127;
end
fid = fopen(outputFileName,'wb');
fwrite(fid, output);
fclose(fid);
disp 'File output complete.';

clear output;
```

Figure 6.2: 3-D pruning MATLAB prototype

6.2 Representation

The mathematical foundations for the extension of the Hough transformation have been given in section 5.2. The question of how to represent a line in 3-D space has been answered. This section will focus on how this extended representation, higher dimensionality and added

complexity affects the rationale of the 2-D Hough transformation. Also given attention here, is how the computations can be done in an efficient manner.

The information necessary to obtain the Hough transform of an image is, besides the image itself, some discretization definition of the parameter space. There are two possibilities; the parameter space can be discretized at a specific resolution, or by the available resources to store the accumulator array, namely by supplying the desired dimensions. In either case, resources should not be wasted. The input image is initially of points with all their coordinates positive. To use as large a portion of the accumulator array as possible, it is suggested that the image is centred at origo [41]. This implementation expects the dimensions of the accumulator array to be specified with the four values, n_ψ, n_ϕ, n_ρ and n_ω. It is suggested that the dimensions of a 2-D accumulator array is as follows [41]:

$$n_\theta = N, \quad n_\rho = 2N$$

Extended to 3-D, this yields

$$n_\psi = N, \quad n_\phi = N, \quad n_\rho = 2N, \quad n_\omega = N$$

Not included in Algorithm 2, is the suggested quantization scheme for digital straight line segments is to use an upper bound straight band [41]. In the Hough transformation, the thickness of this band is suggested to be dependent on the angle, α, to one of the axes. The thickness is given by a function, μ, which is defined as follows:

$$\mu(\alpha) = \begin{cases} \sin(\alpha) & \pi/4 \leq \alpha \leq 3\pi/4 \\ |\cos(\alpha)| & \text{otherwise} \end{cases} \tag{6.1}$$

The Hough transformation is implemented in Java, a language becoming increasingly popular for numerical computing [45]. Measures are taken, to not unnecessarily increase computation time. The primary data types `int` and `double` are used for data storage, as objects tend to bring with them a certain overhead. Another important issue is that of memory allocation, and the time it takes. Even though temporary help vectors ought to be defined inside the loop they are used in, to maintain a clean scope, the cost of reallocation advises against this. On a final note, trigonometrical calculations are also reused to some extent.

The implementation of the Hough transformation process, as described in section 4.2 is shown in source code in Figures 6.3 and 6.4. This procedure is run for every point, `point`, in the image. It operates in two modes, normal, and backmapping, as illustrated by the parameter `isBackmappingRun`. The procedure must be run with this parameter `false` before running it with `true`. The next paragraph will describe the initial run, and the one following it will describe the backmapping run.

First, common vectors are allocated to avoid reallocation at every angular iteration. Second, helper variables for backmapping functionality are initialized. Third, the iteration over the,

Hough 3-D point processing (part 1)

```
1  private final void processPoint(int[] point, boolean isBackmappingRun) {
2      // Allocate vectors only once
3      double[] t = new double[3];
4      double[] tBand = new double[3];
5      double[] u = new double[3];
6      double[] v = new double[3];
7
8      // Backmapping variables
9      int maxFrequency = 0;
10     int maxPsiIndex = -1;
11     int maxPhiIndex = -1;
12     int maxRhoIndex = -1;
13     int maxOmegaIndex = -1;
14
15     for (int psiIndex = 0; psiIndex < this.accumulatorSizePsi; psiIndex++) {
16         for (int phiIndex = 0; phiIndex < this.accumulatorSizePhi; phiIndex++) {
17
18             // define trigonometric factors (for better performance)
19             final double psi = psiIndex*this.dPsi;
20             final double phi = phiIndex*this.dPhi;
21             final double cosPhi = Math.cos(phi);
22             final double sinPhi = Math.sin(phi);
23             final double cosPsi = Math.cos(psi);
24             final double sinPsi = Math.sin(psi);
25
26             final double rho = (
27                 point[0]*sinPhi*cosPsi + point[1]*sinPhi*sinPsi + point[2]*cosPhi
28             )/(
29                 Math.pow(sinPhi*cosPsi, 2) + Math.pow(sinPhi*sinPsi, 2) + Math.pow(
                        cosPhi, 2)
30             );
31             final int rhoIndex = (int)(rho/this.dRho) + this.accumulatorCenterRho;
32
33             // define point T and vector, tp, from T to P
34             t[0] = rho*sinPhi*cosPsi;
35             t[1] = rho*sinPhi*sinPsi;
36             t[2] = rho*cosPhi;
37
38             if (point[0]==t[0] && point[1]==t[1] && point[2]==t[2]) {
39                 // special (rare) case: P==T. all omega are candidates
40                 for (int omegaIndex=0; omegaIndex<this.accumulatorSizeOmega;
                        omegaIndex++) {
41                     this.accumulator[psiIndex][phiIndex][rhoIndex][omegaIndex]++;
42                 }
43
44             } else {
45                 // define tangent plane unit vectors u and v
46                 // define point U in the tangent plane (u = TU = OU-OT)
47                 // U: (phi_u, psi, rho_u)
48                 final double phi_u = phi - this.dPhi;
49                 final double rho_u = rho/Math.cos(this.dPhi);
50                 u[0] = rho_u*Math.sin(phi_u)*cosPsi - t[0];
51                 u[1] = rho_u*Math.sin(phi_u)*sinPsi - t[1];
52                 u[2] = rho_u*Math.cos(phi_u)        - t[2];
```

Figure 6.3: Hough 3-D point processing (part 1)

Hough 3-D point processing (part 2)

```
54                  //  v = u x t
55                  v[0] = u[1]*t[2] - u[2]*t[1];
56                  v[1] = u[2]*t[0] - u[0]*t[2];
57                  v[2] = u[0]*t[1] - u[1]*t[0];
58
59                  if (isBackmappingRun) {
60                      final double omega = this.calculateOmega(t, point, u, v);
61                      final int omegaIndex = (int)(omega/this.dOmega);
62
63                      final int frequency = this.accumulator[psiIndex][phiIndex][
                            rhoIndex][omegaIndex];
64                      if (frequency > maxFrequency) {
65                          maxFrequency = frequency;
66                          maxPsiIndex = psiIndex;
67                          maxPhiIndex = phiIndex;
68                          maxRhoIndex = rhoIndex;
69                          maxOmegaIndex = omegaIndex;
70                      }
71
72                  } else {
73                      // scalar product of t and [0 1 0] to find mu
74                      final double sty = t[1]/rho;
75                      double alpha = Math.acos(sty);
76                      if (alpha > Math.PI/2) {
77                          alpha -= Math.PI/2;
78                      }
79                      double mu = Math.abs(sty); // mu = |cos(alpha)|
80                      if (Math.PI/4 <= alpha && alpha <= 3*Math.PI/4) {
81                          mu = Math.sin(alpha);
82                      }
83
84                      final double rhoLower = rho - mu/2.0;
85                      final double rhoUpper = rhoLower + mu;
86                      int rhoLowerIndex = (int) Math.ceil(rhoLower/this.dRho) + this.
                            accumulatorCenterRho;
87                      int rhoUpperIndex = (int) Math.ceil(rhoUpper/this.dRho -1) +
                            this.accumulatorCenterRho;
88                      for (int rhoBandIndex = rhoLowerIndex; rhoBandIndex <=
                            rhoUpperIndex; rhoBandIndex++) {
89                          final double rhoBand = (rhoBandIndex-this.
                                accumulatorCenterRho)*this.dRho;
90                          tBand[0] = rhoBand*sinPhi*cosPsi;
91                          tBand[1] = rhoBand*sinPhi*sinPsi;
92                          tBand[2] = rhoBand*cosPhi;
93                          final double omegaBand = this.calculateOmega(tBand, point, u
                                , v);
94                          final int omegaBandIndex = (int)(omegaBand/this.dOmega);
95                          this.accumulator[psiIndex][phiIndex][rhoBandIndex][
                                omegaBandIndex]++;
96                      }
97                  }
98              }
99          }
100     }
101     if (isBackmappingRun) {
102         if (maxPsiIndex>0 && maxPhiIndex>0 && maxRhoIndex>0 && maxOmegaIndex>0) {
103             this.accumulatorBackmapped[maxPsiIndex][maxPhiIndex][maxRhoIndex][
                    maxOmegaIndex]++;
104         }
105     }
106 }
```

Figure 6.4: Hough 3-D point processing (part 2)

zenith angle, `phi`, and the azimuth angle, `psi`, is started. The exact angles are retrieved and the normal distance, `rho`, calculated. Being that the range of `rho` is centered at 0, its corresponding index in the accumulator array must be calculated with a shift. The tangent plane normal vector, `t`, is calculated and a special test is performed to handle the situation where the image point, lies on the line extending from origo, and in the direction of `t`. If this is the case, then `accumulator` is updated at `[psi][phi][rho][*]`, where `*` indicates the whole `omega` range. After handling this rare exception, the ordinary process continues. The unit vectors `u` and `v` are obtained with a rotation and a cross product. As this is not a backmapping run, the next step is to calculate the width, `mu`, of the upper bound straight band, the quantization scheme introduced above. An iteration over the band is performed, calculating a `rhoBand` and `omegaBand` value in each time. These values are discretized and used as coordinates at which to increment the `accumulator` array.

In the case of a backmapping run, a second pass, the only difference is that instead of dealing with the upper bound straight band, an `omega` value is calculated for the exact `rho` value. The four current discretized parameters are then compared with the already registered maximal values. At the end of the angular iteration process these four parameter values are used as indices in `accumulatorBackmapped`.

6.3 Visualization

It is quite difficult to judge the success of the procedures above without some form of visualization. In pre-processing, various volume viewers are used to give visual feedback. This proved difficult in the case of the Hough transform, where it is necessary to plot the lines back in the image space. A simple, flexible interface to 3-D visualization is desired. The OpenGL Application Programming Interface (API) is chosen as it is a good alternative, available for Java through Java Binding for the OpenGL API (JOGL).

Two viewports are used, one for the image space and the other for the parameter space. Figure 6.5 shows the `display` method that render both scenes. The image is visualized using `GL_POINTS` or a prototyped voxel primitive. The parameter space, being four-dimensional, is somewhat difficult to visualize. In this implementation, 3-D slices are displayed over time, with ω as the temporal variable. The detected lines are displayed in the image space using rotation and translation transformations [46, 47]. Figure 6.6, which should be read in light of Figure 5.2, shows how these transformations are performed.

Main render function

```
1  public void display(GLAutoDrawable gLDrawable) {
2      final GL gl = gLDrawable.getGL();
3      int windowWidth = gLDrawable.getWidth();
4      int windowHeight = gLDrawable.getHeight();
5
6      gl.glClear(GL.GL_COLOR_BUFFER_BIT);
7      gl.glClear(GL.GL_DEPTH_BUFFER_BIT);
8
9      // Image space viewport
10     gl.glViewport(0, 0, windowWidth / 2, windowHeight);
11     gl.glLoadIdentity();
12     gl.glRotated(-90.0, 1.0, 0.0, 0.0);
13     gl.glTranslated(0.0, 100.0, -10.0);
14     gl.glRotated(rotateT, 0.0, 0.0, 1.0); // z-axis rotation animation
15     this.drawAxes(gl, 100.0);
16     this.drawImage(gl);
17     this.drawLines(gl, 100.0);
18
19     // Parameter space viewport
20     gl.glViewport(windowWidth / 2, 0, windowWidth / 2, windowHeight);
21     gl.glLoadIdentity();
22     gl.glRotated(-90.0, 1.0, 0.0, 0.0);
23     gl.glTranslated(0.0, 40.0, -10.0);
24     gl.glRotated(rotateT, 0.0, 0.0, 1.0); // z-axis rotation animation
25     if (this.tempSleepCounter < this.tempSleep) {
26         this.tempSleepCounter++;
27     } else {
28         this.tempSleepCounter = 0;
29         this.tempValue = (this.tempValue + 1) % this.tempMaxValue;
30     }
31     this.drawAxes(gl, 100.0);
32     this.drawParameterImage(gl);
33
34     this.rotateT += 0.02;
35 }
```

Figure 6.5: Main render function

Drawing Hough lines in image space

```
1  private void drawLines(GL gl, double length) {
2      // ...
3      for (double[] line : this.lines) {
4          double psi = line[0];
5          double phi = line[1];
6          double rho = line[2];
7          double omega = line[3];
8
9           gl.glPushMatrix();
10
11          gl.glRotated(phi, -Math.sin(2*Math.PI*psi/360), Math.cos(2*Math.PI*psi/360),
                0.0);
12          gl.glRotated(psi, 0.0, 0.0, 1.0);
13          gl.glTranslated(0.0, 0.0, rho);
14          gl.glRotated(omega, 0.0, 0.0, Math.signum(rho)*1.0);
15
16          gl.glBegin(GL.GL_LINES);
17          // ...
18          gl.glVertex3d(0.0, -length/2.0, 0.0);
19          gl.glVertex3d(0.0, length/2.0, 0.0);
20          gl.glEnd();
21
22          gl.glPopMatrix();
23      }
24 }
```

Figure 6.6: Drawing Hough lines in image space

Chapter 7

Results and discussion

This chapter gives the results of the implementation. The results are followed by a discussion.

7.1 Results

As mentioned, the input to the procedures of this thesis is the black top-hat transform of the LV cavity data set. The results of this transformation is shown in Figure 7.1. It was obtained with a 3-by-3 solid structuring element after a thresholding operation at level 90 of 127. As can be seen in the figure, the obtained wall detail varies from highly complex ridges to open holes. The volume was visualized in DynamicImager [36].

Figure 7.2 shows the result of the skeletonization process performed by Digital Topology [43]. The skeleton was rendered in DynamicImager [36]. It does not resemble a typical medial axis skeleton in this volume rendering, but a closer inspection reveal the sparseness of the volume. Figure 7.3, rendered by GEHC MicroView [48], show three slices through the model that illustrates this sparseness.

The effect of the pruning process is shown in Figure 7.4. This pruned skeleton was rendered by DynamicImager [36]. Upon comparing this image with the skeleton of Figure 7.2, taking into comparison the different orientation of the model, one can see the skeleton has in fact been reduced.

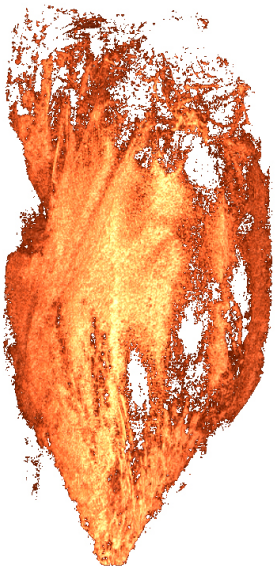

Figure 7.1: LV inner wall, top-hat transform

Figure 7.2: Inner wall skeleton

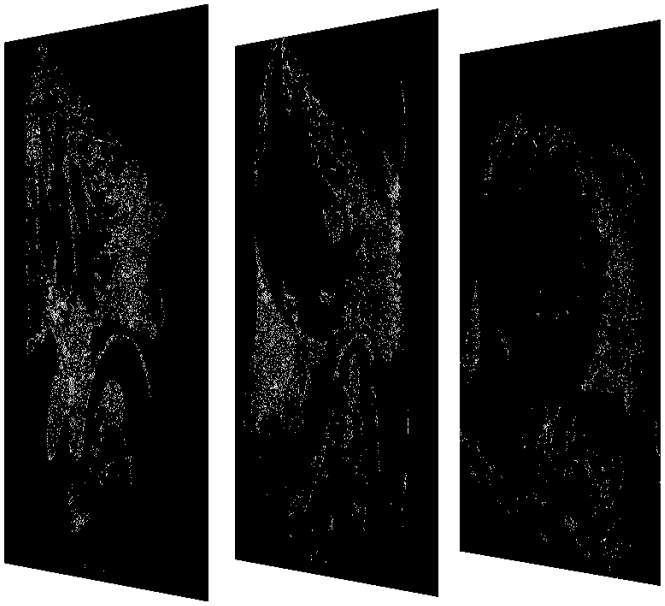

Figure 7.3: Inner wall skeleton slices

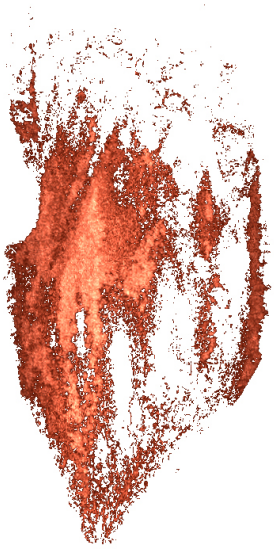

Figure 7.4: Pruned skeleton, seven iterations

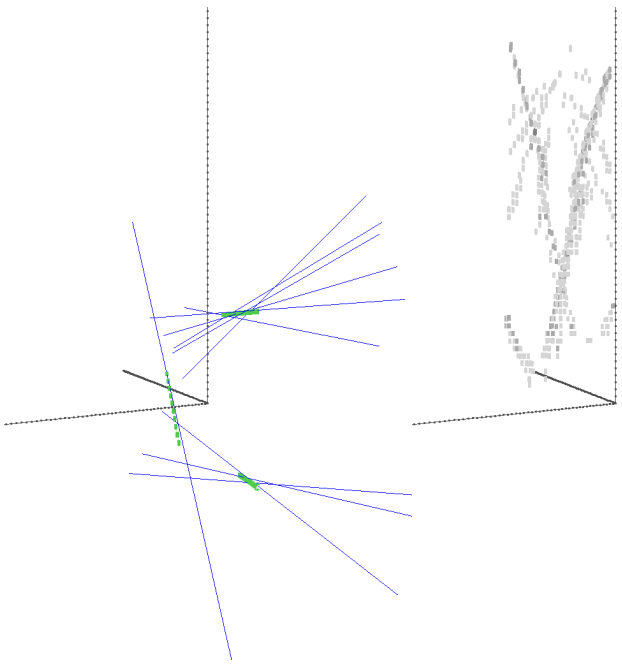

Figure 7.5: Test data, no backmapping, ten most well-supported lines

The results of the Hough transformation are presented as screenshots of the prototype visualizer, with the image space and detected lines on the left, and the parameter space on the right. The simple test image is three short line segments, coloured green, and the detected lines, blue. The intensities of the parameter space have been inverted due to the use of white background; white indicate low frequency, and black indicate high frequency. Figure 7.5 shows the ten most well-supported lines in the image, and Figure 7.6 the five most well-supported. Both of these are runs without backmapping. Figure 7.7 shows the same image, also subjected to backmapping. On this simple image, backmapping is only somewhat visible, but a view of the parameter space reveals that most of the noise has been removed.

Running the Hough transformation on the region $(30 \rightarrow 110, 50 \rightarrow 120, 170 \rightarrow 210)$, and plotting the 100 most well-supported lines yields Figure 7.8. It is difficult to judge whether this is precisely correct, or not. Because of this, the transform is not run on the entire LV cavity data set.

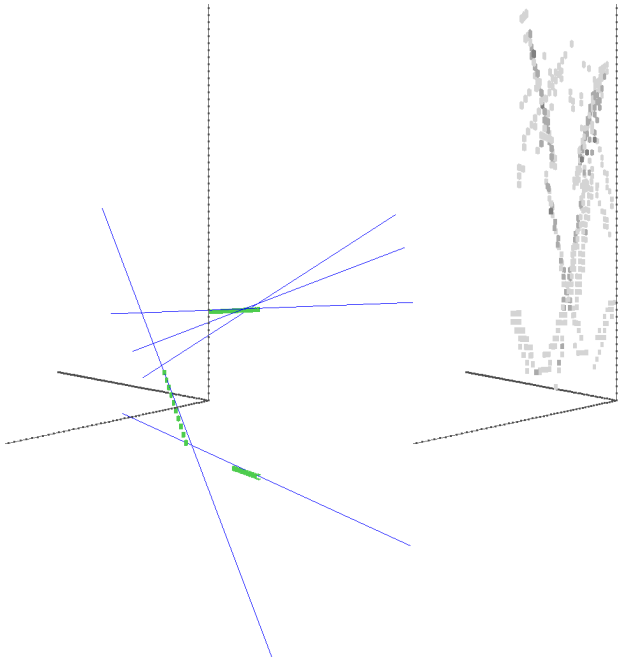

Figure 7.6: Test data, no backmapping, five most well-supported lines

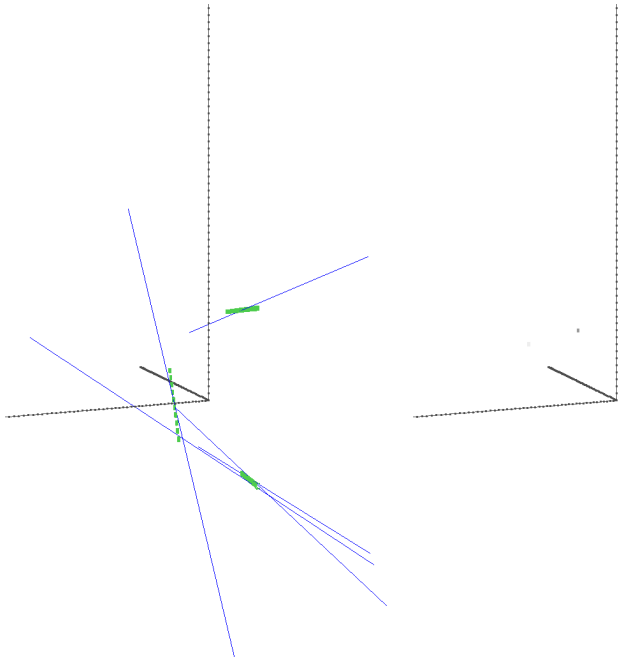

Figure 7.7: Test data, backmapping, five most well-supported lines

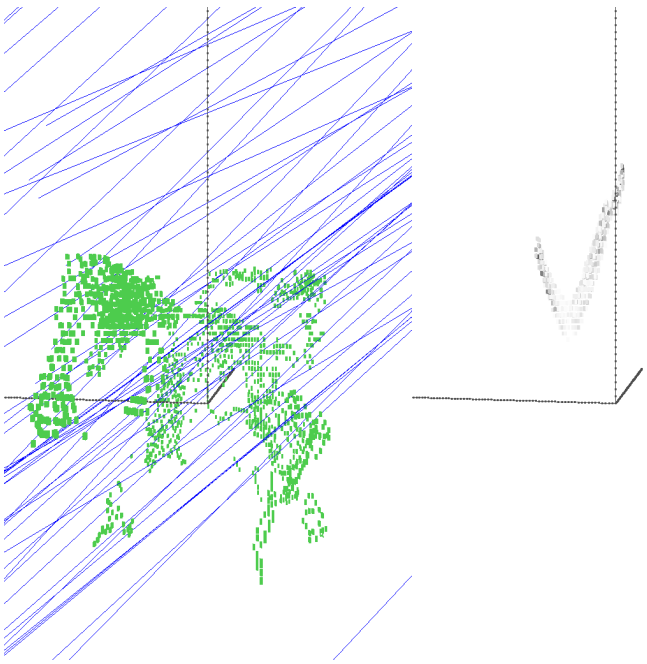

Figure 7.8: LV data, no backmapping, 100 most well-supported lines

7.2 Discussion

The input top-hat transform has large holes due to low degree of surface variation on that part of the cavity wall. This is in accordance with Bergheims description on the variation of the trabeculae carneae. A helical pattern is visible towards the apex. The most significant characteristic of the skeleton is that it is very branched, almost sponge-like. The pruning process did not trim the skeleton as much as expected, meaning that the data set quickly converged. This is surprising for such a large skeleton. One possible explanation is that some of the pruning structuring elements have been left out. Another possibility is that the skeleton is filled with closed loops, which the structuring elements cannot prune away without changing the topology of the volume.

The 3-D Hough transformation seems to produce fair results, indicating that the theoretic work is justified. It does, however, seem to have some erroneous processing of points, perhaps due to round-off errors. To aid debugging, or for educational purposes, the actual computation of the transform can be gradually visualized over time. Another observation is that because the accumulator array is 4-D, an expansion greatly impacts memory consumption; doubling the resolution in each dimension requires $2^4 = 16$ times more memory. Interestingly, increasing the image size does not affect memory usage, only run time. That is if the procedure accepts accumulator dimensions as input — which is the case in this implementation — and not accumulator resolution. A parallel can be drawn to in-place sorting algorithms,

which “transforms a data structure using a small, constant amount of extra storage space” [49].

As mentioned in section 3.2, the fibre orientation is relatively constant locally in the heart wall. The chosen approach with the Hough transformation emphasizes global lines, whereas what is desired is a representation of shorter line segments, generally in the same direction in smaller regions of the data set. Subdivision of the data set into smaller cubes, is proposed as an improvement. Running the Hough transformation on each of these smaller cubes would yield a set of lines supported by the image points in the cube. These lines would therefore follow the desired local trend, and can be clipped by the cube, to obtain a clearer picture.

Chapter 8

Conclusion

In this thesis, an introduction to the human heart, along with techniques to image it, has been given. Background material and related literature on muscle fibres and their properties has also been presented.

From raw data of a highly detailed MRI scan, inner wall detail of the human LV has been isolated using mathematical morphology. 3-D medial axis skeletonization has been performed with the goal of obtaining a minimal representation. A 3-D pruning prototype has been implemented to simplify the obtained skeleton. An extension of the Hough transformation to 3-D has been designed and implemented to detect the general trend of the simplified skeleton. Such an extension to three dimensions has seemingly never been developed before. The extended Hough transformation performs well on test data sets, but does not work properly on the entire ventricle cavity data set. It is suggested that the transformation is applied to smaller regions of the image.

With knowledge of the complex growth process of the heart, from tubes into a double pump, and the detailed studies of Dr. Torrent-Guasp [1], it seems likely that the general trend of the fibre orientation is what is manifested in the trabeculae carneae. These features are locally parallel. This is also a trait of the heart wall muscle fibres. It is important to note that although the isolated wall detail is scattered and discontinuous, all muscle fibres are with no exceptions continuous. They are also smooth and make no abrupt change in orientation. The chosen approach did not succeed in obtaining the fibre orientation, but the notion of using short line segments as a stepping stone to better representations, is somewhat reminiscent of the tensor field employed in DT-MRI. Such a representation might serve as a substitute in situations where diffusion tensor technology is not available.

Chapter 9

Further work

This chapter will provide suggestions for further work on the subject of obtaining and representing muscle fibres.

The 3-D Hough transformation is an interesting procedure that can be improved by connectivity analysis. If computed on subdivisions on the data set, it can find lines in the direction of the local trend. These lines can be used as guides for seeded parametric curves to reconstruct fibre trajectories. Depending on the number of resulting curves, an interpolation process may be implemented to generate intermediate fibre trajectories. If a layer of curves is obtained, subsequent layers can be added as the orientation angle is gradually altered, guided perhaps by one of Equations 3.1 or 3.2.

The concept of guided, seeded curves is fully conceptualized in the technique known as DT-MRI. By measuring how water diffuses in matter, elongated structures such as muscle fibres or neurons, may be reconstructed to astonishing detail. Zhukov and Barr applied DT-MRI to reconstruct the cardiac fibres of the heart [50]. Their research both demonstrates the potential of the technique and provides evidence of the helical heart [1], and as such, Figure 9.1 [50] brings suitable closure.

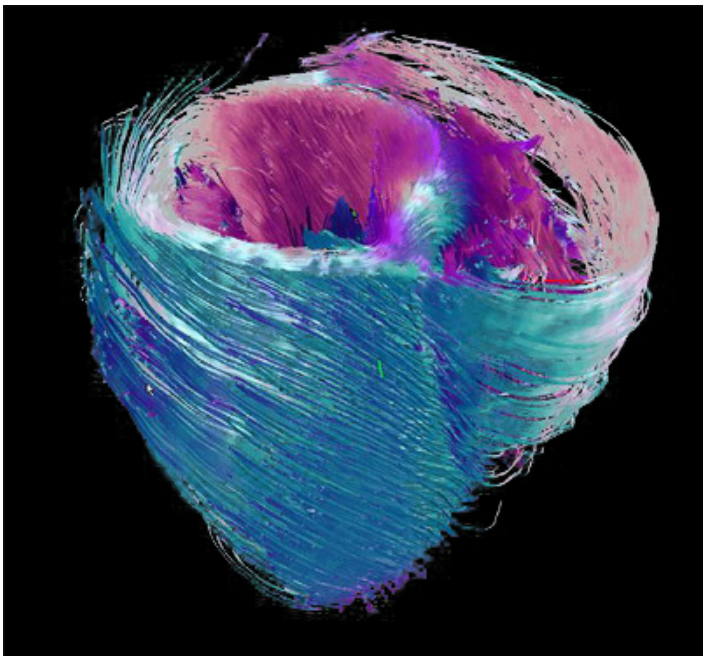

Figure 9.1: Fibre reconstruction of heart wall obtained by DT-MRI (Leonid Zhukov and Alan H. Barr)

Appendix A

Mathematical morphology

This appendix will give an introduction to basic mathematical morphology. All morphological operations can be defined in terms of set theory. The operations *dilation* and *erosion* are used as bases for more complex operations.

A.1 Dilation and erosion

The two fundamental morphological operations are dilation and erosion. They are binary operations that operate on the first set using the latter, the latter being called the *structuring element*. The structuring element controls the magnitude of change and its shape. Dilation, as the name indicates, results in the new set being larger than the original set, unless the origo of the structuring element is displaced . Conversely, with the same condition, erosion shrinks a set.

From [34]: The dilation of A by B, denoted by $A \oplus B$, is defined as

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \varnothing\}. \tag{A.1}$$

The erosion of A by B, denoted by $A \ominus B$, is defined as

$$A \ominus B = \{z|(B)_z \subseteq A\}. \tag{A.2}$$

Erosion and dilation are dual operations. Consider the following: one of the operations is performed on a set, A, with a structuring element, B. Now consider the other operation performed on the complement, $A^{\mathcal{C}}$, of the original set with the reflected structure element,

$\hat{B}$. The result of this operation is the same as the complement of the former. Formally this can be expressed in the following manner:

$$(A \ominus B)^{\mathcal{C}} = A^{\mathcal{C}} \oplus \hat{B} \tag{A.3}$$

A.2 Opening and closing

Opening and *closing* are composite operations in that they are defined using one dilation and one erosion operation. They both have a smoothing effect on the set on which they operate, but differ in how the smoothing is done. As stated in [34], the following defines the opening and closing operations respectively:

$$A \circ B = (A \ominus B) \oplus B \tag{A.4}$$

$$A \bullet B = (A \oplus B) \ominus B \tag{A.5}$$

As can be seen, the opening operation is an erosion followed by a dilation. This results in the removal of protruding details on the set contour that can be contained in the structuring element [34]. The closing operation, a dilation followed by an erosion, fills small gaps. These operations, such as dilation and erosion, are also duals.

Appendix B

Acronyms

API Application Programming Interface

CAT Computerized Axial Tomography

CPU Central Processing Unit

CT Computerized Tomography

DT-MRI Diffusion Tensor Magnetic Resonance Imaging

fMR Functional MR

IDI Department of Computer and Information Science

IMF Department of Mathematical Sciences

ISB Department of Circulation and Medical Imaging

JOGL Java Binding for the OpenGL API

LV left ventricle

MRI Magnetic Resonance Imaging

NMR Nuclear Magnetic Resonance

NTNU Norwegian University of Science and Technology

RF Radio Frequency

RV right ventricle

SINTEF The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology

Bibliography

[1] F. Torrent-Guasp, J. M. Kocica, A. F. Corno, M. Komeda, F. Carreras-Costa, A. Flotats, J. Cosin-Aguillar, and H. Wen, "Towards new understanding of the heart structure and function," *European Journal of Cardio-thoracic Surgery*, vol. 27, pp. 191–201, 2005.

[2] The Helical Heart Company LLC, "The helical heart." `http://helicalheart.com`. Retrieved 2006.12.21.

[3] K. V. Siem, "Approximating fiber orientation in left ventricular human heart wall." Project work, 2006.

[4] H. Gray, "Anatomy of the Human Body." `http://www.bartleby.com/107/`. Retrieved 2006.12.17.

[5] C. Bianco, "How Your Heart Works." `http://health.howstuffworks.com/heart.htm`. Retrieved 2007.06.09.

[6] Wikipedia, "Muscle." `http://en.wikipedia.org/wiki/Muscle`. Retrieved 2007.06.09.

[7] S. I. Rabben, F. Irgens, and B. Angelsen, "Equations for estimating muscle fiber stress in the left ventricular wall," *Heart Vessels*, vol. 14, pp. 189–196, 1999.

[8] Caplex, "hjertet." `http://www.caplex.no/Web/ArticleView.aspx?id=9314788`. Retrieved 2006.11.13.

[9] The Franklin Institute Science Museum, "The Heart: An Online Exploration." `http://www.fi.edu/biosci/structure.html`. Retrieved 2006.11.14.

[10] G. J. Tortora, *Introduction to the Human Body*. Benjamin Cummings, fourth ed., 1997.

[11] T. Arts, "Cardiac myofiber orientation: model and experiment." `http://www-lmc.imag.fr/lmc-edp/Emmanuel.Maitre/visio/arts/myofib/index.htm`. Retrieved 2006.12.20.

[12] Wikipedia, "Left ventricle." `http://en.wikipedia.org/wiki/Left_ventricle`. Retrieved 2007.06.09.

[13] How Stuff Works, "Introduction to Heart Formation." http://health.howstuffworks.com/adam-200118.htm. Retrieved 2006.12.17.

[14] Caplex, "blodomløpet." http://www.caplex.no/Web/ArticleView.aspx?id=9302569. Retrieved 2006.11.13.

[15] Wikipedia, "Pulmonary circulation." http://en.wikipedia.org/wiki/Pulmonary_circulation. Retrieved 2006.12.21.

[16] Wikipedia, "Systemic circulation." http://en.wikipedia.org/wiki/Systemic_circulation. Retrieved 2006.12.21.

[17] A. Støylen, "Basic ultrasound for clinicians." http://folk.ntnu.no/stoylen/strainrate/Ultrasound/index.html. Retrieved 2006.11.24.

[18] C. C. Freudenrich, "How Ultrasound Works." http://health.howstuffworks.com/ultrasound.htm. Retrieved 2006.11.24.

[19] M. Hofer, *Ultrasound Teaching Manual*. Stuttgart/New York: Georg Thieme Verlag, second ed., 2005.

[20] Imaginis Corporation, "Computed Tomography Imaging." http://www.imaginis.com/ct-scan/. Retrieved 2006.12.18.

[21] J. P. Hornak, "The Basics of MRI." http://www.cis.rit.edu/htbooks/mri, 1997. Retrieved 2006.11.28.

[22] B. Hargreaves, "Introduction to Magnetic Resonance Imaging." http://www-mrsrl.stanford.edu/~brian/intromr/. Retrieved 2006.12.20.

[23] M. NessAiver, "MRI Introduction." http://www.simplyphysics.com/MRIntro.html. Retrieved 2006.11.27.

[24] T. A. Gould, "How MRI Works." http://health.howstuffworks.com/mri.htm. Retrieved 2006.11.28.

[25] D. G. Mitchell and M. Cohen, *MRI Principles*. Philadelphia, Pennsylvania 19106: Saunders, Elsevier Science, second ed., 2004.

[26] Imaginis Corporation, "Magnetic Resonance Imaging." http://www.imaginis.com/mri-scan/. Retrieved 2006.12.18.

[27] S. Zhang, D. H. Laidlaw, and G. Kindlmann, *Visualization Handbook*, pp. 327–340. Orlando, FL, USA: Academic Press, Inc., 2004.

[28] Wikipedia, "Diffusion tensor imaging." http://en.wikipedia.org/wiki/Diffusion_tensor_imaging. Retrieved 2007.06.06.

[29] "Diffusion tensor imaging." http://www.sci.utah.edu/research/diff-tensor-imaging.html. Retrieved 2007.06.06.

[30] V. Hurmusiadis and C. Briscoe, *Functional Imaging and Modeling of the Heart*, vol. 3504 of *Lecture Notes in Computer Science*, ch. A Functional Heart Model for Medical Education, pp. 85–91. Springer Berlin/Heidelberg, 2005.

[31] L. Aurdal, "Forelesningsnotater – Matematisk morfologi." `http://www.aurdalweb.com/matmorf.html`, 2003. Retrieved 2006.12.19.

[32] C. A. Glasbey and G. W. Horgan, *Image analysis for the biological sciences*. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[33] A. W. E. W. 2004 Robert Fisher, Simon Perkins, "Hypermedia image processing reference." `http://homepages.inf.ed.ac.uk/rbf/HIPR2/`, 2004. Retrieved 2006.06.17.

[34] R. C. Gonzales and R. E. Woods, *Digital Image Processing*. Upper Saddle River, New Jersey: Prentice Hall, second ed., 2002.

[35] Wikipedia, "Hough transform." `http://en.wikipedia.org/wiki/Hough_transform`. Retrieved 2007.05.21.

[36] T. AS, "Dynamicimager."

[37] P. Bhattacharya, H. Liu, A. Rosenfeld, and S. Thompson, "Hough-transform detection of lines in 3-d space," *Pattern Recognition Letters*, vol. 21, no. 8, pp. 843–849, 2000.

[38] "Doctor Rob", "Equation of a line in three or more dimensions." `http://mathforum.org/library/drmath/view/51782.html`. Retrieved 2007.05.23.

[39] E. W. Weisstein, "Hyperplane." `http://mathworld.wolfram.com/Hyperplane.html`. Retrieved 2007.05.23.

[40] Wikipedia, "Hyperplane." `http://en.wikipedia.org/wiki/Hyperplane`. Retrieved 2007.05.23.

[41] L. d. F. Costa and R. M. Cesar Jr., *Shape Analysis and Classification*. Boca Raton, Florida: CRC Press LLC, 2001.

[42] Wikipedia, "Polar coordinate system." `http://en.wikipedia.org/wiki/Polar_coordinate_system`. Retrieved 2007.05.21.

[43] J. Lamy, "Digital topology." http://hdl.handle.net/1926/304. Retrieved 2007-06-15.

[44] "National library of medicine insight segmentation and registration toolkit (itk)." http://itk.org. Retrieved 2007-06-15.

[45] S. Palu, "Numeric Computations in Java." `http://www.developer.com/java/other/article.php/10936_1364221_1`. Retrieved 2007.06.15.

[46] D. Hearn and M. P. Baker, *Computer Graphics with OpenGL*. Upper Saddle River, New Jersey: Pearson Prentice Hall, third ed., 2004.

[47] D. Shreiner, M. Woo, J. Neider, and T. Davis, "Opengl®programming guide," 2005.

[48] "MicroView." `http://microview.sourceforge.net/`. Retrieved 2006.12.22.

[49] Wikipedia, "In-place algorithm." `http://en.wikipedia.org/wiki/In-place_algorithm`. Retrieved 2007.06.18.

[50] L. Zhukov and A. H. Barr, "Heart-Muscle Fiber Reconstruction from Diffusion Tensor MRI," in *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, (Washington, DC, USA), p. 79, IEEE Computer Society, 2003.