Technische Universität Graz
Erzherzog-Johann-Universität
Institut für Maschinelles Sehen
und Darstellen

Diplomarbeit aus Telematik

Image-Based Verification of Parametric Models in Heart-Ventricle Volumetry

Martin Urschler

Graz, September 2001

durchgeführt in Zusammenarbeit mit der
Univ.-Klinik f. Radiologie
LKH Graz
o.Univ.-Prof. Dr. Dr.h.c. Rainer Rienmüller

Betreuer:
Univ.-Ass., Dipl.-Ing., Dr. techn. Heinz Mayer
Univ.-Ass., Dipl.-Ing., Dr. techn. Regine Bolter

Begutachter:
O. o. Univ.-Prof., Univ.-Doz., Dr. techn. Dipl.-Ing. Franz Leberl

Acknowledgements

I would like to thank Heinz Mayer for preparing me for this work and for bringing me into contact with the people at the Department for Radiology. I'm thankful that Regine Bolter took over the advising and the correction of this master thesis after Heinz left the institute. The people at the Institute for Machine Vision and Graphics are to be thanked for their personal and technical support. Especially the head, Prof. Franz Leberl, and my colleague Matthias Rüther, who contributed to this work due to our various discussions, should be mentioned.

I'm grateful that Prof. Rainer Rienmüller and his team (Ursula Reiter and Gert Reiter) spared much of their time for my questions. Further I really appreciate the conversations with Prof. Milan Sonka at the workshop in Vail, Colorado, which have shown me some alternative ways to go.

Finally I'd like to thank my girlfriend Lisi for her patience with me during the writing of this work.

Erklärung

Ich versichere, diese Arbeit selbst verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Graz, September 2001

Martin Urschler

Abstract

Measuring the volume of heart ventricles is an important issue in heart disease diagnosis. Therefore radiologists investigate computer tomographic images, searching for representative projections of the left ventricle. In the selected image showing the left ventricle with maximum circumference, a parametric model is established by measuring width and height of the left ventricle shape. These parameters are used to approximate the contour of the ventricle with an ellipse and afterwards to calculate the volume of the ventricle assuming an ellipsoid model.
In this work we develop a tool for comparing estimated volumes calculated by the parametric model with volumes based on techniques from digital image processing. Herein we treat computer tomographic image slices as volumetric data sets and define a volume by a set of segmented images and a given voxel size.
We implement three different segmentation techniques and apply them on 31 medical image data sets. These segmentation techniques are Thresholding, Active Contours (Snakes) and Live Wire, a graph-theoretic approach. The algorithms are explained and the results from the volume estimations are compared to each other as well as with the parametric model.
Furthermore we investigate an automation technique, a Hough transform for ellipse detection to find the ellipse contour in the image showing the maximum ventricle projection.
The diversity of the images makes it necessary to utilize interactive segmentation techniques instead of fully automatic ones. The Live Wire algorithm shows the best segmentation performance of the tested algorithms.

Zusammenfassung

Die Bestimmung des Volumens von Herzkammern (Ventrikeln) ist ein wichtiger Teil im Diagnoseprozess. Zu diesem Zweck untersuchen Radiologen computer-tomografische Abbildungen um repräsentative Projektionen des linken Ventrikels zu erhalten. Das gefundene Bild, welches den linken Ventrikel mit maximalem Umfang zeigt, wird verwendet um ein parametrisches Modell zur Volumsbestimmung anwenden zu können. Dazu werden Längs- und Querachse der ellipsenförmigen Kontur des projizierten Ventrikels gemessen. Diese Parameter werden benutzt um das Volumen des Ventrikels durch Berechnung des Volumens eines zugehörigen Ellipsoids anzunähern.

In der vorliegenden Arbeit wird ein Werkzeug zum Vergleich von Volumsabschätzungen nach dem parametrischen Modell und Abschätzungen mit Hilfe von Techniken aus der digitalen Bildverarbeitung entwickelt. Dabei werden computer-tomografische Schichtbilder als Volumsdatensätze aufgefaßt, ein Volumen wird über einen Satz von segmentierten Schichtbildern und der bekannten Größe der Volumselemente definiert.

Drei verschiedene Segmentierungsmethoden werden implementiert und auf die 31 medizinischen Datensätze angewandt. Diese Segmentierungsmethoden sind Schwellwertbildung, Active Contours (Snakes) und Live Wire, ein graphen-theoretischer Ansatz. Die Algorithmen werden vorgestellt und die Resultate aus der Volumsbestimmung werden miteinander und mit den Ergebnissen des parametrischen Modells verglichen.

Weiters wird eine Automatisierungstechnik untersucht, eine Hough Transformation zur Detektion der Ellipsenkontur in jenem Schichtbild mit maximal projizierten Ventrikelumfang.

Die Unterschiedlichkeit der Bilder in den Datensätzen macht es notwendig interaktive anstatt automatischer Segmentierungstechniken anzuwenden. Dabei schneidet der Live Wire Algorithmus am besten von den untersuchten Segmentierungsalgorithmen ab.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Measuring the volume of left and right ventricle is an important issue in heart disease diagnosis. The *ventricles* are the two larger chambers of the heart (left and right *atrium* are the other two). They are responsible for pumping blood through the body during a heart beat. Both ventricles are very important because their muscles have to generate the pressure necessary for pumping. Therefore it is vital to detect diseases affecting both ventricles' abilities to contract and relax efficiently and especially the left ones'. (See Section 2.3.1 for details on anatomical aspects and Figure 1.1a for a schematic.) Radiologists as well as cardiologists are able to diagnose people with such kinds of diseases by determining the left-ventricular volume at two certain points of time. The ratio of these two volumes is an indicator for diseased hearts.

For this reason it is necessary to somehow visualize the heart and its geometry. Opening up the chest of a patient to measure the volume is of course not realistic. Therefore so-called non-invasive techniques have to be used for visualization, like e.g. digital subtraction angiography[1], ultra-sound imaging[1], computer-tomographic (CT) imaging[1][57] or magnetic-resonance (MR) imaging[1]. With the introduction of computer-tomographic and magnetic-resonance imaging into the field of cardiac visualization and volume estimation suddenly far more depictive results could be achieved compared to former techniques. This is one of the reasons why these imaging techniques are widespread nowadays.

In case of cardiac volume estimation one among many possible ways to benefit from computer-tomographic imaging is the so-called *Two-Axes Method by Greene*[35]. Physicians investigate computer-tomographic images, looking for representative projections of the left ventricle. After selecting the image showing the left ventricle with maximum projected area, a *parametric model* is established by measuring width and height of the left ventricle shape. These parameters are used to approximate the contour of the ventricle with an ellipse (see Section 3 and Figure 1.1b). Furthermore this ellipse is used as the base of a rotational ellipsoid. The volume of the ellipsoid can easily be calculated by measuring width A and height B.

There is an ongoing discussion among radiologists regarding which cardiac volume estimation technique is best suited for clinical application, therefore it would be interesting to compare some of these techniques with respect to accuracy or time-consumption and to emphasize their specific advantages and drawbacks.

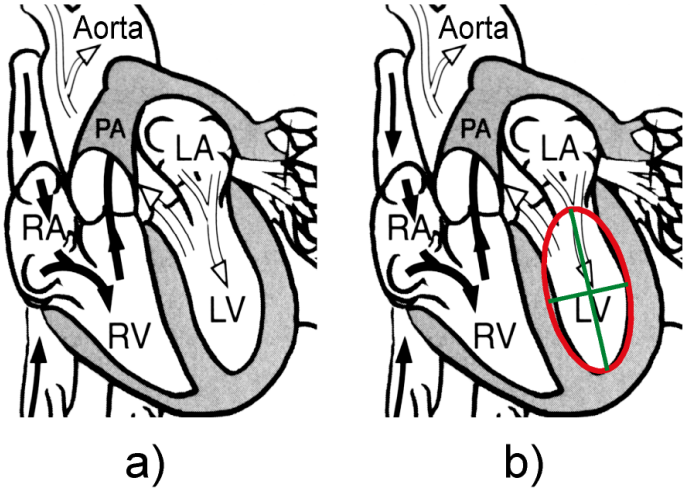

Figure 1.1: The four cardiac chambers. The arrows in a) and b) indicate the direction of blood flow. Blood enters the heart through right atrium (RA) and right ventricle (RV), the pulmonary artery (PA) is the connection to the lungs where the blood is enriched with oxygene. O_2-rich blood enters the left atrium (LA) and after flowing through the left ventricle (LV) it leaves the heart through the aorta. In b) an assumed ellipse contour with width A and height B is additionally shown. (Taken from [52].)

1.2 Project Goals

The primary goal of this work is the implementation of segmentation-based models for left ventricle volume estimation and the comparison with a parametric model by means of a statistical evaluation. Both models operate on a data set of 80 images provided by a computer-tomographic scanner.

The project is carried out in cooperation with Univ. Prof. Dr. Rainer Rienmüller and his team at the *University Hospital Graz, Department of Radiology*. At the department among other devices an *Ultrafast CT scanner* is used for routine medical inspections of hearts as well as for investigating cardiac dysfunction and the risk of cardiac infarction or for optimal surgery timing. These inspections are the basis for calculating left-ventricular function parameters like volume, mass or muscle thickness. From a medical point of view their analysis is required to make a diagnosis. For estimating left-ventricular volume the parametric model (or *Two-Axes Method by Greene*[35], see Section 3) is utilized at the department.

The comparison exclusively concentrates on the volume parameters which are used to derive a *stroke volume* and an *ejection fraction* (the fractional volume of blood ejected). Therefore 31 anonymized patient data sets were provided by Prof. Rienmüller each of them consisting of 80 images, a volume estimation result of the parametric model and manually drawn ventricle contours on the images of interest.

It is necessary to write a tool for conveniently applying the methods and algorithms on the image data sets to compare the parametric model and the segmentation-based model (or *Simpson-Rule Method*[35], see Section 4) . The second goal of this work is to develop a tool including all algorithms necessary to perform the evaluations that integrates into a DICOM environment under Windows NT.

Methods based on the *Simpson-Rule* need images from the complete heart volume to work. In our case the heart volume is projected onto eight images, with each image representing a slice of the body with a certain thickness. This thickness is defined by the CT's resolution in z-direction which, together with the resolution of the imaging plane (x,y), defines the *voxel size* of the CT (*voxel* is the abbreviation for volume element, see Figure 1.2 for an illustration). The left-ventricular border has to be extracted by means of segmentation methods from these images. As soon as we know which pixels are part of the left ventricle the volume can be estimated by counting these pixels on all segmented images and multiplying their number with the corresponding *voxel size*.

In addition limitations due to image acquisition restrictions of the chosen CT scanning mode have to be investigated. The most important restriction comes from the data-set's coarse resolution in z-direction. The CT scan of the heart only consists of eight images, therefore only smaller hearts can be included in the statistical evaluation, because too large hearts would not fit into the scanning volume. Here a segmentation-based model has obvious drawbacks compared to parametric models due to the fact that it needs images from all over the heart while the parametric model only needs a certain number of representative images.

Another restriction caused by limited x-, y- and in our case especially z-resolution is the so-called *partial-volume effect* (see Section 2.1.5 and Figures 1.3,1.2). It is an aliasing problem occurring during the CT image reconstruction stage and directly proportional to the finite size of the discrete volume elements (*voxels*, see Figure 1.2c).

The last limitation which should be mentioned is the influence of the *papillary muscle* (see Figure 1.3), a muscle surrounding parts of the left ventricle. The problem is that the muscle doesn't fit into the approximated ellipsoidal ventricle shape and, although it should be incorporated into the ventricle volume, it can't easily be separated from the ventricle muscle (*myocardium*) because both consist of the same tissue type and therefore both have a similar grey-value on a reconstructed image. In addition radiologists do not agree whether the papillary muscle should be included in the ventricle or not. In this work the papillary muscle will be assumed to be a part of the left ventricle and the impact of this assumption on the different volume estimation techniques will be investigated.

1.3 Concepts and Definitions

This section describes most of the terms, notations and abbreviations used throughout the document, which the reader possibly might not be familiar with:

CT Computer Tomography. A 2D imaging technique based on x-ray absorption.

MR Magnetic Resonance. A 2D and 3D imaging technique based on deflection of atom nuclei due to absorbing electric-magnetic energy.

HU Hounsfield Units. The x-ray attenuation coefficients of different tissues relative to the coefficient of distilled water:

$$\mu_{rel} = \frac{\mu_{tissue} - \mu_{water}}{\mu_{water}} * 1000$$

DICOM Standard for Digital Imaging and Communications in Medicine. It is a protocol for storing and transmitting many different kinds of medical data.

ROI Region of interest. A part of an image an algorithm has to concentrate on.

Cardiac A synonym for the heart.

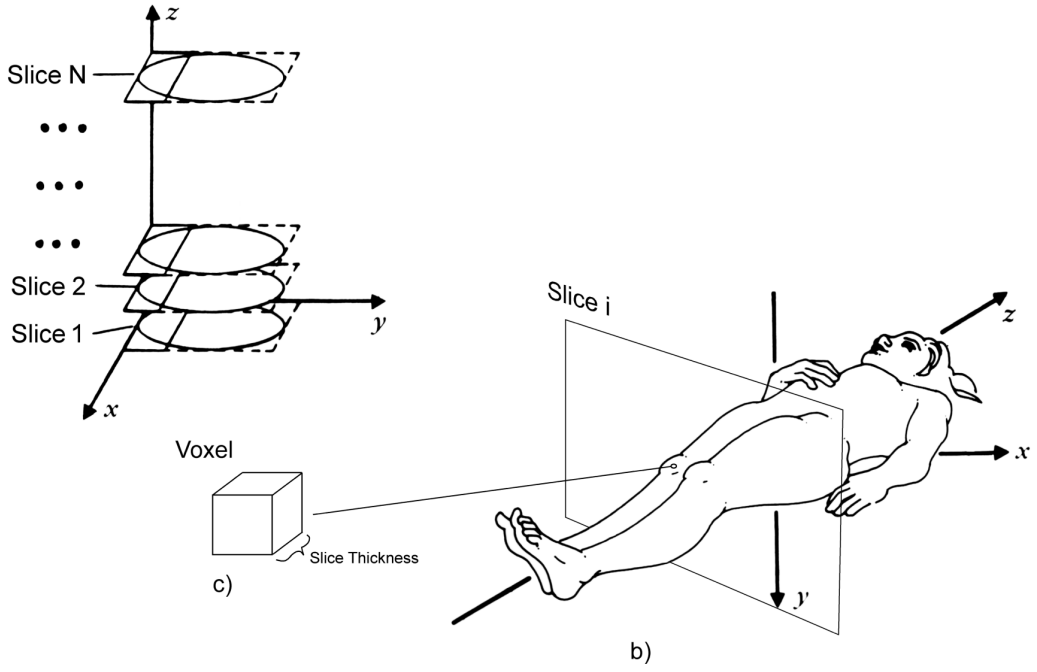

Figure 1.2: CT axis description and voxel illustration. a) shows the commonly used coordinate system of CT scans. b) shows a scanned slice of a patient. c) illustrates an enlarged voxel and its slice thickness. (Partly taken from [20].)

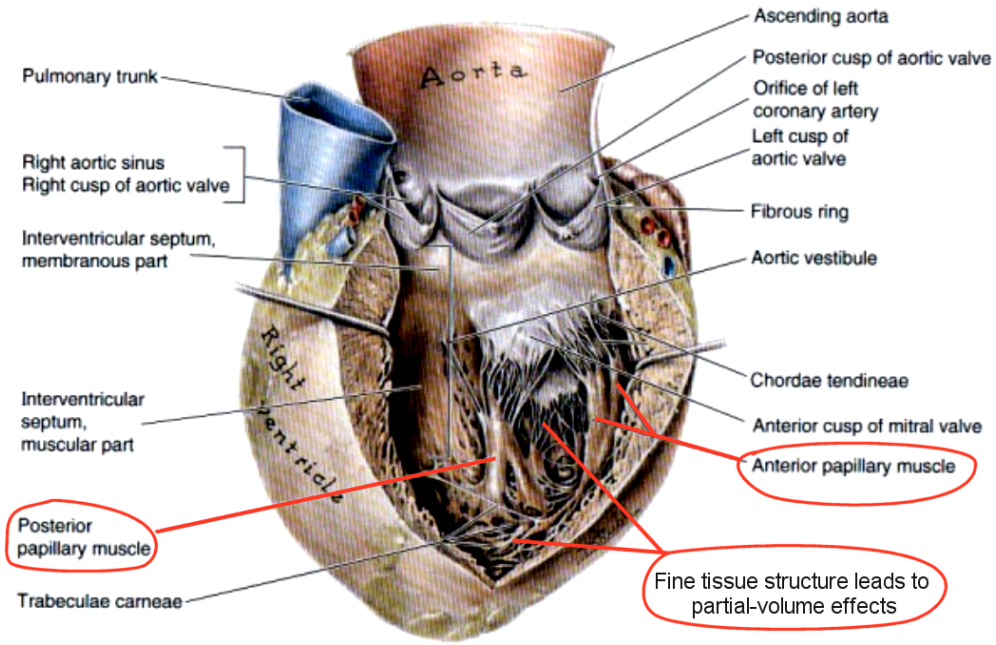

Figure 1.3: Interior of left ventricle. The chamber consists of space for blood, papillary muscles and fine tissue structures which are too small for reconstruction and visualization due to partial-volume effects. (Taken from [33].)

Left/right ventricle The two larger heart chambers. The *right ventricle* is a blood pool where deoxygenated blood from organs and extremities is gathered. The *left ventricle* receives O_2-enriched blood from the lungs and has to pump it back to the organs and extremities.

Left/right atrium The two smaller heart chambers. *Left* and *right atrium* are entry points to left and right ventricle respectively.

Diastole Over time the left ventricle muscle is constantly relaxing and contracting. The duration of relaxation is called *Diastole*.

End-Diastolic This is the point of time when the left ventricle muscle is maximally relaxed. Therefore the ventricle chamber is filled with blood.

Systole Over time the left ventricle muscle is constantly relaxing and contracting. The duration of contraction is called *Systole*.

End-Systolic This is the point of time when the left ventricle muscle is maximally contracted. Therefore the ventricle chamber is nearly empty.

Stroke volume The difference between end-diastolic volume (EDV) and end-systolic volume (ESV):

$$SV = EDV - ESV$$

Ejection fraction The stroke volume divided by the end-diastolic volume:

$$EF = \frac{SV}{EDV} * 100 = \frac{EDV - ESV}{EDV} * 100[\%]$$

Myocardium Is the muscular wall of the heart. It contracts to pump blood out of the heart and then relaxes as the heart refills with returning blood. Its outer surface is called the epicardium and its inner lining is the endocardium.

Papillary muscle Is a muscle which supports the contraction (pumping) of the left ventricle.

RAO projection Right-anterior-oblique projection. Is a view of the heart's front from the right side and slightly oblique.

1.4 Related Work

Many different kinds of work have been performed by researchers concerning the left ventricle since the upcoming of CT and MR. Quantitative evaluations of different volume estimation models have been done by Dulce et al.[15]. They compared magnetic resonance data sets by using a modified Simpson-Rule on a biplane ellipsoid model and the three-dimensional data set of MR images. Rehr et al.[46] did a comparison of Simpson-Rule estimated MR data sets and in vitro (excised) left ventricles. Belohlavek[3] evaluated left-ventricular volumes in 3D-Echocardiography and compared his results with other single- or bi-plane models.

Semelka et al.[50] evaluated the reproducibility of measurements of ventricular dimensions (mass, volume, ejection fraction, and systolic wall stress) obtained with cine magnetic resonance (MR) imaging.

Ranganath [45] performed an automatic contour extraction of left ventricular contours from cardiac Magnetic Resonance Imaging studies. The algorithms were based on active contour models incorporating contour propagation. Another automatization approach for ventricle segmentation in SPECT (Single Photo Emission CT) images was investigated by Newman et al.[42]. Their method involved extraction and fitting of quadric surfaces to approximate ventricle shape.

Goshtasby et al.[21] applied a two-stage algorithm for extraction of the ventricular chambers in flow-enhanced MR images. They approximate location and size of endocardialy surfaces by intensity thresholding and reposition points on the approximated surfaces to nearest local gradient maxima. Afterwards they fit a cylinder into the point set.

A constrained detection of left ventricular boundaries from cine CT images was performed by Taratorin et al.[58] by the use of set-theoretic techniques, a priori knowledge of the heart geometry and temporal dynamics. Weng et al.[62] proposed a learning-based ventricle detection from MR and CT images based on a likelihood measure for region-of-interest detection. Furthermore Philip et al.[43] applied a combination of different techniques on CT images to automatically detect the myocardial contours. These techniques are a boundary matching descriptor, a fuzzy Hough transform and a method from graph searching. Most of these ventricle detection algorithms are applied and tested on short axis projections of the heart chmabers, there were no reports about ventricle segmentation on CT long-axis projections.

1.5 Structure of the Thesis

This section gives an outline of the following chapters by using an image processing block diagram at a rather high abstraction level (Figure 1.4). Chapter 2 will present the *Medical Image Data* used as input for all further considerations and algorithms as well as the acquisition process of the images. Chapter 3 explains the *Parametric Model* in detail. Chapter 4 deals with the algorithms necessary to set up the segmentation-based model. This *Digital Image Processing* stage roughly consists of an *Image Preprocessing* and an *Image Segmentation* step. The topic of Chapter 5 is the implementation of the *Parametric Model*, *Segmentation-based Models* and the *Volume Estimation* technique which are embedded in the herein presented segmentation tool. The estimated volumes are compared in Chapter 6. Chapter 7 states a conclusion and gives an outlook on further work.

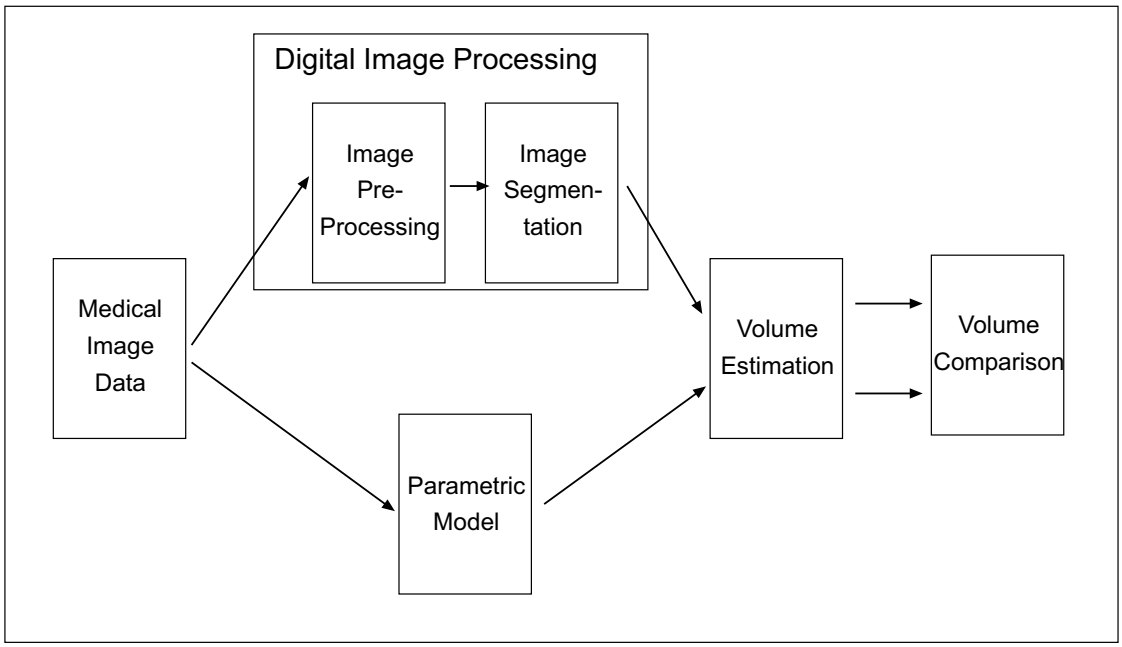

Figure 1.4: Block diagram to illustrate the structure of the following chapters.

Chapter 2

Medical Image Data

This chapter outlines the kind of image data which is the input for all of the later stages as well as the acquisition of the images. It starts with a description of the computer-tomographic reconstruction principle and its application in conventional and Ultrafast CT scanners. Afterwards the DICOM standard and the DICOM file format definition are presented. The last part introduces the specific image data sets provided from the *Department of Radiology, University Hospital Graz.*

2.1 Computer Tomography

Non-invasively acquiring two-dimensional images of the interior of a (human) body by using x-rays goes back to the beginning of this century. But the drawback of losing the third dimension is unacceptable for many applications. Research for improvements lasted until the period between 1968 and 1973. At this time the later inventors of x-ray computer tomography worked on their revolutionary approach to rotate and translate a x-ray source around a body, collect the attenuated x-ray beams with detectors on the other side of the body and reconstruct a three-dimensional volume. For their work on mathematical theory as well as practical application towards the construction of the first operational CT scanner, *Allan M. Cormack* and *Godfrey N. Hounsfield* were awarded with the Nobel Prize in Physiology and Medicine in 1979.

2.1.1 The Tomographic Image Reconstruction Principle

The basic tomographic reconstruction principle is independent of the underlying projection technique. We distinguish three techniques how images can be reconstructed by one-dimensional projections[64]:

Absorption Tomography Intensity differences in a reconstructed image of a body are based on differing absorption capabilities of e.g. tissue, bone or blood. If we have x-rays being attenuated on their way through the body we speak of *x-ray computer tomography*.

Nuclear magnetic resonance imaging is another example, here an external magnetic field interacts with nuclei of atoms possessing a spin which absorbs electric magnetic waves at a radio frequency. This absorption promotes the nuclei to a higher energy level which cannot be maintained for very long. When returning to the former energy state (ground state $\rightarrow$ depends on external magnetic field) the electric magnetic energy is released again, this emission can be measured as a signal which is interpreted as a grey-level image after tomographic reconstruction[1].

Emission Tomography Here the radiation of radio-active substances is measured after being absorbed by the organs and tissues which are to be investigated. Intensity differences are based on the capability to absorb the radioisotopes, specially designed detectors collect the emitted radiation from the organs. Examples for this type of tomography are PET (positron emission tomography) and SPECT (single-photon emission computer tomography).

Runtime Tomography Image reconstruction utilizes differing wave runtimes depending on the kind of used wave and the elasticity properties of the investigated tissue. This principle is used for example in ultrasound-imaging.

Now that we have detected either attenuated or emitted signals, it is possible to reconstruct the scanned three-dimensional object[64]. The principle will be explained using absorption tomography. The tomographic reconstruction process decomposes a three-dimensional object into two-dimensional planes (*slices*). The structure of these slices is determined by sequentially scanning along one-dimensional projection lines (see Figure 2.1). Reconstruction of the attenuation coefficients along each line leads to two-dimensional images, stacking these images results in a three-dimensional volume data set. The reconstruction stage is the trickiest part of the process, because a single detected attenuation coefficient is only an averaged value over all tissue types along its path and therefore we've lost the dimension of depth. But by rotating the signal source around the object while scanning we achieve many attenuation results from different points of view for each point in the volume (see Figure 2.2). The averaged attenuation value of a single path is modeled as a line integral $\int \mu(x,y)d\xi$ with $\mu(x,y)$ being the unknown distribution of the attenuation coefficients with respect to the (x,y) coordinate system and ξ representing the path of the projection. (The (η,ξ) coordinate system is rotated by an angle φ with respect to (x,y), it describes a certain rotated position of the projection line.) Further we can measure attenuated intensities $I(\eta,\varphi)$ from the detectors and we know the intensity I_0 of the energy source. Initial intensity and measured intensity have the following connection:

$$I(\eta,\varphi) = I_0 * e^{-\int \mu(x,y)d\xi}$$

With these ingredients a linear integral equation can be established:

$$\ln(\frac{I_0}{I(\eta,\varphi)}) = \int \mu(x,y)d\xi$$

The solution of this equation can be received by applying an integral transformation consisting of three steps:

- Backprojection of the measured intensity $I(\eta,\varphi)$. In this step the intensity is reprojected and therefore the unknown attenuation coefficients are distributed uniformly along the projection line ξ.
- Additive superposition of all projection lines (corresponding to line integrals). Here we get the position of a point as a result of intersecting reprojected intensities. Nevertheless the resulting attenuation coefficient distribution is not exact but convoluted with a $\frac{1}{r(x,y)}$-function.
- Reconstruction of attenuation coefficients $\mu(x,y)$. To revert the convolution with the $\frac{1}{r(x,y)}$-function, inverse fourier transformation is applied, often in combination with low-pass filtering to reduce noise.

This process is also known as *filtered back-projection*[5] and it leads to an image of the scanned slice.

Figure 2.1: From 3D-object to 1D projections. The tomographic process decomposes the object into 2D slices which are reconstructed from 1D projections like e.g. P,P". (Taken from [64]).

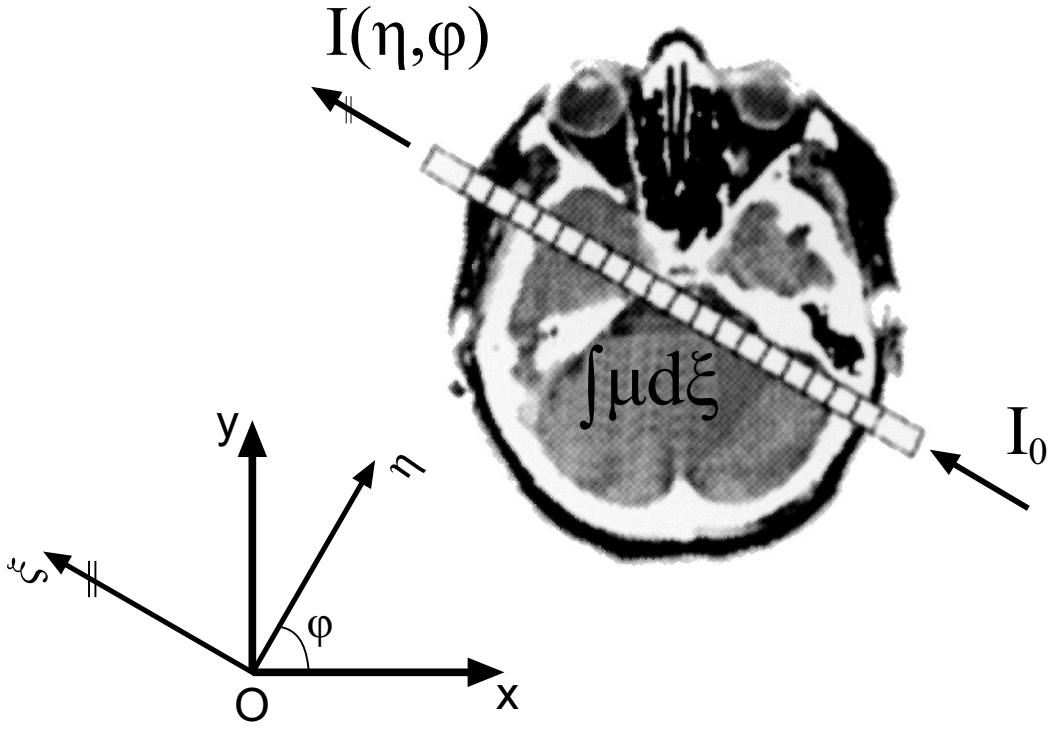

Figure 2.2: Energy attenuation along one projection line. Energy I_0 arrives at the object, an attenuated energy $I(\eta, \varphi)$ reaches the detector. Attenuation along the path ξ is modeled as a line integral $\int \mu(x, y)d\xi$. The (η, ξ) coordinate system is rotated by an angle φ with respect to (x,y). By varying φ and η we get all necessary projection lines and attenuation values. (Taken from [64].)

2.1.2 Conventional CT Scanner

Conventional CT scanners consist of mechanically movable x-ray sources and detectors. The x-ray source is fan-shaped and has a certain slice thickness. The detector consists of a linear array of detector elements. The basic scanning principle is to rotate the source-detector apparatus around the patient and measure the amount of energy received by the detector. From the discrepancy between emitted and received energy an absorption coefficient is calculated (according to the principle described in the previous section) for each rotational position of the apparatus. Different tissues have varying absorption coefficients. This fact is used to create a range of absorption coefficients relative to the coefficient of distilled water called *Hounsfield Units* (HU). Mapping Hounsfield Units to grey-values generates the image of the scanned slice.

However this method has limitations. Mechanical movement of the heavy source-detector apparatus restricts minimum scan time to approximately one second per slice. This scan time is sufficient for many applications but not fast enough for cardiac imaging. The beating heart would lead to unacceptable motion artifacts in the reconstructed image. For this reason an alternative CT scanner configuration called *Ultrafast CT* was developed[57].

2.1.3 Ultrafast CT Scanner

The Ultrafast CT (or Electron Beam CT) at the Department of Radiology is an Imatron Inc. C150 scanner (see Figure 2.3). It consists of fixed x-ray source and detector units. An electron beam is deflected and focused to hit a target ring (a metallic target ring made of tungsten, formerly known as wolfram). At this target ring the energy of the electron beam is converted to x-ray energy which passes through the patient and finally gets absorbed by the detector arrays (see Figure 2.4 and Figure 2.5). There are 2 linear detector arrays in the detector unit. As a consequence it is possible to get 2 adjacent slice images for a single electron beam adjustment. The Ultrafast CT scanner consists of 4 target rings which can be scanned sequentially. Therefore scanning 8 slice levels of a patient can be achieved without moving the patient's table. A scanning sequence to get a pair of slices takes an interval of 50 msec. After completing a scanning sequence a setup-time of 8 msec is needed to readjust the beam for the next sequence. Thus, it is possible to scan 8 slices in a period of 4*50+3*8=224 msec. These scan-intervals are short enough to freeze cardiac motion.
The flexible design with 4 x-ray targets and 2 detector units leads to different operating modes. One of them is called *Movie* or *Cine* mode. Triggered by an external ECG signal all 4 target rings are scanned sequentially. Scanning the first target ring leads to 10 images per detector unit (total 20 images). The acquisition is triggered by the R-wave of the ECG (see Figure 2.6) so that all 20 images are taken within a cardiac cycle. This procedure is repeated with the 3 remaining target rings, triggered by the R-waves of the following 3 heart beats and generating a total of 80 images for 8 different slice positions. They contain a complete cardiac cycle for every slice position consisting of 10 images respectively (see Figure 2.6) [57].

2.1.4 Contrast Medium

To separate different tissues with similar absorption coefficients, i.e. heart muscle and blood, iodinated contrast medium is used. Iodine (300 HU) is capable of absorbing approximately 6 times more x-ray energy than blood or muscle tissue (50 HU). With a certain concentration of iodine contained in blood an easy distinction of iodinated blood and muscle tissue is possible.
Certainly the concentration of iodine decreases over time, but for this application it is assumed that

Figure 2.3: The Ultrafast (or Electron Beam) CT scanner located at the Department for Radiology, University Hospital Graz. It is an Imatron Inc. C150 scanner.

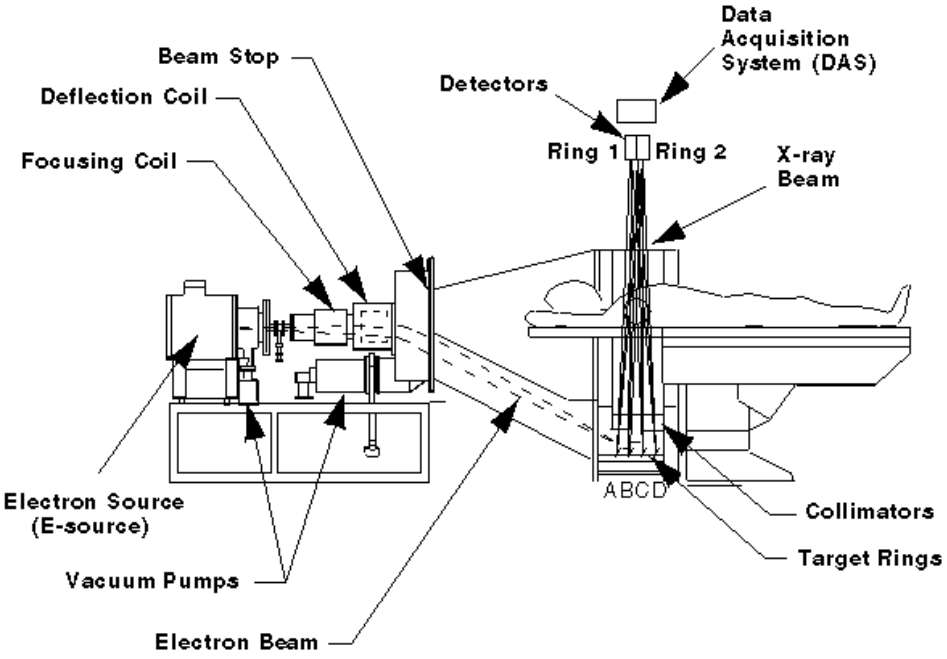

Figure 2.4: Longitudinal view of Ultrafast CT. The electron beam originates at the gun and is accelerated towards the target end. After deflection and focusing it hits one of the four target rings where x-ray radiation is emitted. The attenuated x-ray energy is absorbed by the detector arrays which passes a corresponding signal to the Data Acquisition System. (Taken from [29].)

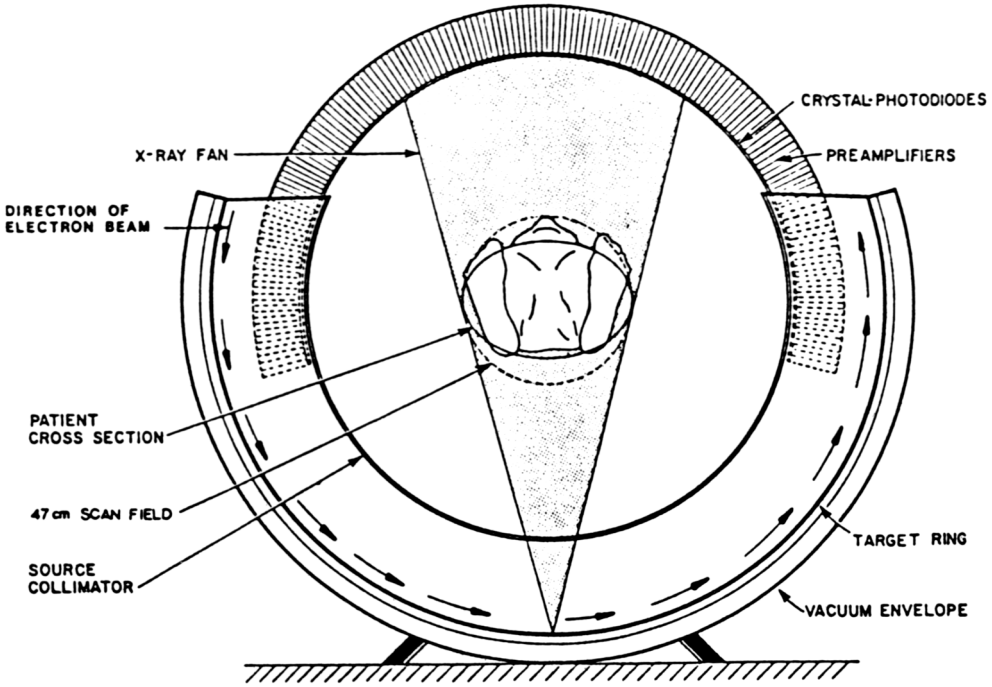

Figure 2.5: Cross-section view of Ultrafast CT. The x-ray fan beam is approximately 30 degree wide. The lower part shows a target ring, the arrows indicate the direction of the moving electron beam. The upper part illustrates the detector array made of crystal photodiodes. (Taken from [57].)

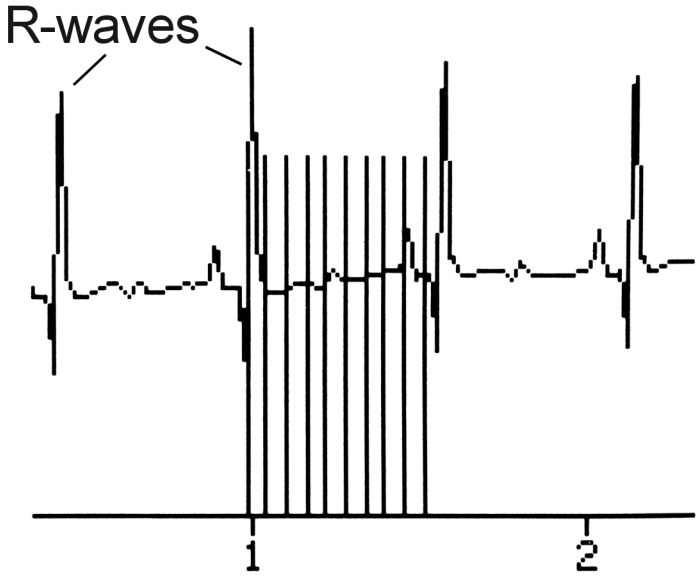

Figure 2.6: ECG trigger signal. A patient's electro-cardiogram showing movie-mode acquisition initiated at the R-wave (the high peak at position 1). Each vertical line represents a pass of the electron beam along a target ring. (Taken from [57].)

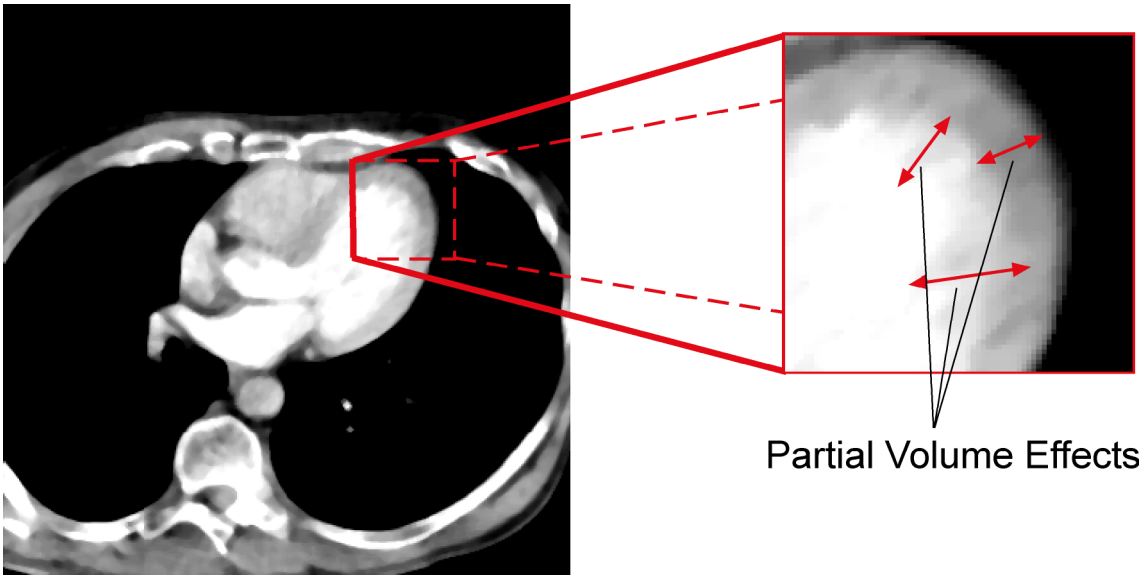

Figure 2.7: Partial volume illustration. The image shows a CT slice of the left ventricle near end-diastolic state. Bright white indicates blood while light-grey represents the heart muscle. Although the partial volume effects occur everywhere in a reconstructed image, they can especially be seen on the enlarged part. The transition between ventricle chamber and muscle is blurred.

the concentration remains at a constant level during the acquisition time. Due to the short total scan time (4 heart beats) the error is negligible.

2.1.5 The Partial Volume Effect

The most important restriction coming from the tomographic image reconstruction technique is the *partial volume effect*. Due to the fact that we have a discretization to finite volumes during the volume scanning stage it is not possible to determine details in the volume which are smaller than this discrete volume. Therefore we have an *aliasing* effect inhibited in the scanning procedure. Shannon's sampling theorem says that it is necessary to use a sampling frequency (as defined by the voxel size) twice as large as the spatial frequency of the smallest image detail we want to reconstruct. In other words all details with a spatial frequency smaller than half of the sampling frequency won't be reconstructed correctly. All of these details located inside the smallest possible volume will be averaged to form a grey-value. As an example many of the small veins situated in the left ventricle are not reconstructed exactly.

In our images we have a scan volume of 0.8mm x 0.8mm in (x,y)- and 8mm in z-direction. Therefore the averaging along the z-direction is much stronger than in the other directions (see Figure 2.7). As we will see later the partial volume effect has quite an impact on segmentation-based models, but not on the parametric model.

Group Nr (hex)	Element Nr (hex)	Name	Entry
0002	0010	TransferSyntax	LittleEndian (defines the byte-order for the rest of the file)

Table 2.1: Example DICOM file entry.

2.2 DICOM

With the upcoming of all kinds of imaging devices, computers and databases used for medical diagnostics it has become necessary to standardize communication processes between these devices. So in 1983 *ACR* (American College of Radiology) and *NEMA* (National Electrical Manufacturers Association) formed a joint committee to develop a standard for transmission and storage of medical information. ACR and NEMA published the first version of their standard in 1985. In 1993 DICOM 3.0 (Standard for Digital Imaging and Communications in Medicine) was presented, which is still the current version. It describes how to utilize a TCP/IP network layer as base for all transmission tasks and includes specifications of various possible information objects, like images from different kinds of imaging devices or therapeutic and diagnostic information. The goals of DICOM are to achieve compatibility and to improve workflow efficiency between imaging systems and other information systems in healthcare environments worldwide. Furthermore the standard supports convenient development and expansion of picture archiving and communication systems (PACS) and interfacing with medical information systems. For more information on DICOM please refer to the homepage[41].

2.2.1 DICOM Image File-Format

For this work especially the image file-format specified in the DICOM standard was of interest, because the input images received from the *Department of Radiology* were stored in this format. As a consequence it is necessary to explain the concept and some of the property fields.

A DICOM image file includes many properties of a medical image beside raw pixel data. In fact a DICOM file consists of three sections which can be distinguished. First of all there is a short *meta-information header* containing meta-information about the rest of the file. Afterwards a section containing image, acquisition and medical information, the *DICOM data set*, is included. Finally there is a section with the raw image data, either compressed or uncompressed. The first two sections are stored in an ASCII manner, the raw image data is stored binary. ASCII data has a common structure, each entry of the sections basically consists of a unique tag, namely the combination of a group number and an element number, and an entry. As an example the most important entry of the meta-information header the *Transfer Syntax* which describes the byte-order of the rest of the file looks like in Table 2.1.

An important subset of properties from the second section of the file is needed to understand the characteristics of the volumetric data set, therefore they are listed in Table 2.2.
There are some additional properties which are not explicitly contained in the data set, but nevertheless they are important:

Slice Gap: There is no slice gap between two adjacent slices generated by the same target ring adjustment. But between two adjacent slice pairs a gap of 4 mm exists.

Patient Orientation: The patient is lying on the table so that the heart is sliced in long-axis configuration (see Figure 2.8c).

Tag	Name	Entry	Description
(0028,0011)	image columns	512	Image is stored with 512 columns but only the central 360 columns contain image information (x-direction).
(0028,0010)	image rows	512	Image is stored with 512 rows but only the central 360 rows contain image information (y-direction).
(0018,0050)	slice thickness	8 mm	A reconstructed image is a projection of an 8 mm thick body region (z-direction).
(0008,1030)	study description	Movie study, ECG	Indicates that the Ultrafast CT is operating in *movie mode* generating a total of 80 ECG-triggered images.
(0028,0030)	pixel spacing	0.833mm x 0.833mm	The scanner's geometric resolution in x- and y-direction is 0.833mm respectively.
(0028,0101)	bits stored	12	The radiometric resolution of the grey-value images is 12 bit.
(0028,0100)	bits allocated	16	Each 12 bit grey-value of the image is stored in 2 bytes (16 bit).
(0028,0102)	high bit	11	Indicates that bits 0 to 11 are the bits storing the grey-value information. (Bits 12 to 15 could contain additional data.)
(0028,1052)	rescale intercept	-1024	The offset which has to be added to get the associated *Hounsfield Unit* for a given grey-value.

Table 2.2: Subset of DICOM properties.

Voxel Size: By considering one certain moment of the heartbeat we get 8 images to define a volumetric data set of dimension 360x360x8 voxels. Taking into consideration the 4 mm gap between the 4 pairs of slices we get a total scanned volume of approximately 300 x 300 x 80 mm^3.

Radiometric Resolution: The images have a grey-scale resolution of 12 bit, they are stored using values between 0 and 4095. These values are derived from the CT's Hounsfield Units by adding the negative rescale intercept 1024.

2.3 Used Data Sets (Long-Axis Movie-Mode Images)

As ground truth for the evaluation *Prof. Rienmüller* from the *Department of Radiology* provided anonymized data sets of many patients and the associated results of the model parameters. These data sets show hearts which were sliced in long-axis configuration. It is also quite common in left ventricle volumetry to use images taken in short-axis configuration. The two configurations are illustrated and compared in Figure 2.8. With short-axis images it is easier to find the border of the left ventricle and the myocardium, because the edges are quite sharp especially in the middle of the left ventricle. Furthermore they usually don't show merged chambers. The problem is that short-axis images are quite inaccurate near the base and the top of the left ventricle due to partial volume effects.

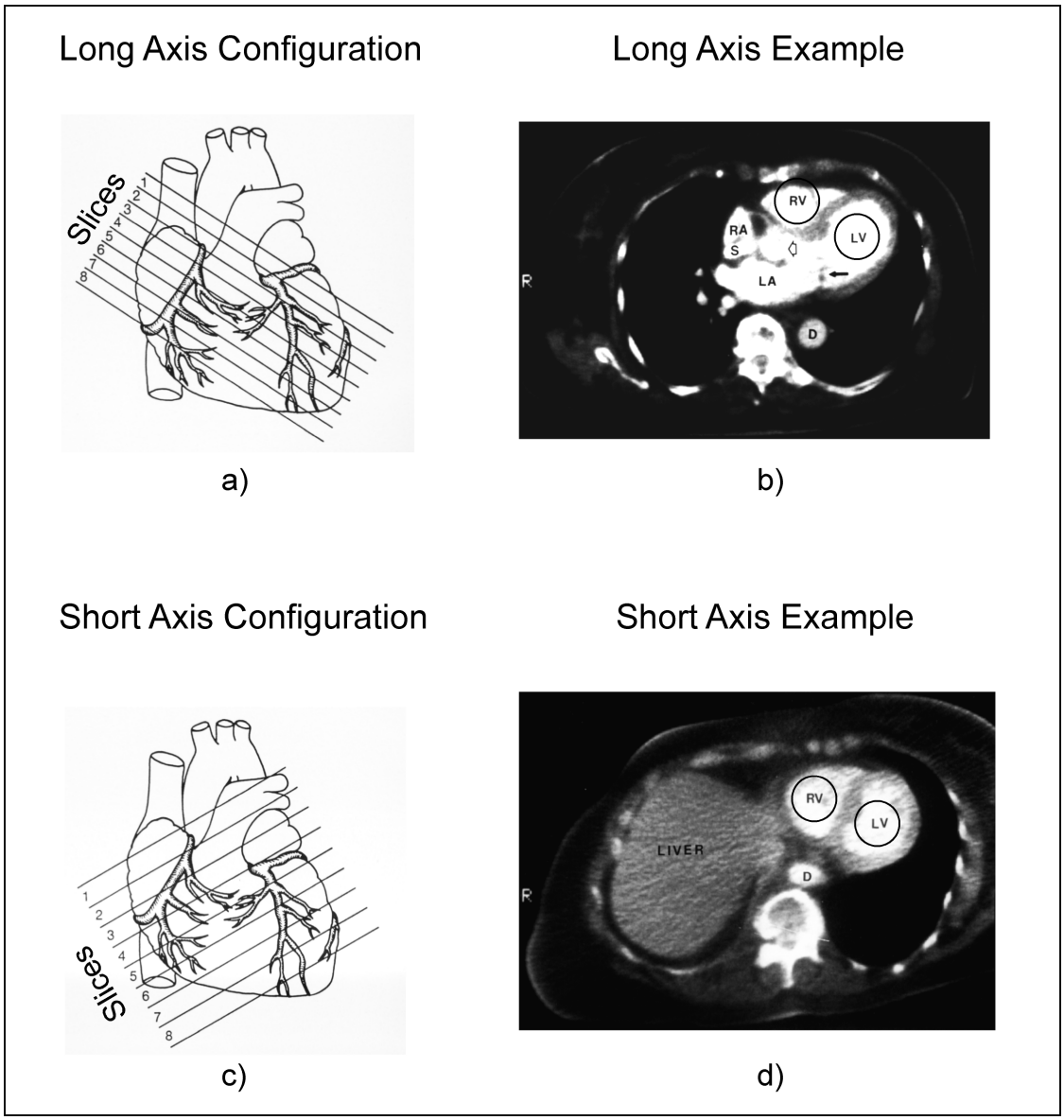

Figure 2.8: CT long-axis (LA) and short-axis (SA) configuration. a) shows long-axis images slicing the heart from the anterior to the inferior surface. b) illustrates a typical LA slice showing all four heart chambers in one image. c) shows short-axis images which are taken from the heart apex to the base. d) illustrates the typical SA slice with rather sharp ventricle walls. In b) and d) circles indicate the position of left and right ventricle. (Taken from [57].)

On the other side long-axis images have more blurred ventricle chamber edges and it is quite often the case that they include merged heart chambers. Nevertheless there is no region which is extremely prone to partial-volume effects like e.g. the ventricle base on short-axis images. Deciding which configuration is more accurate is quite difficult (and not part of this work) because both have advantages and severe drawbacks.

The data set is taken in Ultrafast CT's so-called *movie-mode* (explained in Section 2.1.3), as a consequence each consists of 80 images. Every position in the scan volume (there are a total of eight slice positions) is temporally sampled ten times, therefore we achieve a movie of the beating heart. For the subsequent processing steps it is not necessary to closely investigate all of these 80 images. The parametric model only needs two special images being found, one at the end of diastole and one at the end of systole, to apply fitting ellipse parameters to them. The segmentation-based model works with 16 images, eight images for the volume at the end of diastole and another eight for the volume at the end of systole. The two images from the parametric model have to be a subset of these 16 images.

Figure 2.9 shows an example data-set consisting of 80 images. The first column shows the volume data set at the end of diastole with the left ventricle projected with maximum area. The sixth column shows the volume data set at the end of systole. An enlarged version of these two specific volume data sets is shown in Figure 2.10. The green contours indicate the ventricle borders as specified by a radiologist, these contours were drawn into the image with exactly the same grey-level scaling as shown in Figure 2.10. Further it can be seen that not all of the images hold information about the left ventricle. (The row denotation of both images is identical!) The images at the top and at the bottom of the volume do not contain projected ventricle areas, so they can be neglected.

2.3.1 Anatomical Aspects

For a better understanding of the input images some aspects of cardiac anatomy will be explained next. The heart consists of four chambers, left and right atrium as well as left and right ventricle. Figure 2.11 shows them and also indicates the direction of blood flowing through the body with arrows. The atria are located upstream to their respective ventricles and serve as conduits to supply the main pumping chambers of the heart (the ventricles) with blood. Left and right ventricle have to deliver blood to the body (systemic circulation) and to the lungs (pulmonary circulation). Normally blood from organs, head and extremities is collected in the right atrium and then pumped into the right ventricle. The right ventricle pumps the blood through the pulmonary artery to the lungs where it is enriched with O_2. The O_2-rich blood from the lung travels through a vein to the left atrium and further to the left ventricle. In the left ventricle the necessary pressure to pump the oxygenated blood back to organs, head and extremities is generated by a contraction of the myocardium (ventricle muscle)[52].

Figure 2.8b shows a long-axis transverse section of the heart as seen on CT images. There RV denotes the right ventricle, LV the left ventricle, RA the right atrium and LA the left atrium. D is the descending aorta, a part of the largest body artery, which connects the heart with the lower body parts. S is the superior vena cava, a major vessel (vein) in the chest that drains blood from the upper part of the body into the heart.

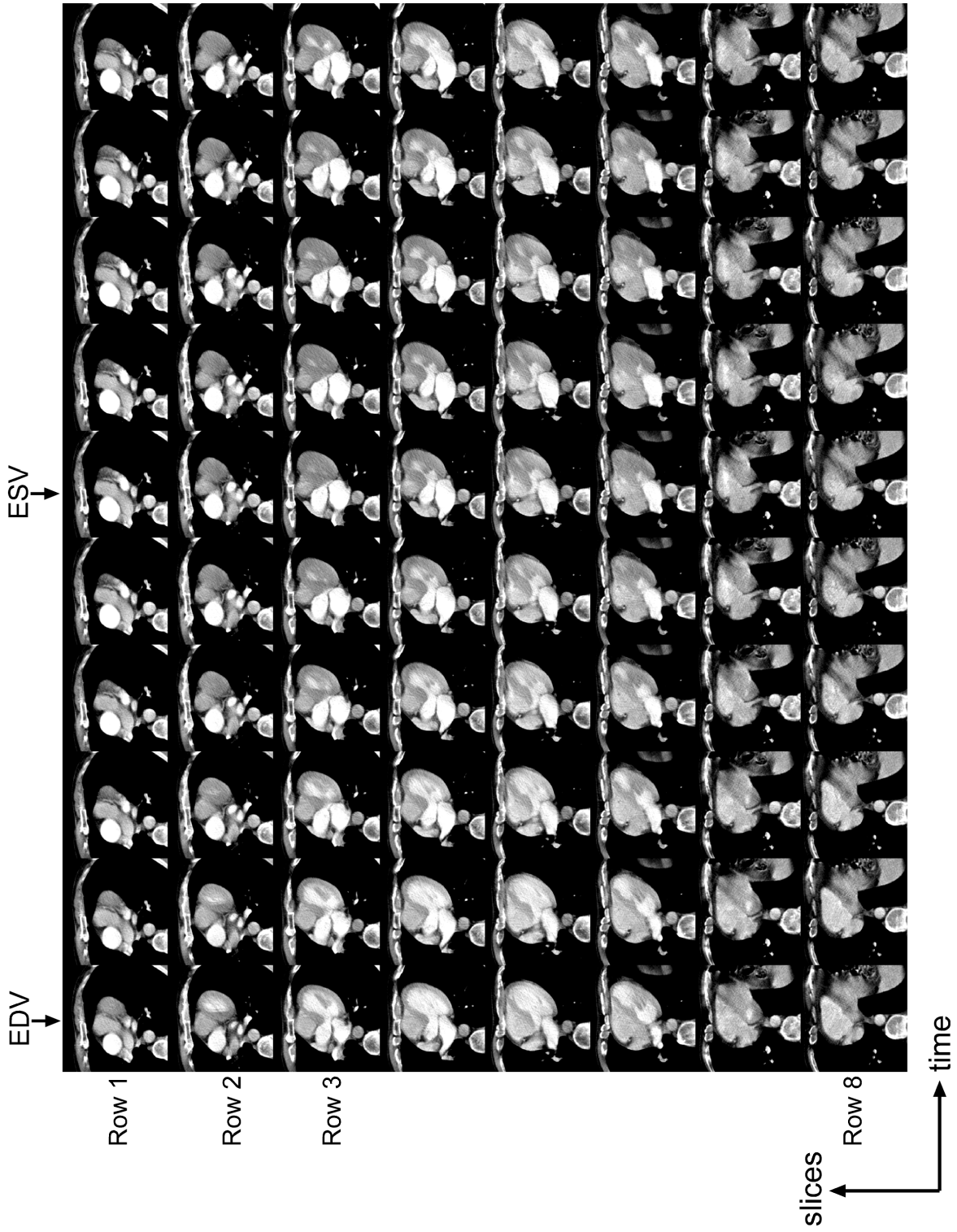

Figure 2.9: One of the data sets containing 80 images. Each row represents a spatial position, the first row shows the topmost slice while the last row shows the bottom slice with respect to the body. A row consists of 10 columns, representing different points of time. So in a row from left to right we have a single heart beat. Column 1 (EDV) shows the volume data set at end-diastole and column 6 (ESV) shows the volume data set at end-systole.

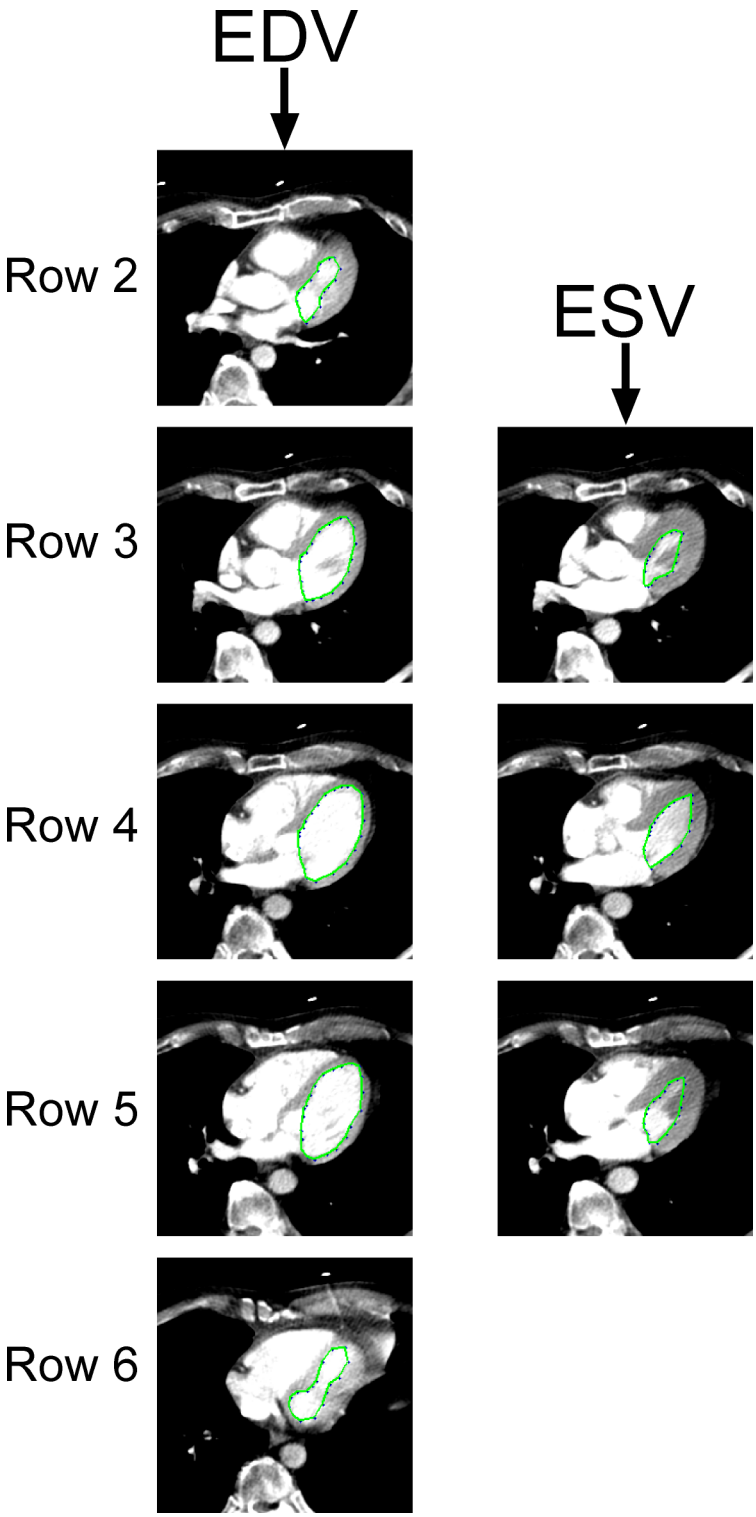

Figure 2.10: Data set 10 with drawn-in contours. The left image series shows a part of the volume at the end of diastole. The right one shows a part of the volume at the end of systole. From both sets only those images are presented, which have an influence on the ventricle volume. The green contours indicate the left ventricle contours as specified by a radiologist.

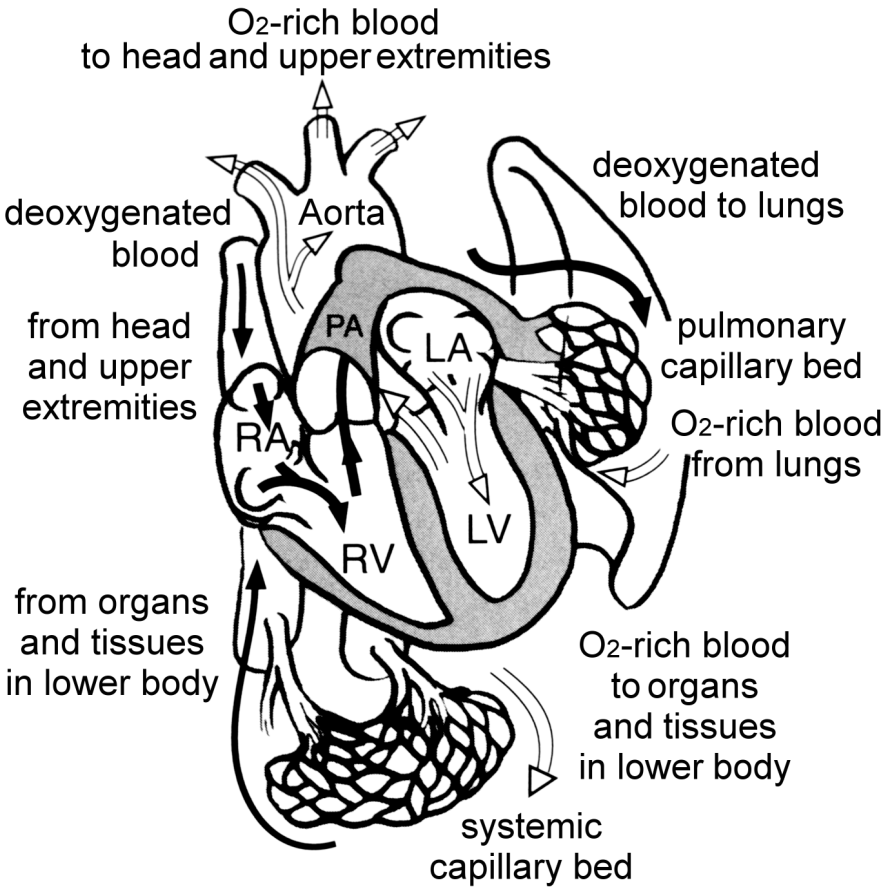

Figure 2.11: Schema of the direction of blood flow through the cardiac chambers, the lungs and the body. Abbreviations: LA: left atrium; LV: left ventricle; PA: pulmonary artery; RA: right atrium; RV: right ventricle. (Taken from [52].)

Chapter 3

Estimating Volumes by means of Parametric Models

3.1 Introduction

The volume of a healthy heart is approximately 90-120 ml at the end of diastole and 20-40 ml at the end of systole. To estimate these volumes it is necessary to apply invasive or non-invasive medical investigation techniques. A common example for an invasive method is a cardiac catheterisation investigation while non-invasive methods (CT, MR, ...) were already mentioned in former chapters. The cardiac catheter or *angiogram* is a special x-ray investigation technique in which angiograms are taken of the coronary arteries and the left ventricle. Therefore catheter tubes are placed in the artery of the right arm for example and then manoeuvred to the heart where radiographic contrast or *dye* is injected. This radiographic contrast is used to get an x-ray image of the left ventricle in the so-called 30 degree right-anterior-oblique (RAO) projection (see Figure 3.1).

If images are regarded at a certain projection and parameters are measured from these images we speak of a *parametric model*.

3.2 Two-Axes Method by Greene used in Cardiac Catheter Ventriculography

This method is based on the *Three-Axes Method by Arvidsson*[35] which needs two projection planes (two different images) and three axes to estimate a left-ventricular volume. Greene et al. modified it in 1967 to use it on single-plane projections (RAO projections) taken from cardiac catheterisation[35]. Prerequisite for all axes-methods is the assumption that the left ventricle can be approximated by a rotational ellipsoid or a spheroid. In the three-axes method from the general formula for a spheroid

$$V = \frac{\pi}{6} * M_1 * M_2 * L$$

with L being a measured long axis and M_1,M_2 being two measured short axes from the two projection planes respectively, a specialized formula is derived by taking into account the two used projection planes and their geometries.

Due to the observation that the two projected short axes are quite similar, Greene et al. investigated how Arvidsson's method could be modified to use only a single projection plane. They showed that it is acceptable to calculate the left-ventricular chamber volume with the formula:

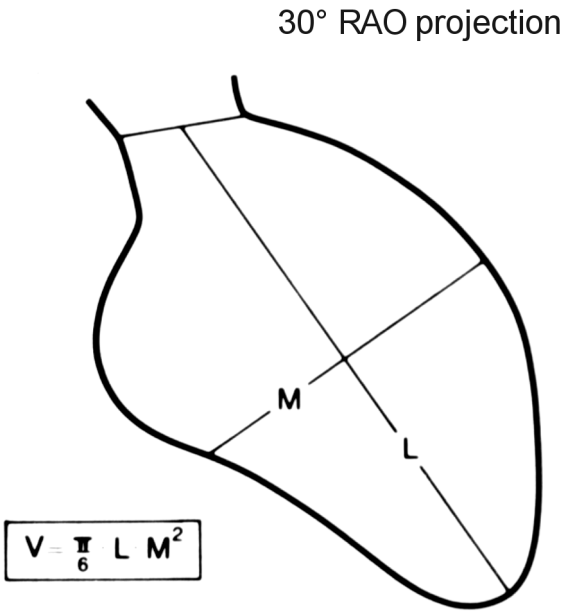

Figure 3.1: Left ventricle right-anterior-oblique (RAO) projection. This projection of the left ventricle is received in cardiac catheterisation investigations. M is the short axis, L the long axis and V the calculated volume.

$$V = \frac{\pi}{6} * M^2 * L \tag{3.1}$$

with M being the short axis and L being the long axis (see Figure 3.1). Correlation with other volume estimation techniques and post-mortal measuring techniques appeared to be acceptably good.

3.3 Two-Axes Method by Greene used in Computer Tomography

Figure 3.2 shows a schematic of a CT image slicing the heart near the plane of the heart valve. This configuration can be achieved using Ultrafast CT's long-axis projection mode. Comparing Figures 3.1 and 3.2 it can be seen that the left ventricular contour in both projections is similar. So it can be expected that the two-axes method by Greene can also be applied to CT images provided that the right images for measuring ellipse parameters can be extracted from a CT image data set. In order to evaluate if there really is such a strong linear correlation between the two methods Rienmüller at al.[47] compared them. They report a strong correlation (correlation coefficient r=0,96 with 47 data sets) provided that the assumption of estimating the left ventricle with a rotational ellipsoid or a spheroid is valid.

As a result this method is applied today in routine diagnosis by Prof. Rienmüller and his team at the *Department of Radiology, University Hospital Graz*. Therefore the procedure will be explained in more detail. First of all it is necessary to find the two representative images on which the two-axes method is to be applied, given a data set of in our case 80 images. These two images are at end-diastole, i.e. when the left-ventricular muscle (myocardium) is maximally relaxed and the ventricle chamber is filled with blood, and at end-systole when the myocardium is maximally contracted. To

find the right end-diastolic image we have to look for a projection of the left ventricle with maximized area and circumference. By determining this image we have already defined the slice position in the volume where we have to look for the end-systolic image, because it has to occur in the same slicing plane. From the 10 possible images the specific image where the left ventricle has an area/circumference minimum is the end-systolic image. It normally occurs with a temporal distance of four to six images from the end-diastolic image. These steps and the following ones are currently performed by a trained radiologist.

Now ellipse parameters have to be extracted from the two identified images (see Figure 3.2). Therefore a horizontal line (H) parallel to the patient's back and a vertical line (V) perpendicular to H are drawn into every image being investigated. Furthermore a line (L) can be established which forms an angle of 60 degree with the horizontal line H and passes through the ventricle apex (point A). This line represents the long axis of the left ventricle. In some cases when it is visually obvious that L does not match the left ventricle long axis, a slight correction of the line's angle is made. To get an ellipse width we need two points on L defining a range. Point A is one of them, the second one, point B, is established by intersecting the connection line of the atrio-ventricular pits (G) with L. Perpendicular to L lies the short axis (M). The left-ventricular borders intersected with M yield the points C and D. Finally A,B,C and D define an ellipse, by measuring the two lengths $\overline{AB}$ (between A and B) and $\overline{CD}$ (between C and D) respectively the two-axes method can be applied by calculating

$$V = \frac{\pi}{6} * \overline{AB} * \overline{CD}^2$$

Note that depending on the application (measuring whole ventricle volume, ventricle chamber volume or myocardium volume) A,B,C and D are positioned slightly differently.

Figure 3.3 shows an example for measured ellipse parameters at end-diastole and end-systole respectively.

3.4 Restrictions

As important as having a technique for volume estimation is the knowledge about restrictions of the specific technique. The parametric model explained in this section has a disadvantage concerning inclusion of the papillary muscle, because it includes the papillary muscle volume into its estimation in any case. Radiologists do not agree on the fact if it should be included or not. Therefore it would be much better if an estimation model supports choosing between inclusion and non-inclusion instead of preventing one aspect. Another drawback of the model is operating expense. It takes quite long to select two images from a data set of 80 images and to manually measure the necessary ellipse parameters. This has to be done by trained radiologists who in addition tend to get tired after some time of accurate working. This is another possible error source, but it isn't specific to this model or radiologists, because every work that has to be accurate leads to mental fatigue.

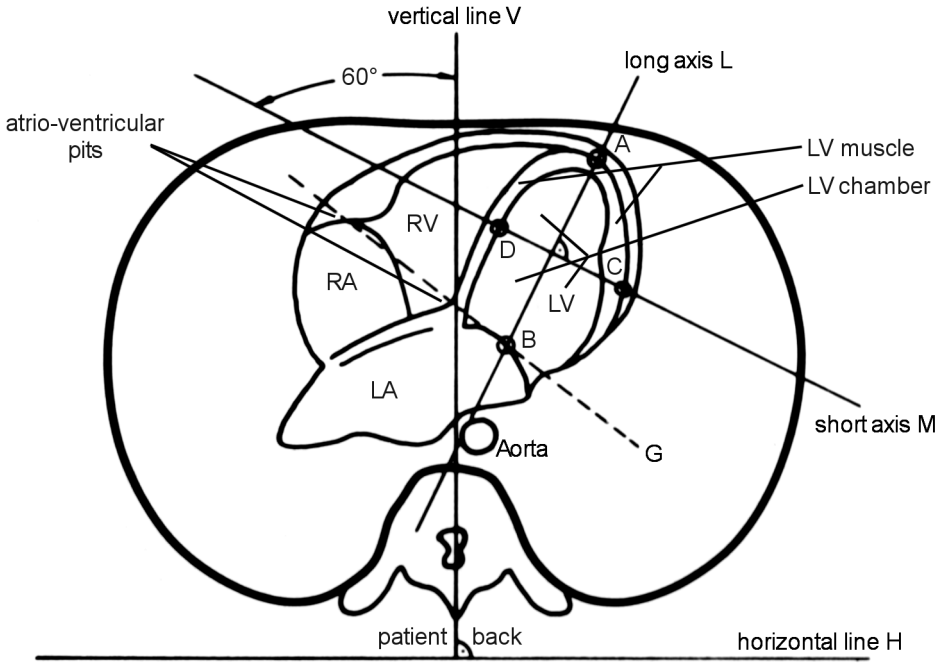

Figure 3.2: CT transverse section of the heart. The points A,B,C and D define an estimated ellipse. Explanation: RA - right atrium; RV - right ventricle; LA - left atrium; LV - left ventricle; L - ellipse long axis; M - ellipse short axis; A,B,C,D - ellipse points. (Taken from [57].)

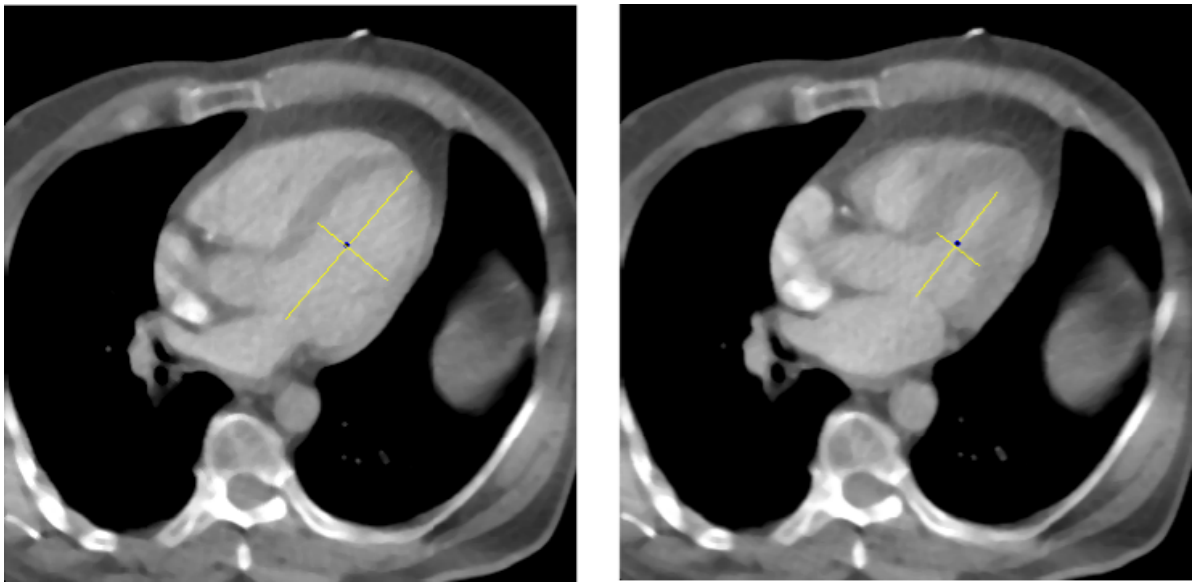

Figure 3.3: Left ventricle volume estimation using the parametric model. The left image shows the selected image at end-diastole while the right one shows end-systole. In both images measured ellipse width and height are drawn in.

Chapter 4

Estimating Volumes by means of Digital Image Processing

This chapter presents all digital image processing techniques which are necessary to estimate the left-ventricular volume as far as they have been used in the implemented segmentation tool. The digital image processing block diagram in Figure 4.1 shows an enlarged part of Figure 1.4.

Initially we have an image data set with 80 images of size 512x512 pixels. The first step necessary is to crop the center part (360x360 pixels) of these images because the outer part contains no useful information, it is reserved for overlaying e.g. patient data in the images. Afterwards the data set has to be scaled to an appropriate grey-level range and filtered to suppress noise. Now it is possible to apply segmentation algorithms on the images.

A data set of 80 images is quite large. Considering the fact that only 16 images (eight images at end-diastole and eight at end-systole) are needed for all evaluations, it would obviously be very helpful to automatically find these 16 images. The first step to reach this goal is to locate a region of interest around the left ventricle. Afterwards the image at end-diastole can be found by looking for the largest ellipse in the regions of interest of all images. This ellipse can also be used to provide an initial ellipse contour to a full-automatic segmentation algorithm. The image with the largest ellipse defines the volume at end-diastole, the volume at end-systole can be found by looking for the image with the smallest ventricle area.

4.1 Image Preprocessing Techniques

4.1.1 The Window/Level Concept for Scaling

Basically we have different families of operations which can be applied to images. One of them are grey-scale transformations[55], they do not depend on pixel positions and only operate on a pixel's brightness. A transformation T of a pixel brightness p from scale [p_0,p_k] into brightness q $\in$ [q_0,q_k] is given by:

$$q = T(p)$$

Some common grey-scale transformations like inversion or thresholding are shown in Figure 4.2a. Figure 4.2b demonstrates a window/level transformation as a special case of a piecewise linear scaling operation.

The medical images have a grey-level resolution of 12 bit, i.e. there are grey-values between 0 and 4095. These values correspond to certain x-ray absorption coefficients of different tissues,

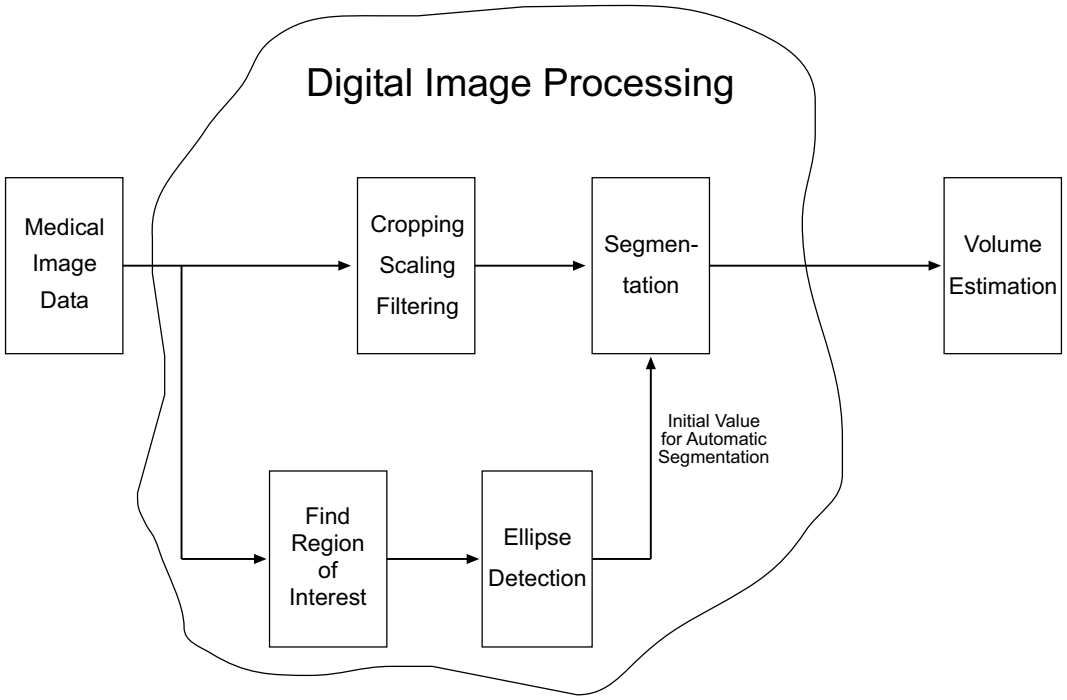

Figure 4.1: Digital image processing overview. This block diagram shows the necessary steps to extract an estimated volume from medical images using digital image processing techniques.

bones, or liquids. The Hounsfield Unit range between -200 HU and 300 HU (this maps to a grey-scale range between 824 and 1324) approximately contains all interesting organic matters for the later image processing stages. So it is possible to filter unwanted organic substances by simply dropping corresponding grey-values. To get a better contrast in the images it is now possible to scale the remaining grey-values to the full range between 0 and 4095. This whole process is called *window/level transformation*.

Supposing pixel brightness to be a function p(x,y), the following equation describes a piecewise linear scaling operation:

$$q(x,y) = \begin{cases} q_1 & \forall_{p(x,y)} : p(x,y) \leq p_1 \\ \frac{q_2 - q_1}{p_2 - p_1}(p(x,y) - p_1) + q_1 & \forall_{p(x,y)} : p_1 \leq p(x,y) \leq p_2 \\ q_2 & \forall_{p(x,y)} : p(x,y) \geq p_2 \end{cases}$$

In the special case of a window/level transformation [p_1,p_2] is the range that should be enhanced and [q_1,q_2] is equal to the full range [q_0,q_k]. The obvious purpose of these operations is a better contrast of the grey-values representing the heart tissues as well as the iodinated blood while suppressing tissues like lungs or bones.

4.1.2 Image Filtering for Noise Suppression

Due to the image acquisition principle the data sets do contain a certain level of noise. To smooth the noise filter masks have to be convoluted with the image. Possible linear filter masks are averaging filters or Gaussian filters. Another approach to remove noise is a non-linear median filter, which

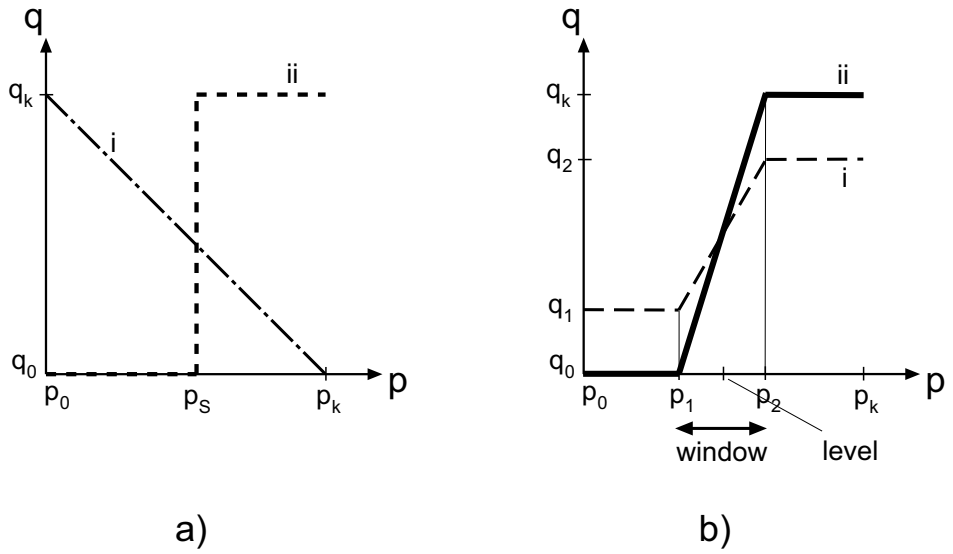

Figure 4.2: Grey-scale transformations. a) shows some common transformations like inversion (i) or thresholding (ii) with threshold p_s. b) shows a piecewise linear scaling operation (i) where the grey-scale range [p_1,p_2] is mapped to a new range [q_1,q_2]. Values outside [p_1,p_2] are suppressed while values in this range are contrast-enhanced. Special case (ii) is a window/level transformation where [q_1,q_2] equals [q_0,q_k].

investigates a local *k*x*k*-neighbourhood of a given pixel and replaces it with the median value of the k^2 sorted values in this neighbourhood[55].

4.1.3 Automatic Identification of a Region of Interest

The most important observation on the way to find a region of interest (ROI), i.e. the region around the left ventricle, is the fact that the left ventricle is responsible for pumping blood through the body. So we can derive a ROI by utilizing the temporal information that we get with the image data set. Temporally adjacent images show distinct movements of the heart while regions like the bones are nearly motionless during the short scanning period.

We have 8 slice locations with a temporal resolution of 10 images respectively. First of all it is necessary to apply some preprocessing steps. All images are filtered to reduce the noise level and scaled to a grey-value range where the iodinated blood is accentuated. This range is equal for all images, it corresponds to a certain range of Hounsfield Units. Afterwards by taking the difference between pixels in temporally adjacent images, we are able to establish result images with pixel values distinct from zero where pixel differences exceed a certain noise threshold. These result images are added to form a final image representing all pixel regions where motion occurs in the whole data set. It contains a pixel-based ROI, fitting a coordinate-axis parallel rectangle into the morphologically enhanced pixel-set leads to a rectangular ROI. Figure 4.3 shows a block diagram of Algorithm 1.

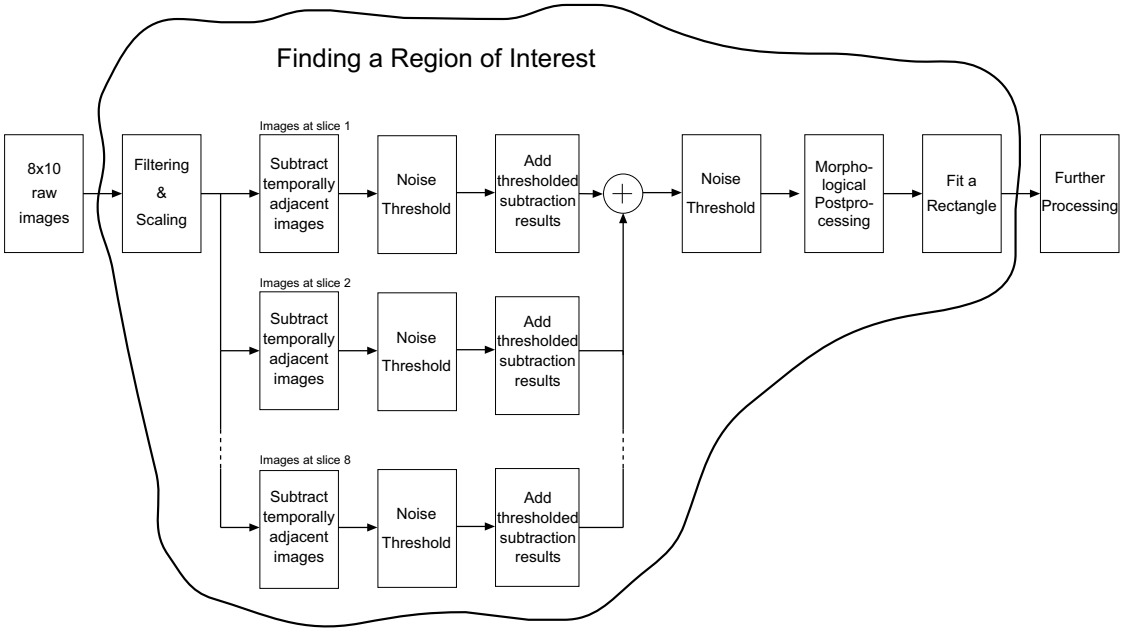

Figure 4.3: Block diagram Region of Interest. Illustration of the necessary steps to find a region of interest given a data set of 80 images.

Algorithm 1 Find a region of interest given an image data set with 80 images.

findRegionOfInterest

1: filter all 80 images with a 5x5 Gaussian filter mask to suppress noise
2: scale all 80 images so that the iodinated blood is easily distinguishable from the ventricle muscle
3: initialize a result image R with 0
4: **for all** 8 slice locations **do**
5: **for all** 10 temporally adjacent images `img` **do**
6: subtract `img`(i-1) from `img`(i) forming image T
7: set pixel values in T which are below a noise threshold to zero
8: add image T to result image R
9: **end for**
10: **end for**
11: set pixel values in R which are below a noise threshold to zero
12: apply morphological operation 'Open' on R to suppress outliers
13: find a coordinate-axis-parallel rectangle enclosing those pixels of R which are larger than zero

4.1.4 Hough Transform for Ellipse Detection

There are two motivations concerning this work to find ellipses in a preprocessing step. First of all finding the largest ellipse in the region of interest around the left ventricle over all images of the medical image data set is likely to define the image at end-diastole and therefore also the volume at end-diastole. This observation which follows from the fact that the left ventricle on the end-diastolic image always has a more or less elliptic shape helps in reducing the complexity of the volume estimation problem from 80 to 16 images.

Furthermore an automatically extracted ellipse would be an important starting condition for a contour-driven full-automatic segmentation algorithm like Active Contours (see Section 4.3).

Related Work

Finding ellipses in digital images is an intensively studied area due to the fact that this problem occurs quite often in different sub-disciplines of digital image processing. Many common objects have exact or approximated elliptic forms when projected onto an image. The by far most commonly used technique for detecting ellipses is the *Hough Transform (HT)*. The basic concept of the HT is to define a mapping between an image space and a discrete parameter space. Transformation rules define accumulation points in the parameter space. By finding local maxima in the parameter space the analytic description of the geometric primitives can be derived[55]. The basic HT concept was introduced by Hough in 1962[27].

A problem of the basic HT principle for detecting ellipses is the memory- and computation-intensive five-dimensional parameter space (usually consisting of 2 parameters for an ellipse center, a rotation angle as well as ellipse width and height) that has to be used. As a consequence more recent papers describe specialized algorithms for ellipse detection by HT. Yip et al.[66] propose a multi-step approach using a two-dimensional accumulator array for more efficient calculation time. In [67] Yoo et al. show an ellipse detection method from the polar definition of conics while Wu et al.[65] introduce an elliptical object detection method by using geometric properties. These HT-based approaches have in common, that they use geometric information to reduce the originally five-dimensional parameter space.

A fast method for ellipse detection using a hierarchical approach besides parameter space decomposition is proposed in [22], it is especially suitable if there are multiple ellipses of different sizes in an image. A different approach for ellipse detection was presented in [39] where choosing random points for HT was taken into account for better efficiency. A promising approach on the way to real-time applications is the k-RANSAC algorithm for ellipse detection[8].

A Fast Ellipse Detector Using Geometric Symmetry

For this work another HT-based approach for detecting ellipses was chosen[26]. The reason to work with this paper was that it stated to be a fast approach which is easy to implement and quite robust if few ellipses appear in the image. Due to the fact that most of the ventricle images show no ellipses of significant size with exception of the images at end-diastole (which are tried to be detected), this last condition seems to be fulfilled appropriately. A further advantage is that no edge direction has to be calculated for the algorithm, this fact leads to a better accuracy.

The basic concept of the proposed algorithm, a reduction of HT's five-dimensional parameter space, is similar to other algorithms. Nevertheless the way to achieve this goal is different. First of all a global geometric symmetry is used for finding potential ellipse center points. Then all feature points in the image are classified into several subimages with respect to these center points. These

subimages, of which the ellipse center is known, are transformed into a three-dimensional parameter space (consisting of ellipse rotation angle as well as ellipse width and height) and a three-dimensional HT is applied.

Summing up the algorithm consists of two stages, the *symmetric center location phase* and the *parameter estimation phase*.

Phase 1: Symmetric Center Location Starting with an image f an edge detection mechanism is used[55] (e.g. sobel filter, Laplacian of Gaussian or Canny edge detector) to find boundary points of potential ellipses. The resulting image F is taken as input for the symmetric center location step. First of all a blank binary image G is initialized. Afterwards a horizontal scanning procedure is applied on the image F, this means that image rows are scanned from left to right. For each boundary point (i,j) of F all other points (k,j) in the same row are looked at and the pixel (u,j) with $u = \frac{i+k}{2}$ is set to one in the binary image G (see Figure 4.4a). Repeating this step for all rows in the image leads to a resulting binary image G which holds information about potential symmetric vertical axes points of an ellipse. (Symmetric to horizontal image rows.) On the points of image G an ordinary Hough transform[55] is applied to extract lines. Now for each extracted line l_v all symmetric points in F relative to l_v are grouped into a subimage F_h.

This procedure is repeated for each F_h produced by the last step. After initializing a blank image G again, vertical scanlines (see Figure 4.4b) are scanned from top to bottom. For each boundary point (i,j) of F_h all other points (i,k) in the same column are looked at and the pixel (i,u) with $u = \frac{j+k}{2}$ is set to one in the binary image G. This step is repeated for all columns in the image leading to a binary image G which holds information about potential symmetric horizontal axes points. (Symmetric to vertical image columns.) Again applying a Hough transform on the points of image G extracts lines l_h. Grouping all symmetric points in F_h relative to l_h into a subimage F_{hv} leads to a number of subimages which contain possible ellipse points symmetric to a specific ellipse center. The cross point of l_h and l_v is a valid ellipse center point, this is shown in the paper[26] by proving three theorems.

Phase 2: Parameter Estimation Now that we have reduced our initially 5-parameter problem to a 3-parameter problem by finding candidate ellipse center points, it is necessary to extract the remaining three parameters (a: half-length of the ellipse major axis, b: half-length of the ellipse minor axis and Θ: the ellipse rotation angle) for each subimage F_{hv}. The equation for an ellipse translated to the center of the local coordinate system (x=0,y=0) is expressed like this:

$$dx^2 + exy + fy^2 = 1 \tag{4.1}$$

with d,e and f being unknown parameters. It would be possible to utilize these three parameters for the 3D accumulator array but in order to be able to work with more depictive parameters it is necessary to transform them into a,b and Θ. Therefore we need to know at least three points lying on the ellipse and after substituting them into Equation 4.1 we can directly solve the resulting three by three equation system with respect to the unknowns d,e and f. Directly solving this equation system has a big disadvantage, because there are many possible point configurations where the three by three matrix is either singular or leads to a numerically instable result. This is the case if points lie symmetrically on the local coordinate axes or if the ellipse is not rotated or degrades to a circle. For this reason the implementation of the proposed algorithm is slightly changed in this point. Singular value decomposition is used for solving the equation system. As a consequence of the former algorithm steps we have a total of six points from which we know that they lie on the ellipse. These six points define an overdetermined six by three equation system, which can be solved by utilizing singular value

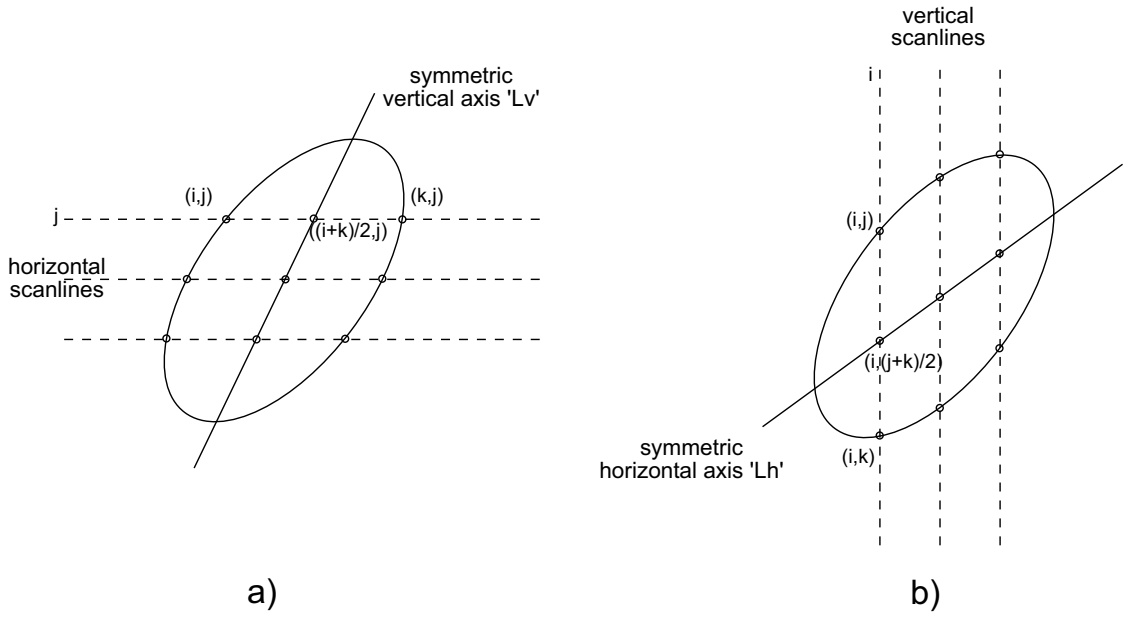

Figure 4.4: Ellipse Hough transform scanlines. a) shows horizontal scanlines. For each point (i,j) on the scanline all other points of the row (k,j) are investigated and center points $(\frac{i+k}{2}, j)$ are calculated. Applying the Hough transform on these center points extracts potential symmetric vertical axes. b) shows the same procedure for vertical scanlines leading to potential symmetric horizontal axes. (Taken from[26].)

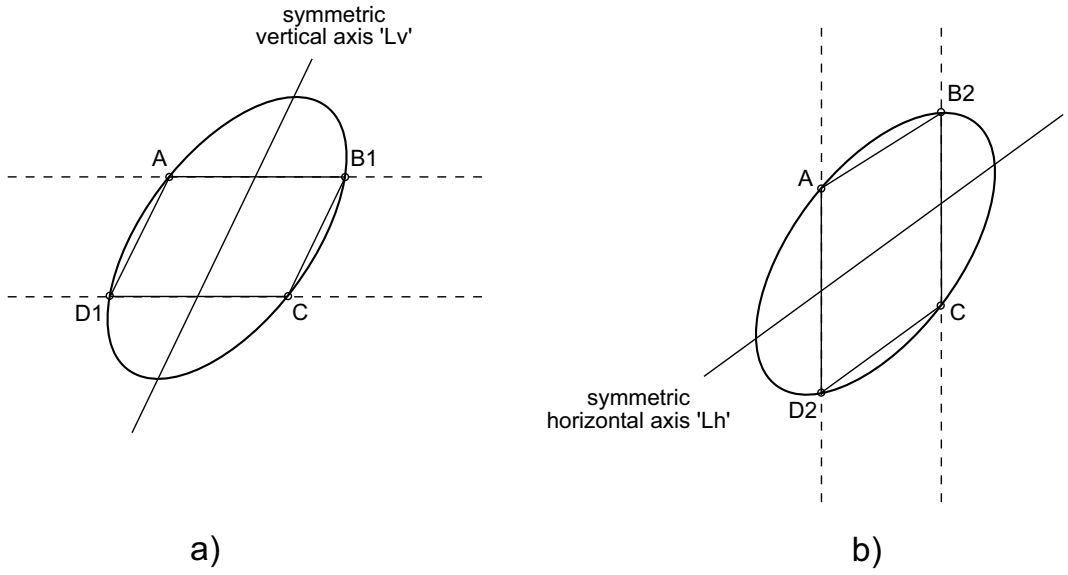

Figure 4.5: Ellipse point quadruple forming a parallelogram. Boundary point A and its symmetric point C relative to the center of ellipse E and their symmetric points relative to the symmetric vertical and horizontal axes of E form two parallelograms: (a) AB1CD1, (b) AB2CD2. (Taken from[26].)

decomposition[44]. The result is a parameter triple resembling the best approximated ellipse given these six points in a least-squares manner.

Afterwards the parameters a,b and Θ can be obtained by the following equations:

$$\Theta = \frac{\tan^{-1}(\frac{e}{d-f})}{2} \tag{4.2}$$

$$a = \sqrt{\frac{1}{d * \cos^2\Theta + e * \sin\Theta * \cos\Theta + f * \sin^2\Theta}}$$

$$b = \sqrt{\frac{1}{f * \cos^2\Theta - e * \sin\Theta * \cos\Theta + d * \sin^2\Theta}}$$

The triple (a,b,Θ) is considered a possible ellipse configuration and the corresponding entry of the accumulator array A(a,b,Θ) is increased. Finding local maxima in the accumulator array after it has been filled leads to the ellipse parameters for subimage F_{hv}. The ellipse with the largest entry in all accumulation arrays of the subimages is considered to be the most important ellipse of the image. This entry can be found by common techniques for locating local maxima in accumulation arrays, which have to be used in all kinds of Hough transform algorithms.

The only remaining problem is how to find six points that are located on an ellipse curve to substitute them into Equation 4.1. For this reason another theorem is proven in the paper, the result of the theorem is shown in Figure 4.5. Let E be an ellipse with center (0,0) and A be a point on E. C is the symmetric point of A relative to (0,0). B1 and D1 are the symmetric points of A and C relative to l_v and B2 and D2 are the symmetric points of A and C relative to l_h, respectively. Then AB1CD1 and AB2CD2 are parallelograms. So all we have to do is find point tuples (A,C) and corresponding points B1,B2,D1 and D2 in our subimage F_{hv} which fulfill the above condition. If such a point set is successfully located we can solve Equations 4.1 and 4.2 and use the result for increasing the corresponding accumulator array cell.

Algorithm Details

In addition to the presented concepts the ellipse detection method introduces some implementation details which have to be considered carefully. Due to the complex information in the images which, despite selecting a region of interest as a preprocessing step, is still quite difficult to work with, the Hough transform algorithm has to be implemented in a way that specifying boundaries for the searched ellipse is possible.

The Hough transform algorithm therefore gets passed some input parameters:

- minimum width of the ellipse (in pixel)
- maximum width of the ellipse (in pixel)
- width discretization, i.e. the number of possible discrete values between minimum and maximum width
- minimum height of the ellipse (in pixel)
- maximum height of the ellipse (in pixel)
- height discretization, i.e. the number of possible discrete values between minimum and maximum height

- minimum rotation angle of the ellipse (in radians, with respect to x-axis)
- maximum rotation angle of the ellipse (in radians, with respect to x-axis)
- rotation angle discretization, i.e. the number of possible discrete values between minimum and maximum rotation angle
- noise threshold for locating local maxima in the 3D accumulation array

The algorithm itself is embedded in the following schema (compare Algorithm 2) for automatically locating the image with maximum ellipse area (end-diastolic image).

Algorithm 2 Find the end-diastolic image given an image data set with 80 images.

findEndDiastolicImage(Images imgs)

1: calculate a region of interest
2: **for all** images *imgs* **do**
3: filter with a 7x7 median filter to suppress noise
4: scale image so that the iodinated blood is easily distinguishable from the ventricle muscle
5: apply a global threshold to remove grey values which are not originating from iodinated blood
6: apply a 3x3 sobel filter to obtain a gradient image
7: perform the Hough transform to detect ellipses
8: calculate the area of the largest found ellipse
9: **end for**
10: report the image containing the largest ellipse as end-diastolic image

For calculating the gradient magnitude of an image (i.e. detecting edges) one of many possible methods is to filter with a 3x3 Sobel kernel[55].

4.2 Segmentation by means of Thresholding

Image segmentation is a very important step in most image processing applications and it has to be utilized for many different kinds of problems. Segmentation means that an image is either partially or completely divided into disjoint image regions. Most of the time high-level information of the specific problem domain is needed for an accurate segmentation.

If G is an image a complete segmentation of G is a finite set of regions R_i with i=1,...,N having the following properties:

$$G = \bigcup_{i=1}^{N} R_i$$

$$R_i \cap R_j = \emptyset \qquad i = 1...N;\ j = 1...N;\ i \neq j$$

Segmentation techniques are commonly divided into three groups[55]:

Thresholding is the simplest segmentation method. It is only used if every region is more or less easily distinguishable from the image background.

Edge-based Covers a wide range of segmentation techniques which have in common that they rely on edges extracted from the image by some sort of edge-detection mechanism. Regions are defined by their borders.

Region-based These methods try to find image regions directly by applying a certain homogeneity criterion on neighbouring pixels of an image.

The basic thresholding algorithm is sufficient if the underlying scene of an image fulfills certain constraints. Image objects should be clearly distinguishable from the background and the lighting conditions should be uniform. If these or similar conditions are met, a transformation of an input image f to a segmented binary image g looks like:

$$g(x,y) = \begin{cases} 1 \text{ for } f(x,y) \geq T \\ 0 \text{ for } f(x,y) < T \end{cases}$$

with T being the grey-level threshold value, $g(x,y) = 1$ denoting an image object and $g(x,y) = 0$ denoting the background. Further it is possible to segment an image into different region classes by applying more than one grey-level threshold. This is only successful if there are different homogenous image regions. To verify this condition the image histogram provides useful information.

It is obvious that simple thresholding will not be very accurate on the CT images of the heart. There is a certain level of noise incorporated which troubles the thresholding algorithm. Furthermore the histograms of the images show that there is no trivial separation into different grey-level ranges. Especially the separation of the left ventricle from the other heart chambers is impossible, because all chambers are filled with contrast medium enhanced blood resulting in the same x-ray absorption coefficient. Nevertheless a thresholding algorithm will be applied on the CT images for comparison reasons. Figure 4.6 shows an example histogram calculated from one of the CT images.

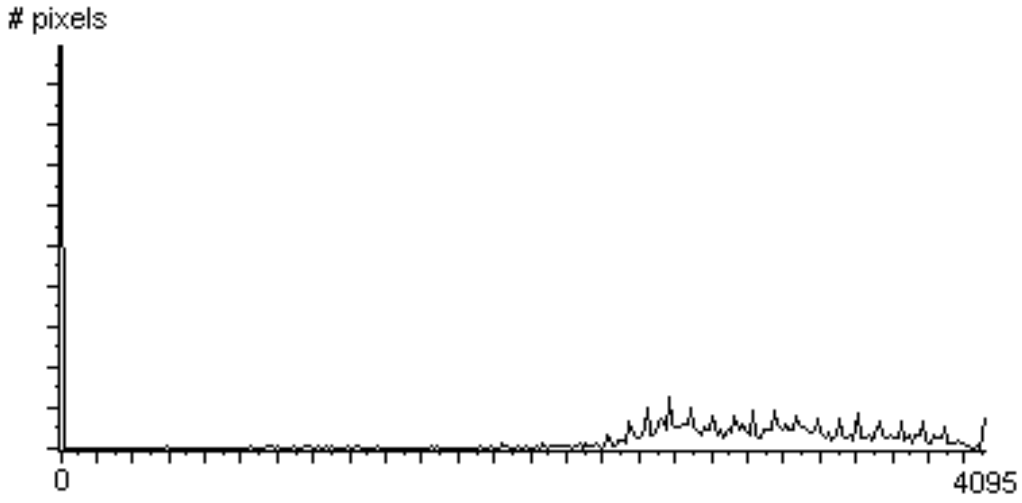

Figure 4.6: An example for a histogram taken from one of the CT images. We can see a high peak around zero representing black pixels and a continuous area at the higher grey-levels showing muscle tissue and blood.

4.2.1 Optimal Thresholding

The actual thresholding algorithm implemented for the evaluation of the CT images makes use of some ideas from optimal threshold estimation based on the histograms of images. Optimal thresholding techniques are more powerful than the simple thresholding algorithm of the previous section, a

simple iterative algorithm to select a threshold based on an image with a bi-modal histogram (i.e. more or less clearly separable background and object grey-levels) is given in [55]. Bi-modal histograms represent grey-level probability distributions which are interpreted as consisting of two Gaussian normal distributions. The mean values of these approximated Gauss-functions can be used for defining the grey-value regions to be separated.

After applying a certain window/level transformation on the CT images, three distinct grey level regions can be identified. We have black background, dark grey muscle tissue and light grey to white pixels showing blood. With this visual information we can suppose that the histograms of these images are tri-modal. So all we have to do for threshold segmentation is to locate the grey-value separating muscle tissue from blood. This procedure is described in Algorithm 3.

Algorithm 3 Find an optimal threshold approximation in an image with tri-modal histogram.

findOptimalThresholdApproximation(image *img*)

1: calculate histogram *h* of *img*
2: set the first histogram bin representing black pixels to zero
3: remove empty histogram bins by linear interpolation of the two nearest non-zero histogram values
4: smooth histogram by convolution with Gaussian kernel of size 30x1 (sigma is 4 pixels)
5: locate the two largest local maxima (*m1*, *m2*) in histogram *h*
6: sort the two local maxima (now *m1* < *m2*)
7: select the average value between maxima 1 and maxima 2 as threshold ($thr = \frac{m1+m2}{2}$)
8: apply threshold *thr* on the image

After calculating the histogram of an image the very high peak at the first histogram bin is set to zero. There are many black pixels in the image, they correspond to grey-value zero and are therefore located in the first histogram bin. To successfully smooth the histogram function with a Gaussian filter it is necessary to remove empty bins from the histogram. Empty bins are grey-levels which do not occur in the image resulting in histogram values of zero. They occur due to a preceding window-level transformation or another contrast stretching method, where a small range of grey levels is stretched to the full possible range. We use linear interpolation of the two nearest non-zero histogram values to remove empty bins. Finally we have to look for two more local maxima in the histogram function and the averaged value out of these two maxima is used as an approximation for an optimal image threshold. The reason why this threshold is only an approximation of an optimal threshold is that only mean values but no standard deviations of the assumed Gaussian normal distributions are taken into account for its determination.

It should be noted again that this kind of algorithm is quite useful for locating the blood regions of the images but fails if a more accurate segmentation is to be achieved. Therefore the actual implementation of the algorithm only works on the region of interest defined in Section 4.1.3. Nevertheless the inaccuracies of the ROI and the threshold algorithm lead to a strongly inaccurate result as can be verified in the evaluation section.

The Gaussian kernel of the algorithm's convolution stage is a 30x1 kernel representing a sampled Gaussian function between -14 and 15 with a standard deviation of 4 pixel. It is convoluted with the histogram function consisting of 4096 values.

4.3 Segmentation by means of Active Contours (Snakes)

Active Contours and more generally *Deformable Models* have a long and successful tradition in medical image processing. A lot of research went into this field in the last decade leading to many different models with applications in segmentation, shape representation, matching or motion tracking. Segmenting structures from medical images is difficult due to the sheer size of data sets and the complexity of the investigated anatomic shapes. Low-level image processing techniques are not able to consider non-local information so a considerable amount of expert knowledge usually is required for an accurate processing. Here the concept of deformable models shows its advantages by combining low-level bottom-up constraints defined by the images and high-level top-down knowledge provided by an expert.

4.3.1 Literature Survey

The most important and also most often referenced deformable model is the *Deformable Contour* or *Active Contour Model* which is popularly known as *Snakes*. It was initially presented by Kass et al.[32] in 1988 and was one of the foundations of the whole field. The following section will give a survey of the basic ideas behind *Snakes* and its formulation. Although they were originally developed for computer graphics and computer vision applications, the potential of deformable models in medical imaging has been quickly realized after their first formulation. According to [54] deformable models are distinguished into *parametric-* and *geometric deformable models*. Parametric deformable models ([32], [10] and [38]) contain explicit contour or surface representations, making it possible to interact directly with the models but lacking flexibility if splitting or merging of deformed parts is required. Opposed to this fact geometric deformable models ([6], [7] and [37]) handle topological changes naturally by implicitly representing curves and surfaces as a level set of a higher-dimensional scalar function. Some books devoted to the two kinds of deformable models are [4], [53] and [51].

During research of deformable models many extensions were proposed with the goal of increasing robustness and performance. One possible improvement is the incorporation of additional *a priori* knowledge about the problem domain by means of a training step while another improvement, which is often used in conjunction with the first one, also models *global shape properties*. Examples are *Deformable Fourier Models*[56], which represent contours as a fourier series, or *Active Shape Models*[12] and their extension *Active Appearance Models*[11], which basically leave the domain of contours or surfaces by incorporating sets of labeled points to represent structures. Many of these references were taken from [54] where even more literature references can be found for deeper investigation. Other related keywords are *United Snakes*[34], *Adaptive Snakes*[9], *Tamed Snakes*[28] or *Region Competition*[68].

In medical imaging there are few areas where deformable models have not been applied. Utilizing Active Contours ranges from segmenting leg ulcers (skin wounds)[31], segmenting vessels in angiography[60], automatically detecting the fovea in angiography[25] over bone segmentation[49][36] to segmentation of brain[16] and skull[48]. And this list is far from being complete.

Finally some applications of deformable models for segmenting heart ventricles can be found in [19], [24] and [23] as well as in the papers mentioned in the reference area of Chapter 1.4.

4.3.2 Basic Formulation of Active Contours

Active Contours or *Snakes*[32] are based on a continuous spline representing the contour. The corresponding energy minimization algorithm iteratively alters the spatial location of the spline with respect

to the coordinate system of the underlying image under the influence of internal and external forces. Mathematically, a parametric representation of a Snake's spatial position is $\mathbf{v}(s) = (x(s), y(s)), s \in [0, 1]$. With this we can write an energy functional as

$$E(\mathbf{v}) = E_{int}(\mathbf{v}) + E_{image}(\mathbf{v}) + E_{con}(\mathbf{v}). \tag{4.3}$$

Here $E_{int}(v)$ denotes the internal energy functional which is defined to be

$$E_{int}(\mathbf{v}) = \frac{1}{2}\int_0^1 \alpha(s)\left|\frac{\partial \mathbf{v}}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 \mathbf{v}}{\partial s^2}\right|^2 ds \tag{4.4}$$

and consists of a first-order derivative representing the contour's potential tendency for stretching and a second-order derivative managing its ability to bend. $\alpha(s)$ and $\beta(s)$ are weighting parameters to control the relative importance of tension and rigidity, they are often supposed to be constants.

The second term $E_{image}(\mathbf{v})$ now establishs a connection between the contour and the image by defining some image feature (or features) as an energy term

$$E_{image}(\mathbf{v}) = \int_0^1 E_{feature}(\mathbf{v}(s))ds. \tag{4.5}$$

This term is often called potential energy, examples for interesting features are the image intensity function $E_{feature\_intensity}(x, y) = -w_i I(x, y)$ or the gradient of the image

$$E_{feature\_edge}(x, y) = -w_e \left|\nabla \left[G_\sigma(x, y) * I(x, y)\right]\right|^2 \tag{4.6}$$

where w_i and w_e are positive weighting factors, $G_\sigma(x, y)$ is a two-dimensional Gaussian function with standard deviation σ, $*$ is the two-dimensional convolution operator and ∇ the gradient operator.

The last term $E_{con}(\mathbf{v})$ denotes external constraint energies that can be used for user interaction like defining fixed contour points or including so-called "springs" and "volcanoes". These two terms stem from Kass et al. who built an user interface for Snakes and defined springs as means for pulling the Snake to a desired image location and in opposition included volcanoes from which contours are pushed away. Local minima can be avoided using these tools.

Potential energy and constraint energy often are referred to as external energy

$$E_{ext}(\mathbf{v}) = E_{image}(\mathbf{v}) + E_{con}(\mathbf{v}). \tag{4.7}$$

The energy minimization process, which can be understood as a balancing of the three weighted energy terms until a location of minimum energy is found, is independent from the detailed representation of the energy functionals. It involves the problem of finding a curve $\mathbf{v}(s)$ that minimizes Equation 4.3 leading to a problem of variational calculus. Therefore $\mathbf{v}(s)$ must satisfy the Euler-Lagrange equation:

$$\frac{\partial}{\partial s}(\alpha\frac{\partial \mathbf{v}}{\partial s}) - \frac{\partial^2}{\partial s^2}(\beta\frac{\partial^2 \mathbf{v}}{\partial s^2}) - \nabla E_{ext}(\mathbf{v}) = 0 \tag{4.8}$$

in the case where the weight functions $\alpha(s)$ and $\beta(s)$ are both constant. When they are not constant a discrete formulation of the energy functional is chosen by approximating derivatives with finite differences. The bending and rigidity terms of the internal energy are approximated like this

$$\left|\frac{\partial \mathbf{v}_i}{\partial s}\right| \approx |\mathbf{v}_i - \mathbf{v}_{i-1}|^2 = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \tag{4.9}$$

$$\left|\frac{\partial^2 \mathbf{v}_i}{\partial s^2}\right| \approx |\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}|^2 = (x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2 \tag{4.10}$$

See the appendix of [32] for a more detailed explanation of the necessary numerical steps for solving the equation.

4.3.3 Greedy Snake - A Fast Algorithm for Active Contours

There are some problems with the implementation of Snakes following the paper by Kass et al[32]. The involved numerical minimization procedure from variational calculus is unstable and contour points have the tendency to bunch up on strong feature portions. Therefore Amini et al.[2] proposed a dynamic programming algorithm for minimizing the energy functional resulting in a more stable but slower method. As a consequence Williams et al.[63] presented a fast "greedy" algorithm and showed that its stability and flexibility is equal to the dynamic programming schema while its speed is more than an order of magnitude faster. This greedy algorithm was chosen to be implemented in this work because of its straight-forward implementation without losing stability and robustness.

The first remark stated in the paper is an enhancement of the discretization of the bending term. It is shown that Equation 4.9 is not properly defined. Using $|\mathbf{v}_i - \mathbf{v}_{i-1}|^2$ as bending term causes the curve to shrink, because this is actually minimizing the distance between two points. Further it encourages the effect of points bunching up over strong feature regions. For this reason the bending term looks as follows in the greedy approach:

$$\left|\frac{\partial \mathbf{v}_i}{\partial s}\right| \approx d_{avg} - |\mathbf{v}_i - \mathbf{v}_{i-1}| = d_{avg} - \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{4.11}$$

where d_{avg} is the average distance between adjacent points of the contour.

Additionally it is stated that the curvature (or rigidity) term of Equation 4.10 is only meaningful if the points of the contour are relatively evenly spaced. This is not necessarily the case when using $|\mathbf{v}_i - \mathbf{v}_{i-1}|^2$ as bending term discretization. But the modification described in Equation 4.11 ensures evenly spaced contour points.

The implementation of the greedy approach doesn't guarantee to find global energy minima due to the fact that it only takes into account a local neighbourhood of the currently processed contour point. The local neighbourhood naturally implies that contour points can only move on the discrete grid of the underlying image during evaluation of the algorithm as opposed to the numerical minimization by variational calculus. It is an iterative algorithm, during each iteration a local neighbourhood is investigated for every contour point. The energy minimization step is performed at each neighbour point and the processed contour point is moved to the location yielding minimum energy. So-called fixed points are supported, it is possible to specify points with weights of zero to inidcate that they are not allowed to change location. Algorithm 4 demonstrates the pseudo code of the greedy approach and Figure 4.7 shows how the algorithm works.

The outmost loop of the algorithm iteratively checks if a convergence criterion is already fulfilled. The implementation supports two convergence criteria, a minimum number of points being moved during an iteration step and a maximum number of iteration steps. It is not sufficient to work only with the first criterion, because the algorithm sometimes produces a few points which are constantly moving between two adjacent pixel locations even if the rest of the contour has already converged. Calculation of d_{avg} simply happens in a loop over all points v_i summing up their x- and y-coordinates

Algorithm 4 Perform the greedy Snake algorithm on an image given an initial contour.

greedySnake(source image *img*, initial point list *v[0...N-1]*)

while not yet converged **do**
  calculate d_{avg}, the average distance between adjacent points *v[i], v[i-1]*
  for all contour points *v[i], i=0...N-1* **do**
    if *v[i]* is a fixed contour point **then**
      skip actions on *v[i]*, coninue with next contour point
    end if
    for all locations in the local *m*x*m*-neighbourhood of *v[i]* called *v[i]'* **do**
      calculate $E_{bend} = |d_{avg} - |v[i]' - v[i-1]||$
    end for
    normalize E_{bend} to lie between [0,1] by multiplying E_{bend} with $\frac{E_{bend} - E_{bend_{min}}}{E_{bend_{max}} - E_{bend_{min}}}$
    for all locations *v[i]'* **do**
      calculate $E_{curv} = |v[i-1] - 2 * v[i]' + v[i+1]|^2$
    end for
    normalize E_{curv} to lie between [0,1] by multiplying E_{curv} with $\frac{E_{curv} - E_{curv_{min}}}{E_{curv_{max}} - E_{curv_{min}}}$
    for all locations *v[i]'* **do**
      calculate E_{img} = positive gradient magnitude of image *img* at location *v[i]'*
    end for
    normalize E_{img} to lie between [-1,0] by multiplying E_{img} with $\frac{E_{img_{min}} - E_{img}}{E_{img_{max}} - E_{img_{min}}}$
    for all locations *v[i]'* **do**
      calculate E as weighted sum: $E(v_i') = \alpha * E_{bend}(v_i') + \beta * E_{curv}(v_i') + \gamma * E_{img}(v_i')$
    end for
    move investigated point *v[i]* to that neighbourhood location with minimum E
    perform an optional high-level feedback step making sharp contour corners possible
    check for convergence (maximum number of iterations, minimum number of moved points)
  end for
end while

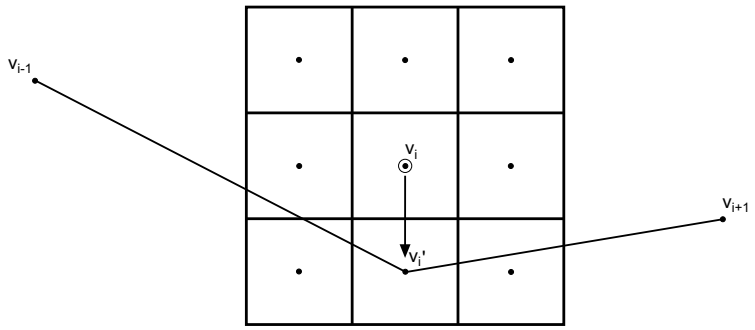

Figure 4.7: Greedy Snake algorithm example. The energy function is computed at $\mathbf{v}_i$ and each of its eight neighbours. The point before and after it on the contour are used in computing the continuity and smoothness constraints. The image energy comes from the gradient magnitudes of the underlying pixel image. The location having the smallest energy value is chosen as the new position of $\mathbf{v}_i$, here it is $\mathbf{v}_i'$.

followed by a division by the total number of points. After calculating E_{bend} and E_{curv} a normalization step is indicated. Multiplying each energy term with the factor $\frac{E_{\min}-E}{E_{\max}-E_{\min}}$, where $\mathrm{E}_{\min}$ and $\mathrm{E}_{\max}$ denote minimum and maximum values in the local neighbourhood, makes sure that they lie between 0 and 1. Here a little difference to the description in the paper occurs, because we don't use the suggested multiplication factor $\frac{E}{E_{\max}}$. It is an absolute measure and it would lead to a scaling of the value domain producing an unwanted influence on the weights. Normalization between 0 and 1 is a prerequisite for building a weighted sum of the energy terms later on, because meaningful weighting needs terms lying in the same domain. The image energy, which is derived from the gradient magnitude of the source image by utilizing a floating point Sobel filter[55], is normalized by multiplying the positive energy terms with the factor $\frac{E-E_{\min}}{E_{\max}-E_{\min}}$. Again $\mathrm{E}_{\min}$ and $\mathrm{E}_{\max}$ denote the minimum and the maximum in the local neighbourhood. The slight difference in the numerator compared to the other two normalization factors cause a normalization between -1 and 0, where high gradient magnitudes correspond to small values. Note that the different ranges of the factors have no influence on the minimization procedure, it is only a translation by -1 in the energy value domain. If $E_{\max} - E_{\min} < 5$ then $E_{\min}$ is assigned the value $E_{\max} - 5$ to prevent large differences where gradient magnitude is nearly uniform. As a consequence of all these considerations the Snake is attracted to regions with strong edges. After calculating the weighted sum of the energy terms, finding the minimum in the local neighbourhood and moving the currently processed point to its location, an optional high-level feedback step is included in the schema. In this step the curvature at each point on the new contour is determined by the following formula

$$c = \left| \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{|\mathbf{v}_i - \mathbf{v}_{i-1}|} - \frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{|\mathbf{v}_{i+1} - \mathbf{v}_i|} \right|^2$$

and if the curvature is a maximum, larger than a noise threshold and the gradient at this location is larger than a certain threshold, the β-weight of the investigated point is set to zero. This allows the contour to form sharp corners at curvature maxima with strong underlying edges.

The Snakes paradigm is a powerful tool to combine high-level knowledge of a problem domain and low-level image information for segmentation purposes. Therefore many kinds of problems can be

handled with the same model by laying emphasis on high-level or low-level information respectively. Furthermore it is possible to include automatization steps into a segmentation algorithm based on Snakes. A disadvantage of the algorithm is the high number of parameters that have to be tuned.

4.4 Segmentation by means of Graph-Theoretic Border Detection (LiveWire)

In image analysis and especially in medical image segmentation there are situations when automatic and certain semi-automatic segmentation techniques fail or lead to non-optimal solutions. As a consequence a user has to correct results in a manual fashion. Providing the user with a tool that supports manual segmentation by speeding up the process, giving immediate feedback and making the results more repeatable would be a far better choice in such a case.

Therefore graph-theoretic approaches for border detection were researched beside others by two groups in the 1990's. Falcao et al.[18] proposed their *Live Wire* and *Live Lane* segmentation paradigms and at the same time Mortensen et al.[40] presented a similar principle, called *Intelligent Scissors*, also based on the Live Wire concept. Both groups cooperated in the research of the basics behind *Live Wire* but they presented their outcomes independently. Falcao et al. later proposed a faster version of their *Live Wire* algorithm[17].

An example for another related paper is by Thedens et al.[59], who applied graph searching methods on image sequences, taking temporal and spatial information into account. They tested their approach on magnetic resonance images showing short-axis projections of the heart and extracted myocardial borders with their method.

4.4.1 The Live Wire Segmentation Paradigm

Image segmentation techniques are currently classified into two groups. *Automatic* segmentation techniques avoid user interaction while *interactive* techniques support the user in manually segmenting images. Due to the fact that the first group simply doesn't work yet for certain kinds of problems, optimizing user-steered techniques seems to be an important goal. A tight and active control of the user during the interactive segmentation process would significantly reduce the total time spent on the process.

Basically the segmentation process can be considered to consist of two tasks. Object *recognition* and object *delineation*. Recognition is the task to roughly decide where an object is located, this is a comparatively easy task for an human operator but very difficult for a machine (an algorithm). On the other side an exact delineation of the roughly recognized object is an easy task for a machine as opposed to an human. As a consequence it appears to be an effective strategy to exploit the synergy between human operator and computer algorithm for interactive segmentation. This strategy is the basis for the *Live Wire* approach.

The *Live Wire* algorithm utilizes methods from graph theory. An image or generally a pixel array constitutes a directed graph where the pixel vertices are graph nodes and oriented pixel edges represent edges of the graph. Graph edges are weighted with costs which are derived from image gradient information. The basic problem of finding a boundary segment is therefore converted to finding a minimum-cost path between start and end vertex of the segment. To find this optimal path *dynamic programming* is used.

Initially the user has to specify a start point on a boundary of the image. Now while moving the mouse cursor around, a globally optimal path connecting initial and current point is computed and displayed. By moving the mouse cursor close to a boundary, the *Live Wire* snaps onto it, due

to weighting the graphs's edges with image gradient information. It is possible to include intelligent features like automatic selection of new start points along a non-changing curve segment or on-the-fly training of gradient values.

4.4.2 The Intelligent Scissors Algorithm based on the Live Wire Paradigm

An image is represented as a 2D scene **C,** being a pair (C, g) where C is a finite rectangular array of pixels and $g(p) : \ C \rightarrow [L, H]$ is a function mapping each pixel $p \in C$ to an intensity value. A pixel p represents a node of the underlying graph. The edges of this graph are defined differently in the two papers mentioned above. Falcao et al.[17] propose edges to lie exactly on the crack between two pixels, having a certain direction. Therefore they define a bel (short for boundary element) as a pair of 4-adjacent pixels (or graph nodes). Every $bel = (p, q) \in C$ has a location (depending on the locations of pixels p and q) and an orientation, which is defined so that p is always inside an image boundary and to the left of the bel.

In opposition Mortensen et al.[40] simply define graph edges as the connection of two 8-adjacent pixels. The remaining part of this section will only deal with the definitions and considerations in Mortensen's paper, because the implemented algorithm was mainly inspired by this paper. In both papers a local cost function is assigned to the edges to weight their probability of being included in an optimal path. Mortensen et al. present six feature components derived from an underlying image:

$f_Z(q)$ Multi-scale Laplacian of Gaussian (LoG) zero crossings at pixel q

$f_G(q)$ Gradient magnitude at pixel q

$f_D(p, q)$ Gradient direction with respect to pixels p and q

$f_P(q)$ Intensity value at edge pixel q

$f_I(q)$ Intensity value at the "inside" of the boundary of which pixel q is part of

$f_O(q)$ Intensity value at the "outside" of the boundary of which pixel q is part of

A local cost function $l(p, q)$ is basically derived like this:

$$l(p, q) = \omega_Z * f_Z(q) + \omega_G * f_G(q) + \omega_D * f_D(p, q) + \omega_P * f_P(q) + \omega_I * f_I(q) + \omega_O * f_O(q)$$

Each ω is the weight of the corresponding feature function. These weights can easily be adjusted but it should be mentioned here that it is much easier to find weights which work well in a wide variety of images than in the Snake algorithm. Further it is possible to adjust some weights automatically by training steps.

Local cost feature functions

The Laplacian zero crossing and the two gradient features are static cost functions, they don't change during algorithm runs and can be pre-computed for an image. The other three cost functions are dynamic, they are only incorporated if there is an on-the-fly training step. In this case they will be adapted during this step.

Laplacian zero crossing is an edge-detection operator leading to a binary image as result, with detected gradients being represented by a logical 1. This binary result depends on a certain standard deviation chosen for the LoG operator[55]. The standard deviation σ is reflected by the kernel

size of the LoG convolution kernel, e.g. a σ of $\frac{1}{3}$ pixel leads to a 5x5 kernel. To get a multi-scale Laplacian operator it is possible to combine the binary results of different standard deviations (i.e. convolutions with kernels of different sizes) by a weighted summation, with the weights referring to the "importance" of the scale. LoG is an operator for edge localization which seeks zero crossings in the second derivation of the Gaussian-smoothed image. These zero crossings are represented by oppositely signed Laplace responses of neighbouring pixels. In the paper using a 5x5 and a 9x9 LoG kernel is proposed, the summation of both results is performed with weights of 0.45 and 0.55 respectively. The cost function $f_Z(q)$ calculates to

$$f_Z(q) = \begin{cases} 1; & \text{if } I_L(q) = 0 \\ 0; & \text{if } I_L(q) \neq 0 \end{cases}$$

where $I_L(q)$ is the weighted sum of the Laplacian zero crossings at pixel q. This means that f_Z is zero if and only if the Laplacian from each kernel gives a zero-crossing at pixel q and one if no kernel gives a zero-crossing, otherwise $0 < f_Z < 1$.

The *magnitude of the image gradient* is necessary for distinguishing between stronger and weaker edges, in contrast to the Laplacian operator where we have a binary output for the occurrence of an edge. It is computed by approximating the partial derivatives of an image in x and y direction. The paper proposes to utilize derivatives of Gaussian kernels with different scales. If I_x and I_y are the result images representing the partial derivatives, then the gradient magnitude G is equal to $\sqrt{I_x^2 + I_y^2}$. Due to the fact that the cost function $f_G(q)$ needs to be low for large gradients (strong edges), the static function is computed by subtracting each gradient value from the image's gradient maximum followed by a division by this maximum. As a consequence we get a function scaled between 0 and 1.

$$f_G \; = \; \frac{\max(G') - G'}{\max(G')} = 1 - \frac{G'}{\max(G')}$$

Here $G\prime$ is equal to $G - min(G)$. Similar to the Laplacian function it is proposed to use different scales. But in this case the results are not summed up, the most appropriate result is chosen for each pixel independently. A possible criterion is to choose for each pixel the kernel leading to the largest gradient magnitude.

Gradient direction is strongly related to the gradient magnitude feature but adds a certain degree of smoothness to the boundary definition. Sharp changes in boundary direction are assigned high costs. The gradient direction is defined by the angle between the partial derivatives I_x and I_y. Suppose $D(p) = [I_x, I_y]^T$ denotes the unit vector of gradient direction at a point p. Then $D(p)$ is exactly perpendicular to the boundary tangent vector $D'(p) = [-I_y, I_x]^T$. Please note that $D(p)$ and $D'(p)$ have to be unit vectors so we have to normalize the partial derivatives I_x and I_y before establishing the vectors. Now the feature cost $f_D(p, q)$ is formulated as

$$f_D(p, q) = \frac{2}{3\pi} \left\{ \cos^{-1}[d_p(p, q)] + \cos^{-1}[d_q(p, q)] \right\}$$

where

$$d_p(p, q) = D'(p) \bullet L(p, q)$$
$$d_q(p, q) = L(p, q) \bullet D'(q)$$

are vector dot products and

$$L(p, q) = \frac{1}{\|p - q\|} \begin{cases} q - p & \text{if } D'(p) \bullet (q - p) \geq 0 \\ p - q & \text{if } D'(p) \bullet (q - p) < 0 \end{cases}$$

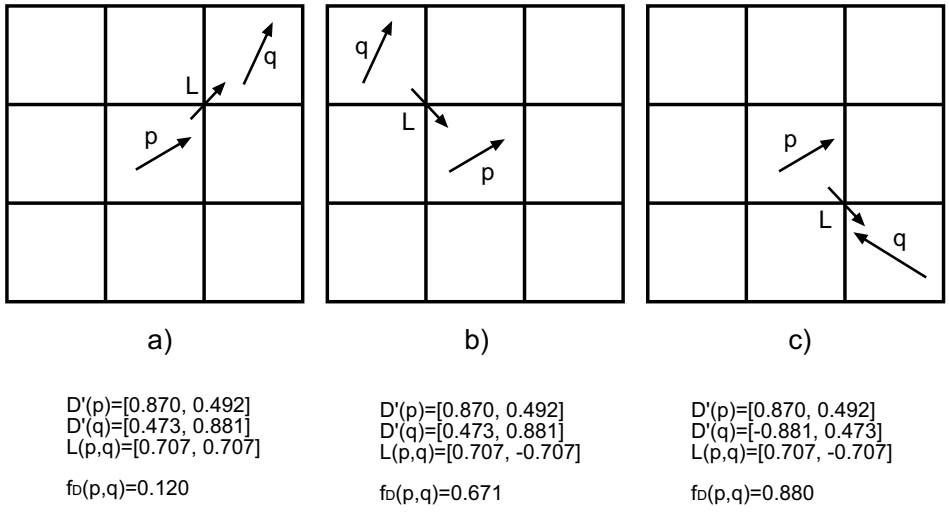

Figure 4.8: Three examples for Live Wire gradient direction computation. The formula for f_D is: $f_D(p,q) = \frac{2}{3\pi}\left\{\cos^{-1}[D'(p) \bullet L(p,q)] + \cos^{-1}[L(p,q) \bullet D'(q)]\right\}$. The arrows near pixels p and q indicate gradient directions. In case a) the gradient directions of p and q are similar to each other and to the link L(p,q) between them leading to a low cost function value. In b) the gradient directions are similar to each other but not to L and in c) all three directions are different. As a consequence the function cost values are larger. Taken from [40].

is the direction of the link between p and q so that the difference between p and the direction of the link is minimized. In other words $L(p,q)$ is defined so that it points in the same direction as $D'(p)$. Some example computations are given in Figure 4.8. The purpose of these calculations is to associate low costs to pixel pairs (p,q) where the two gradient directions at the pixels and the direction of the vector linking the two pixels $L(p,q)$ are similar.

If we do not utilize the on-the-fly training step, these three feature functions have to be combined to form a *static* local cost function $l_S(p,q)$.

$$l_S(p,q) = \lfloor \omega_Z * M * f_Z(q) + 0.5 \rfloor + \lfloor \omega_N * \omega_G * M * f_G(q) + 0.5 \rfloor + \lfloor \omega_D * M * f_D(p,q) + 0.5 \rfloor$$

In this function p and q is a pair of adjacent pixels. ω_Z, ω_G and ω_D are the relative weights of the feature components. Their sum is 1 if no on-the-fly training is incorporated, otherwise the sum is lower than 1. The weight ω_N is a compensation factor for the gradient magnitudes which introduces the differing Euclidean distance of the diagonal and the direct neighbours.

$$\omega_N(p,q) = \begin{cases} 1; & \text{if } p \text{ and } q \text{ are diagonal neighbours} \\ \frac{1}{\sqrt{2}}; & \text{if } p \text{ and } q \text{ are direct neighbours} \end{cases}$$

M is an integer value representing the discretization of the cost function. Due to the fact that the graph searching algorithm utilizes a discrete number of cost function values for performance reasons, it is necessary to map the feature function values lying between 0 and 1 to a discrete range. This is performed by multiplying the weights with M and rounding each weighted function to its nearest integer value. E.g. if ω_Z is 0.25, M is 256 and $f_Z(q)$ lies between 0 and 1, the resulting discrete feature cost function $\lfloor \omega_Z * M * f_Z(q) + 0.5 \rfloor$ has a range between 0 and 64.

On-the-fly training of gradients

The pixel value features f_P, f_I and f_O only have meaning after training. $f_P(p)$ simply represents the underlying intensity value of the image at pixel p. This value is scaled to lie between 0 and 1 by dividing it by its maximally possible value, i.e. 255 for 8 bit grey-scale images. The inside and outside pixel values f_I and f_O for a pixel p are samples at a distance k from p in the gradient direction or the opposite direction, respectively.

$$f_P(p) = \frac{1}{2^8 - 1} I(p)$$
$$f_I(p) = \frac{1}{2^8 - 1} I(p + k * D(p))$$
$$f_O(p) = \frac{1}{2^8 - 1} I(p - k * D(p))$$

$D(p)$ denotes the unit vector of the gradient direction and k is a constant value defining a sampling distance. The factor $\frac{1}{2^8-1}$ has to be replaced by an appropriate term if images with a grey-scale resolution different from 8 bit are used. Bilinear interpolation of four surrounding points of $p \pm k * D(p)$ is used for exact determination of the intensity $I(p \pm k * D(p))$.

Due to the fact that the Live Wire algorithm alreay performed very well on the images without incorporating a dynamic cost map in addition to the static one, no implementation of the on-the-fly gradient training was carried out. So we do not give any more descriptions on this topic but instead refer to the paper [40] for this topic.

Optimal path searching

Finding optimal paths (i.e. paths which minimize the total cost) in graphs can be solved by Dijkstra's algorithm[13]. This algorithm solves the single-source shortest-path problem on a weighted, directed graph for the special case where edge weights are non-negative. The graph-search algorithm is initialized with a start or seed point $\mathbf{s}$, with a cumulative cost of 0, being placed on an initially empty list L, called active list. As an operator may choose an arbitrary point as goal pixel (and does it subsequently during mouse movement), the optimal paths from the seed point to all pixels of the image have to be computed and stored for later usage. Points p are placed on the active list in a sorted order based on the cumulative cost, i.e. the sum of all local costs of the corresponding path from p to the seed point $\mathbf{s}$. Therefore all other points are initialized with an infinite cost and an iterative creation of a minimum cost spanning tree based on the local cost function is carried out. In each iteration the point p with minimum cumulative cost is removed from L and the total cost of its not yet removed neighbours is computed. Then these neighbours are inserted into the active node list. Algorithm 5 demonstrates the procedure to create a minimum-cost map with respect to a given seed point.

Now a detailed description of this algorithm shall be given. First of all there are various data structures involved. The result of the algorithm is a map of minimum cost pointers. This means that we have a 2D array of the same size as the underlying image. Each array element holds information about a direction. This direction can be one of { North, NorthEast, East, SouthEast, South, SouthWest, West, NorthWest }. As a consequence by indexing the minimum cost pointer map with a certain pixel we get the direction of the minimum-cost neighbour pixel. This leads us instantaneously to the trivial part of the Live Wire algorithm, reporting a minimum path between two given points, as shown in Algorithm 6.

Then we have a status map and a cumulative cost map. Both maps are also 2D arrays with the same size as the underlying image. The cumulative cost map stores for each pixel of the image the

Algorithm 5 Create Live Wire minimum-cost map with respect to a given start point.

liveWireInit(Image *img*, Point *start*, MinimumCostMap *min_cost_map*)

pre-calculate static local cost function $l_S(p, q)$
calculate *NR_BUCKETS*, the number of distinguished discrete costs
NR_BUCKETS $= M * \sqrt{img\_width^2 + img\_height^2}$
initialize *L*, the sorted list of active nodes, a vector with *NR_BUCKETS* elements
each element is a stack of points
initialize a cumulative cost map *cum_cost_map*
of size $(img\_width * img\_height)$
each entry in the *cum_cost_map*
is an integer value denoting the point's current cost
initialize a *status_map* of size $(img\_width * img\_height)$
each entry in this status map is one of {NotProcessed, Expanded, Visited}
the status map is initialized with NotProcessed for all points
cum_cost_map[start] = 0
status_map[start] = Expanded
insert point *start* into *L[0]*
while *L* is not empty **do**
    remove point *p* with minimum cost from active nodes *L*
    status_map[p] = Expanded;
    for all 8-adjacent neighbour points *q* of *p* with *status_map[q]* != Expanded **do**
        $cost =$ *cum_cost_map[p]* $+ l_S(p, q)$ # compute neighbours cumulative cost
        if *status_map[q]* == Visited AND *cost* < *cum_cost_map[q]* **then**
            remove *q* from active nodes *L*
            status_map[q] = NotProcessed
        end if
        if *status_map[q]* == NotProcessed **then**
            cum_cost_map[q] = cost
            min_cost_map[q] = direction pointing to *p*
            insert/re-insert point *q* into *L[cost]*
        end if
    end for
end while

Algorithm 6 Report shortest path between start and end point.

reportShortestPath(Point *start*, Point *end*, MinimumCostMap *min_cost_map*, PointList *shortest_path*)

clear *shortest_path*
current_point = end
insert *current_point* into *shortest_path*
while *current_point* != *start* **do**
    direction = min_cost_map[current_point]
    move *current_point* according to *direction*
    insert *current_point* into *shortest_path*
end while

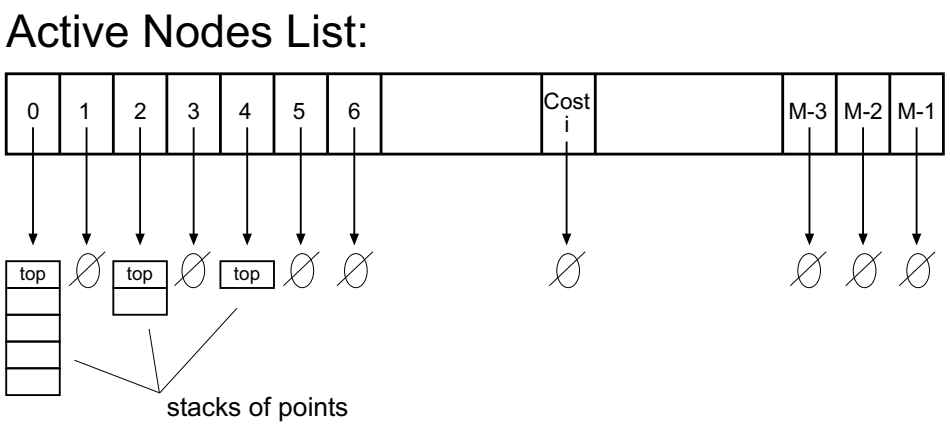

Figure 4.9: Live Wire list of active nodes. This figure illustrates the sorted list of active nodes data structure. It consists of an array of M "buckets", each of which is pointing to a stack of points with the same cumulative cost value. This value is also the index of the array. In this example only 3 buckets have non-empty stacks.

currently evaluated cost. So the entry at the start point's position is set to 0 at the beginning and after the algorithm has finished, the cumulative cost map stores the *minimum* local cost of each pixel with respect to the neighbours of the pixel. The cost map is supported by the status map, where one of the following status values is stored for each pixel: { NotProcessed, Expanded, Visited }. At the beginning of the algorithm all pixels are assigned status 'NotProcessed' (this means that the pixel has an infinite cumulative cost), except the start pixel which is 'Expanded'. During iteration status 'Expanded' means that the pixel was already removed from the active nodes list while 'Visited' means that the pixel has already been member of a set of neighbours. Further there is the set of neighbours holding up to eight neighbours of an 'Expanded' pixel.

The most important data structure is the sorted list of active nodes L, which is shown in Figure 4.9. It is an array with NR_BUCKETS entries, where NR_BUCKETS denotes the number of possible discrete local cost values multiplied with the maximum distance in the image, the image diagonal $\sqrt{img\_width^2 + img\_height^2}$. For each possible cost value $c \in [0...M-1]$, $L[c]$ stores a list of points with equal cumulative path costs. As such the order of the points is arbitrary, therefore they are organized as a stack.

After inserting the start point into the active nodes list L at index 0 (start point has a cumulative cost of 0), the main while-loop of the algorithm performs its iterations until L is empty. This happens after all pixels of the image have been processed. During each iteration first of all the point with the smallest cost value p is removed from L and its status is set to 'Expanded', i.e. there is no need to further process this point.

The special structure of the active nodes list makes it possible to do the removal in nearly constant time. We only have to verify if the element with the current index (starting with zero) still holds a stack of points. If so we can pop one point from the stack and report it. If not we have to move on seeking the next-higher index of the active nodes list which holds a stack of points and remove the top point from there. Obviously this seeking procedure can not be performed in constant time, but the insertion of new points is done in a way that there is always a concentration of points near the currently processed index. The reason for this is that newly inserted points do always have a slightly higher cost than the currently processed point, because they are a little further away from the seed point.

The next step in the while-loop is to determine a set of neighbours of p. All 8-adjacent neighbour pixels with status equal to 'NotProcessed' or 'Visited' are investigated. For each pixel q of this set the cumulative cost is calculated by adding the cost value of pixel p from the cumulative cost map and the result of the static local cost function $l_S(p, q)$, assuming that on-the-fly training is disabled. Now we determine if q is already contained in the active nodes list by checking if $status\_map[q]$ equals 'Visited'. If this is true and additionally the cost value calculated in the last step is lower than the already stored cost of pixel q, then we have to remove q from L and set its status to 'NotProcessed' to be able to re-insert it at the updated position.

The final step of the while-loop is the check if $q\prime s$ status is 'NotProcessed', in this case we have to insert (re-insert if it was removed in the previous step) q into L at the proper position and to update the cumulative cost map as well as the map of minimum cost pointers.

One remark should be made about removing points to be able to properly re-insert them at an updated position. Due to the stack structure of the points in the active nodes list, it is not possible to remove points in constant time, because they have to be searched in the stack implementation, a doubly linked-list. Mortensen et al. propose a technique to guarantee point removal in constant time by storing for each pixel a pointer to the current location in the array of stacks. Our implementation doesn't include such a feature, because if we choose a high value for M, the stacks do not grow very large and the removal time is nearly constant.

An example for the algorithm is shown in Figure 4.10. It is a demonstration of the graph search algorithm creating a minimum cumulative cost map and a shortest path pointer map. Figure 4.10a is the initial local cost map with the seed point circled. For demonstration use the local costs are pixel based ($l(q)$) not link based ($l(p, q)$). The first algorithm step puts the seed point into the list of active nodes and sets its cumulative cost to zero. Now the costs to the seed points neighbours are calculated (the seed point is "expanded"), diagonal neighbours are additionally weighted with a factor of $\sqrt{2}$ to correct the influence of differing euclidean distances between diagonal and direct neighbours. It should be mentioned that the actual Live Wire algorithm performs this correction by multiplying direct neighbours with $\frac{1}{\sqrt{2}}$. Now all neighbours are put into the active nodes list and the point with the smallest cost is removed from this list, being the next expanded point. In our case this is the right neighbour of the seed point.

The expanding procedure is continued by always removing the minimum cost point from the list of active nodes. The active node lists L during a) and b) are also illustrated in the figure. The stacks of points consist of elements with a x- and a y-component. There are 2 situations between step b) and c) where points have to be removed from L. These points are (3,8) at cost 7 and (3,6) at cost 11. The grey arrows show their change of location in L after removing and re-inserting. This re-insertion corresponds to a newly found path in the cumulative cost map with a lower cost, it can be verified at the points North-East and South-East from the seed point.

Figure 4.10d shows the situation with 5 points expanded and Figure 4.10e shows the final cumulative cost map holding minimum total costs for each point to reach the seed point as well as the pointers specifying the optimal path to the seed point.

Manual and Automatic Path Cooling

For the final closed contour definition it is of course not possible to apply only a single starting point, because in this case start and end point are equal, so the shortest path would have a length of zero. So path cooling has to be used to provide path segments which in combination lead to a closed contour. In the *Intelligent Scissors* paper *Automatic Path Cooling* is presented. Path segments which remain stable during the mouse pointing procedure for a certain time are seen as candidates for new seed

points. Some of them are selected and the minimum cost path map is recalculated for a new seed point. The implementation in the segmentation tool does not include this feature, but simply selects new seed points manually after a mouse click with the left button. This click starts the recalculation of the minimum cost path and pushes the minimum cost path defined by the last seed point and the current seed point (the previous end point of the path) onto a stack of minimum cost paths. This stack is used for all other operations like e.g. drawing the current Live Wire contour.

As a consequence these algorithms and techniques provide minimum cost pointer maps making it possible to interactively select minimum cost paths to the seed point from it. As mentioned above Algorithm 6 is used for this selection process. The minimum cost paths now only have to be visualized and used for a contour definition of a crop region. Applying this technique on the end-diastolic or end-systolic images leads to a volume estimation.

4.5 Calculating the Volume

The final step of all segmentation-based volume estimation techniques is the calculation of the volume. Therefore the segmented pixels of all images are counted and multiplied by their voxel size. The segmentation comes from anyone of the algorithms presented in the previous sections. All of them have in common that they define a binary mask on the input images which distinguishes between ventricle and background pixels. In the field of radiology the technique of defining the counting result of voxels to get a volume is called *Simpson Rule*. The DICOM file format stores the voxel size by means of three entries:

- pixel spacing in x- and y-direction (0.83333 mm x 0.83333 mm)
- slice thickness in z-direction (8 mm)
- slice gap (4mm)

With this information voxel size is defined as $pixel\_spacing\_x * pixel\_spacing\_y * slice\_thickness$ leading to a result of 5.555555 mm^3. The special configuration of the Ultrafast CT with its slice gap of 4 mm between adjacent pairs of target rings (see Section 2.1) makes it necessary to interpolate three more volume slices into the existing eight for the final result. For this reason the counted pixels between second and third, between fourth and fifth and between sixth and seventh slice are averaged respectively. The averaged result is multiplied by the voxel size $pixel\_spacing\_x * pixel\_spacing\_y * slice\_gap$ (in our case: 2.777777 mm^3) and added to the other slices for the final volume. The whole algorithm is illustrated in Algorithm 7.

Algorithm 7 Calculate the volume of eight segmented images by means of Simpson Rule.

calculateVolume

1: initialize accumulation variable with zero
2: **for all** eight segmented binary images **do**
3: count the pixels representing object data (non-zero pixels)
4: multiply pixel count by voxel size
5: add multiplication result to accumulation variable
6: **end for**
7: **for** every second binary image **do**
8: interpolate a pixel count from the binary image and its neighbour image by averaging the existing pixel counts
9: multiply the interpolation result by the voxel size for the slice gaps
10: add multiplication result to accumulation variable
11: **end for**
12: report the accumulated result as calculated volume

4.6 Summary

In this chapter the implemented algorithms for this work were presented. The region-of-interest algorithm based on temporal information and the ellipse detection mechanism using Hough transform impose automatization to a certain extent. Three algorithms were implemented to perform the segmentation which is necessary to calculate volumes by the *Simpson-Rule*. These are an approximation of an optimal thresholding algorithm, an implementation of the basic deformable model called Snakes, and a graph-theoretic border tracing algorithm with the help of dynamic programming, the Live Wire approach. Finally an implementation of the *Simpson-Rule* to calculate volumes has been described .

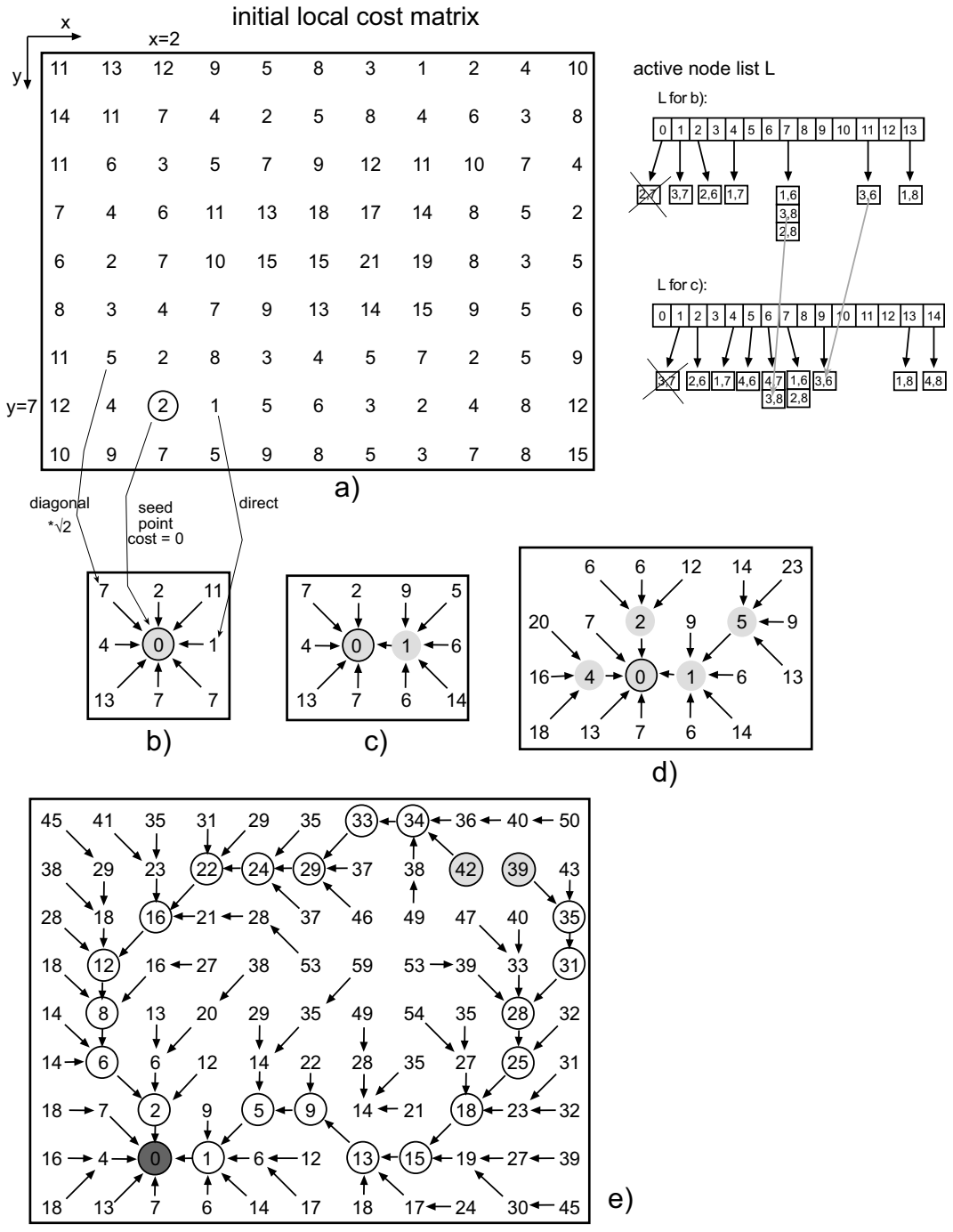

Figure 4.10: Live Wire costs and algorithm example. a) Initial local cost matrix, circle around seed point. b) Seed point (shaded) expanded, note the diagonal costs being multiplied with an euclidean correction factor. c) 2 points (shaded) expanded. d) 5 points (shaded) expanded. e) Finished cumulative cost map and shortest path map with the two light-grey shaded points as path end points and the dark-grey one as seed point. By pointing the mouse cursor on one of the end points the shown path indicated by circled points is reported as minimum cost path. Taken from [40].

Chapter 5

Implementation

To conveniently apply different sorts of image processing algorithms to images it is necessary to somehow set up a framework where all implemented modules are put together. In our case this application had to run on Windows NT to integrate into the existing network environment at the Department of Radiology. Further the application had to be capable of importing the medical image data sets which were specified in the DICOM file format. So Windows NT and Microsoft Visual C++ 6.0 were chosen as development environment.

Of course it is not advisable to start writing image processing applications from scratch, so three assisting libraries were used to support the implementation. These libraries were the *Intel Image Processing Library* and the *Open Computer Vision Library* for image processing functionality as well as the graphical user interface toolkit *Qt*. *Giplib*, the image processing library under development at our institute, was also taken into consideration for supporting the application, but computation time issues seemed to be more important than a clean object-oriented design and platform independence. Nevertheless the key algorithms implemented during this work will also be ported to the *Giplib*.

5.1 Environment & Libraries

5.1.1 IPL & OpenCV

Both, the *Image Processing Library (IPL)* and the *Open Computer Vision Library (OpenCV)*, are written in C++ and originate from Intel ([30],[14]). But these are already the only similarities because the OpenCV is an open-source high-level library for image processing and computer vision, while the IPL is only distributed in binary form and consists of low-level image processing routines. The most important feature of the IPL is its optimized library structure. The dynamic linked libraries of the IPL include a concept to detect the processor type at runtime and load the corresponding 'dll', which includes all of the libraries functionality in a processor architecture optimized version. The only disadvantage of this concept (but the biggest advantage for Intel) is that the IPL only works fast on Intel processors.

The concept of the OpenCV is different because it defines an interface for an open-source library. Nevertheless the only actual implementation for Windows is utilizing the IPL for low-level image processing tasks. It is still in a beta stadium, but there is a large group of users who contribute to the development with their bug reports and corrections. For our application IPL version 2.5 and OpenCV beta 1 were chosen.

The IPL consists of the following functionality blocks:

- Image creation and access. 1 bit, 8 bit signed and unsigned, 16 bit signed and unsigned, 32 bit signed and 32 bit float images are supported.
- Arithmetic and Logical Operations.
- Image Filtering. There are linear and non-linear filter techniques, pre-defined filter kernels and methods to specify convolution masks.
- Linear Image Transforms. Fast Fourier Transform and Discrete Sinus Transform.
- Morphological Operations.
- Color Space Conversions.
- Histogram and Thresholding functions as well as image statistics.
- Geometric Transforms.

The OpenCV among other things adds the following higher level functionality:

- Contour Processing.
- Geometric Fitting. Fitting ellipses, lines, rectangles, circles into images.
- Feature Detection. Filtering, Canny edge detection, Hough transform.
- Image Pyramids.
- Camera Calibration.
- View Morphing.
- Active Contours.
- Gesture Recognition.
- Drawing Primitives like ellipses, lines, circles, polygons.

5.1.2 Qt

For a convenient way to apply algorithms to images it is very useful to construct a graphical user interface combining both blocks. *Qt*, a product from Trolltech[61], is a suitable library to perform this task. Basically Qt is an open-source (only for non-commercial use) object-oriented C++ toolkit supporting many different platforms in the Unix and Windows worlds. A big advantage of Qt is the large number of classes and tools which are already included, making it useful in many different kinds of applications. The most prominent example for a Qt project is the K desktop environment KDE on Linux which uses Qt as its basis.
The following list summarizes the most important features of Qt:

- Object orientation and a modular concept in connection with a provided API consisting of many classes.

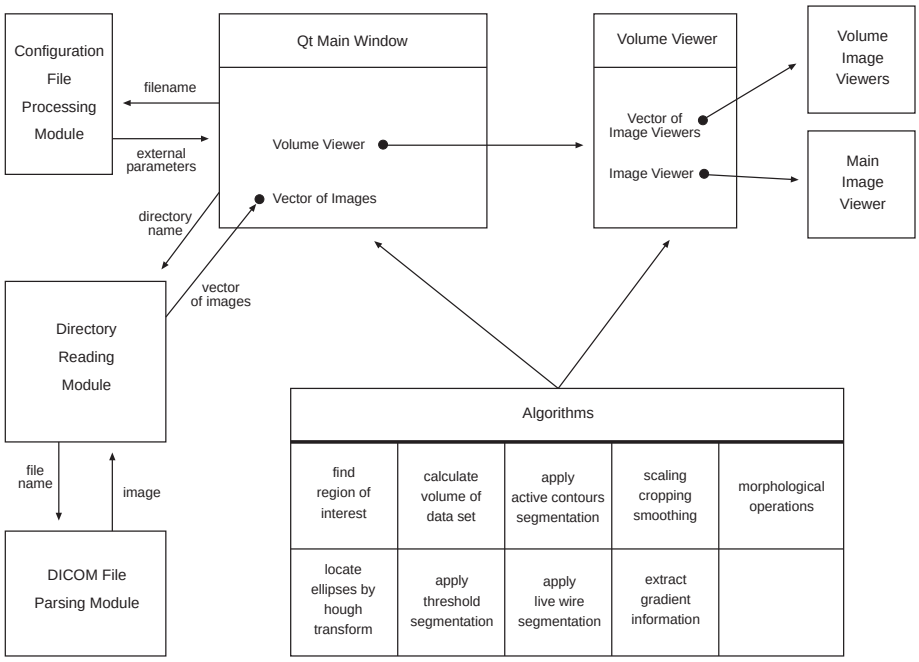

Figure 5.1: Block diagram of Segmentation Tool. The basic structure of the application is shown with its functional blocks for image parsing, viewing and processing. The central block is *Qt Main Window* which is responsible for constructing a GUI and bringing together all other modules.

- Easy customization of widgets to personal needs by inheritance.
- Portability.
- Intelligent and transparent component concept. Qt objects have the capability to easily pass messages to each other by the signal/slot concept incorporated in the library. It is possible for objects to define signals which can be emitted at any time. These signals can be connected to so-called slots of the same or another object where appropriate reactions are executed. This leads to a type- and object independent transparent communication method.

For our implementation version 2.3.0 of the Qt toolkit was used.

5.2 Segmentation Tool

Figure 5.1 shows the basic structure of the developed application called 'Segmentation Tool'. The central block of the application is the *Qt Main Window* which is responsible for displaying the graphical user interface and integrating image parsing, viewing and processing algorithms. A typical work flow could be like follows. After starting the application the main window is created. A screenshot of the main window is illustrated in Figure 5.2.

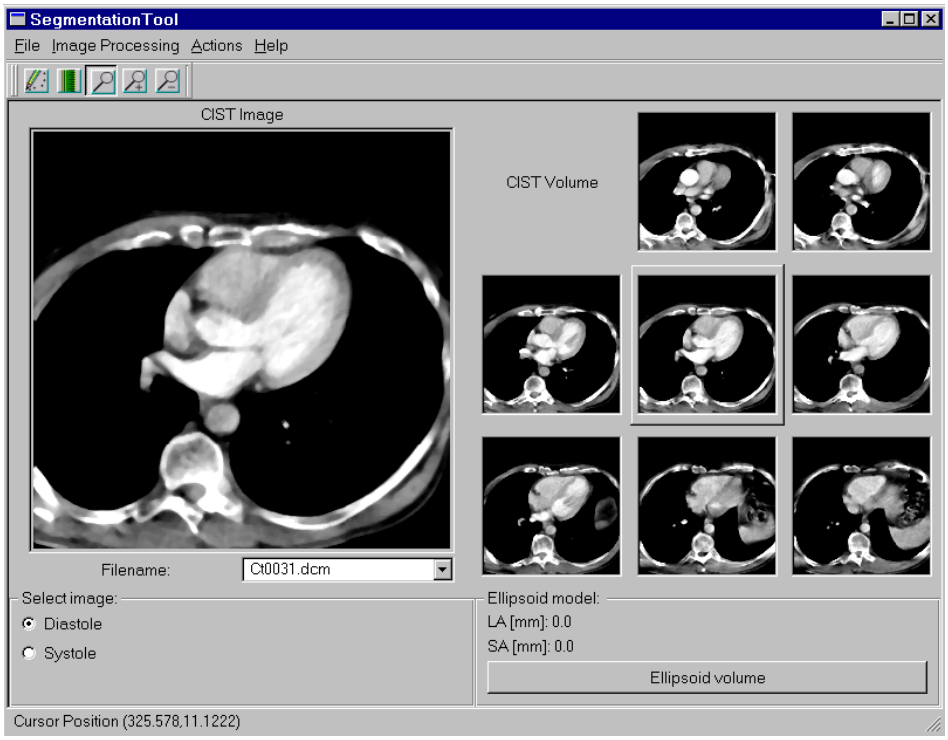

Figure 5.2: Segmentation Tool screenshot.

First of all a *configuration file* is *processed*. This configuration file contains a large number of parameters to easily customize many different sections of the application. Some examples for externally definable parameters are:

- MAIN_WINDOW_WIDTH & MAIN_WINDOW_HEIGHT
- DISPLAY_GRAYLEVELS
- HOUNSFIELD_LOW_LVL & HOUNSFIELD_HIGH_LEVEL (for definition of window-level transformations)
- SLICE_GAP_MM (the slice gap of CT image slices)
- SLICE_THICKNESS_MM (the slice thickness of CT image slices)
- PIXEL_WIDTH_MM & PIXEL_HEIGHT_MM (pixel width and height of CT image slices)

Some other parameters specify default values for the various algorithms or default directories for image parsing.

The next logical work flow step is to load image data from files into memory. Therefore the graphical user interface provides a file dialog to choose a directory name. This directory name is passed to the *Directory Reading Module* which extracts the corresponding filenames of the directories DICOM files and initializes an empty vector of images. Each filename is transmitted to the *DICOM File Parsing Module* which parses the DICOM file and returns an image to the *Directory Reading*

Module. There the images of a data set are gathered in the image vector data structure. After all images have been read, the image vector is returned to the *Qt Main Window* where the images wait for further processing steps.

At this point a more detailed explanation of the *DICOM File Parsing Module* should be given. The decision whether to implement a simple parser from scratch or to use an existing library was influenced by the fact that libraries like **ImageMagick** or tools like **IrfanView** do not support DICOM files with a grey-level resolution of more than 8 bit. So it was necessary to implement a module supporting the actual medical images with their grey-level resolution of 12 bit. This module is only able to read a very small portion of the DICOM image file format, it is restricted to reading images with arbitrary width and height but only one channel of unsigned 16 bit image data, where 12 bit are used for storing pixel values and bit 11 is the highest bit. The two bytes forming a 16 bit pixel can be stored in little or big endian format. The image is stored into a 16 bit unsigned **IplImage** data-structure (IPL_DEPTH_16U) as memory representation and is passed to the *Directory Reading Module*.

After we have loaded image data from files into main memory applying preprocessing algorithms like detecting a region of interest or detecting a large ellipse for selecting an end-diastolic image becomes possible. Additionally the images are displayed by means of the *VolumeViewer* and *ImageViewer* objects. The volume viewer is basically a container object consisting of a main image viewer and 8 image viewers displaying the current volume defined by the image of the main image viewer. The *ImageViewer* class defines a Qt widget with the capability of displaying an **IplImage** and additionally supporting many image manipulation techniques like zooming, scrolling, setting regions of interest of arbitrary form or measuring distances in image space.

Finally the *Algorithm* module provides all image processing algorithms to be applied on the image data. These algorithms range from basic preprocessing techniques like scaling, cropping, smoothing, gradient calculation or morphological operations over automatization methods like the Hough transform for ellipse detection and the region of interest detection to the different image segmentation techniques. We reused some low-level algorithms from the *Intel Image Processing Library*, while most of the high-level algorithms do consist of implementations which were especially performed for this work. Preprocessing algorithms operate in the main window section while all others are utilized as part of the volume viewer module.

Chapter 6

Experiments & Results

For the statistical evaluation of the different algorithms 31 medical image data sets were provided by the *Department of Radiology, University Hospital Graz*. Each data set consists of 80 images. Furthermore *Prof. Rienmüller* drew end-diastolic and end-systolic contours in the images by means of the implemented segmentation tool application. Results of the parametric estimation method are also required for the evaluation, therefore the end-diastolic and end-systolic volume estimation from the patient's diagnosis (performed by an operator after the CT scanning procedure) were provided. To show that different medical operators (and even the same operator at different points of time) tend to produce slightly varying estimation results, the parametric model was applied a second time on each data set by Prof. Rienmüller. The differing results are of course unintended but nevertheless nearly unavoidable. Different levels of concentration depending on the number of already processed data sets or the tiredness of the operator during the work is one possible explanation. Another one is the experience of the operator with radiological and (in our case) cardiological aspects but also differing interpretations of certain medical aspects can lead to varying results. Figure 6.1 illustrates the correlation of these two estimations. Especially noticeable is the really poor correlation of the end-systolic volumes with $r = 0.64$. In addition to the reasons mentioned above, the difficulties in defining the end-systolic left ventricle shape lead to further inaccuracies.

Table 6.1 shows all of the provided information of the image data sets. All of the examinations were done in the years 2000 and 2001. Columns "ED slice" and "ES slice" show the numbers of the image which were chosen as end-diastolic and end-systolic image respectively. Columns "EDV1" and "ESV1" contain the results of the parametric volume estimation while columns "EDV2" and "ESV2" contain the older results of the parametric model from the actual patient examination. The data sets 23 and 24 as well as 9 and 10 are of special interest. Both pairs of sets are taken from the same person within a period of approximately one year.

For all subsequent evaluations "EDV1" and "ESV1" will be taken as reference values, because "EDV2" and "ESV2" only contain 30 data sets (the examination of data set 24 is not available).

6.1 Automatic Selection of End-Diastolic and End-Systolic Images

6.1.1 Evaluation of the Region of Interest Algorithm

The algorithm for automatic ROI extraction of Section 4.1.3 is applied to all data sets. Verification is performed visually, i.e. by inspecting the region of interest and deciding if it is large enough to include the whole left ventricle on all images of the data set. Further assessment of the ROI size (it

Nr	ED slice	ES slice	EDV1 [ml]	ESV1 [ml]	EDV2 [ml]	ESV2 [ml]	Nr	ED slice	ES slice	EDV1 [ml]	ESV1 [ml]	EDV2 [ml]	ESV2 [ml]
1	41	46	69.6	18.9	82	20	17	21	26	63.2	7.9	74	14
2	22	28	69.2	30.7	74	32	18	51	57	86.5	17	96	21
3	31	38	82.3	12.6	82	8	19	41	45	54.4	28	65	42
4	42	47	52.7	10.2	84	20	20	31	37	85.4	15.4	99	28
5	41	44	51.7	6.1	51	22	21	42	46	85.7	21.1	78	20
6	41	46	56.7	16.4	62	23	22	41	46	123.4	27.4	135	39
7	51	55	58.9	4.8	76	14	23	51	55	68.5	10.8	76	9
8	41	46	80.7	23.8	97	26	24	51	57	104.1	24.6	N.A.	N.A.
9	31	37	81.4	25.1	94	27	25	41	45	54.7	28.6	48	21
10	41	47	76.8	33.3	81	26	26	51	55	68	5.7	66	16
11	41	46	65.3	21.2	76	29	27	32	35	71.3	12	74	45
12	41	45	63.1	10.5	74	10	28	41	46	97.7	22	99	28
13	31	36	114.9	42.3	87	40	29	41	46	98	18.6	70	25
14	31	35	70.2	20.1	77	26	30	31	36	87.7	18	78	18
15	31	37	99.9	13	105	21	31	31	36	52.1	19.1	60	18
16	51	55	40.8	5.4	41	7							

Table 6.1: Patient data sets with volume estimation results from the parametric model. Column ”ED slice” shows the number of the image which was chosen as end-diastolic image by Prof. Rienmüller during the evaluation session. Column ”ES slice” shows the number of the end-systolic image. Images in the data set are numerated from 1 to 80. Columns ”EDV1” and ”ESV1” contain the results of the parametric volume estimation performed by Prof. Rienmüller while columns ”EDV2” and ”ESV2” contain the older results of the parametric model from the actual patient examination.

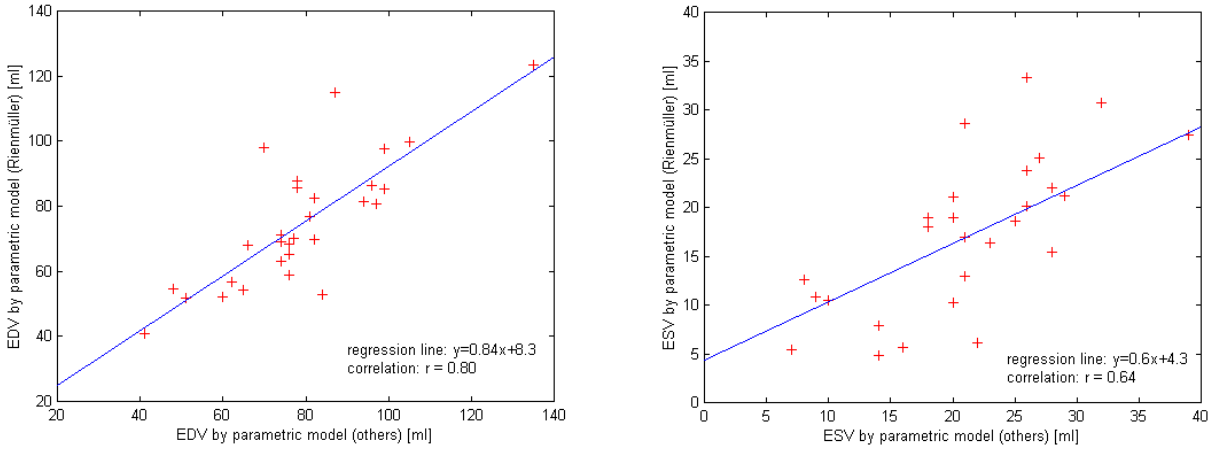

Figure 6.1: Evaluation of two different manual parametric volume estimations. End-diastolic and end-systolic volumes of 30 data sets. The correlation is not as high as expected, this shows the uncertainty inherent in the parameter estimation process.

could be too large, including many unwanted details which interfere with the following algorithms) is not performed due to lack of a proper measurement method.

The algorithm applies a window-level transformation (window: 300 HU, level: 50 HU) to the raw images. Afterwards each image is processed with a noise threshold of 0.015 after the *"Subtract temporally adjacent images"* stage. This value was chosen empirically taking into account that the images are converted to floating point images and scaled between 0.0 and 1.0 before subtraction. The corresponding key for this threshold in the configuration file is GET_ROI_NOISE_THRESHOLD. The second noise threshold, applied to result image R (the image after adding all thresholded subtraction results), is chosen by extracting the maximum grey-value of R and taking 12% of it. Finally the coordinate-axis parallel bounding rectangle is corrected by an empiric translation with respect to the positive x-axis. Testing the algorithm showed that the extracted ROI is always too large and too far to the left in the images. The translation coefficients in pixel are derived from the region of interest extents. Suppose the ROI rectangle lying between $roi\_min = (x1, y1)$ and $roi\_max = (x2, y2)$. Then the translated rectangle lies at

$$\begin{aligned} roi\_min' &= (x1', y1) \\ x1' &= x1 + 0.15 * roi\_width + 30 \\ roi\_max' &= (x2', y2) \\ x2' &= x2 + 0.15 * roi\_width + 50 \end{aligned}$$

The presented parameters remained unchanged during evaluation of all 31 data sets.

Given these parameters 30 from 31 regions of interest were extracted in a visually correct way. Only data set 21 was processed wrong. Three data sets (6, 7 and 18) produced very large ROI's which are not optimal for later processing steps. Figure 6.2 shows the ROI rectangles at the end-diastolic points of time, i.e. when the left ventricle is largest, for all data sets. Summing up this algorithm seems to be a very useful preprocessing step for many possible kinds of higher-level algorithms, which require a reduction of information. Therefore it will be used in later evaluations, especially in the automatization technique for finding the end-diastolic image by use of an Hough transform.

6.1.2 Evaluation of the Ellipse Detection Algorithm

For automatically locating an end-diastolic image (i.e. that image of a data set with the largest projected left ventricle) it is useful to look for ellipses as described in Section 4.1.4. Applying this algorithm to all 31 data sets incorporates a tradeoff between time consumption and optimal results. The more general the necessary parameters are chosen the worse are the results. But it is not possible for a practical algorithm to tune parameters before working on a data set, this would simply be too costly with respect to its time-consumption. For this reason we have decided to utilize a fixed set of parameters for the algorithm while evaluating all data sets.

First of all a region of interest is extracted from the images to lower the complexity of the ellipse detection problem. Otherwise the algorithm would only randomly find the correct ellipse in the images because there are many elliptic structures in them. The algorithm gets passed the following parameters:

- minimum width of the ellipse: 20 pixel
- maximum width of the ellipse: 70 pixel
- width discretization: 70

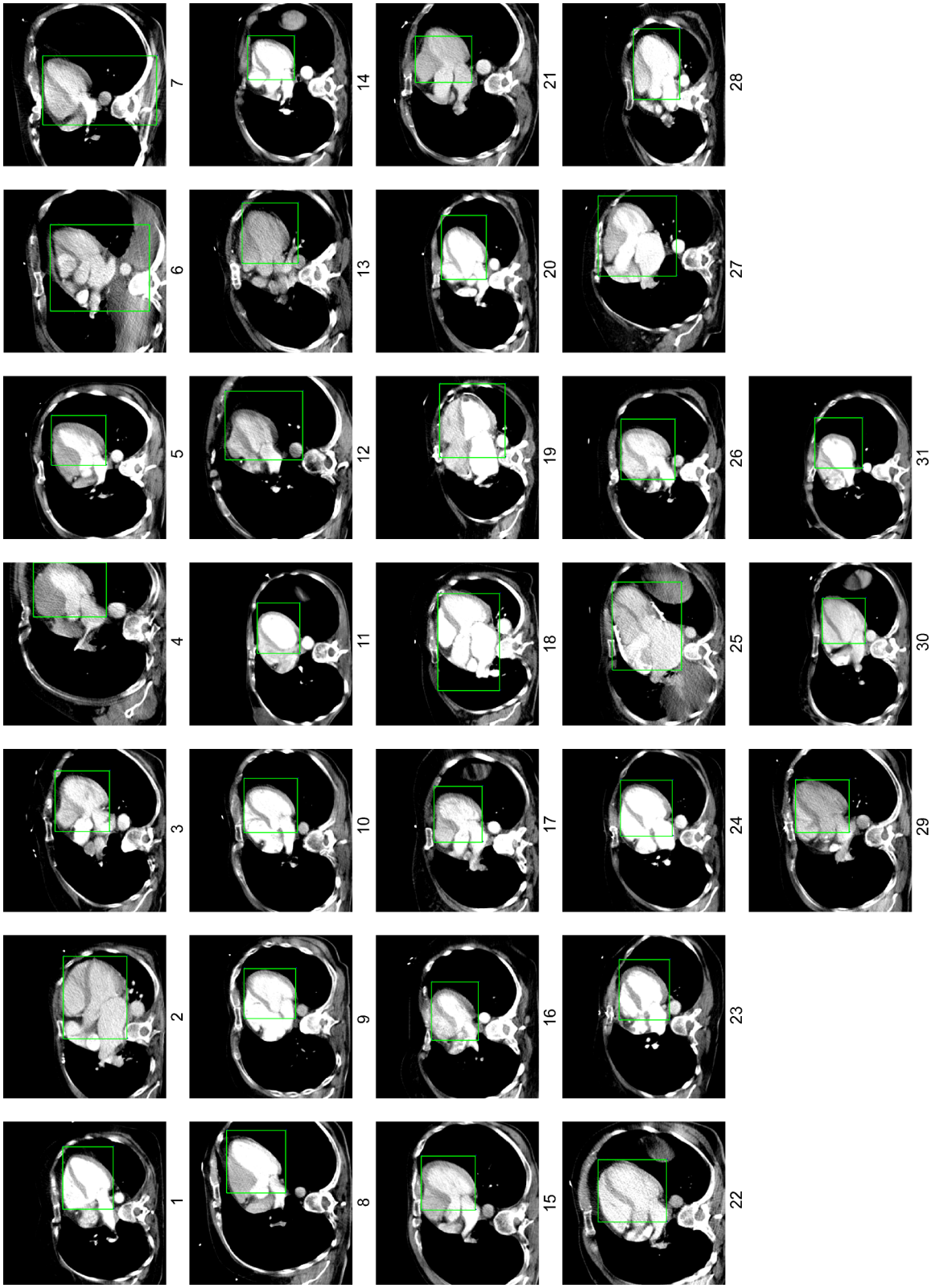

Figure 6.2: Regions of Interest result. All 31 data sets are represented by their end-diastolic image. It can be seen that 30 ROI's are correct (only data set 21 is too small) and from these only three ROI's are too large (6, 7 and 18).

- minimum height of the ellipse: 10 pixel
- maximum height of the ellipse: 45 pixel
- height discretization: 50
- minimum rotation angle of the ellipse: $\frac{9.5\pi}{18} rad$
- maximum rotation angle of the ellipse: $\frac{17\pi}{18} rad$
- rotation angle discretization: 30

Then a window-level transformation (window: 300 HU, level: 50 HU) and a 7x7 median filter performs the necessary preprocessing. After scaling the threshold value 3500 separates iodinated blood regions from the other tissues (the possible range of grey-values is 0 to 4095). This value was defined by empirically testing many data sets. The thresholded image is filtered with a 3x3 sobel kernel leading to a gradient magnitude image on which the Hough transform algorithm is applied. During the Hough transform stage there are two sections where a common Hough transform to detect lines is performed. On the one hand to detect horizontal axis candidates and on the other hand for vertical axis candidates. Both stages need a certain set of parameters. For the horizontal axis candidates we look for at most 12 lines consisting of at least 25 pixels (i.e. entries in the Hough accumulation array). The possible angle of the lines lies between $\frac{0.5\pi}{18}$ and $\frac{8\pi}{18} rad$ with respect to the x-axis. At most 10 vertical axis candidate lines consisting of at least 12 pixels are looked for in the other stage, the possible range of angles is $\frac{3\pi}{18}$ to $\frac{\pi}{2} rad$. Finally entries in the 3D accumulation array of the algorithm's last step have to be larger than a noise threshold of 3 preventing pixels, which are randomly grouped so that the algorithm mistakes them for an ellipse, from being detected.

From the 31 possible data sets only 28 were taken for the evaluation. Data sets 6,7 and 18 yield to large regions of interest according to Section 4.1.3, as a consequence the Hough transform algorithm would fail or just randomly detect a correct ellipse. The results of the evaluation are illustrated in Table 6.2. These results show the rather poor performance of the algorithm because from the 28 data sets only 8 data sets were processed correctly (marked with a '+'), in 4 data sets too large ellipses were detected ('~') while 16 data sets lead to incorrect outcomes ('-'). The subjective rating indicated with '+', '~' or '-' is the author's opinion about the correctness of the located ellipse. The rating corresponds in most cases to either temporally or spatially adjacent images of the data sets. (E.g. images 31 and 41 are spatially adjacent, while images 41 and 42 are temporally adjacent.)
Figure 6.3 illustrates different data sets with good, moderate and bad results.

There are many reasons for the poor performance of the ellipse detection algorithm on the data sets. The most important one should be noted first, the selected Hough transform algorithm is not robust enough for this kind of data! There are too many parameters to tune in the algorithm for producing a proper result and a set of tuned parameters is very likely to be useless in another data set. Furthermore the algorithm's performance decreases if there is too much additional information in the image beside the ellipse to be located. The two internal Hough transform stages, where symmetry axis candidates are tried to be detected are very susceptible to smooth and continuous ellipse edges. But the ellipse edges in our images are noisy and often have large gaps inbetween.

Another effect is the strong dependency on an accurate region of interest, not too small so that no ellipse is cut off and not too large to minimize unnecessary information. As a consequence the unavoidable inaccuracies of the region of interest algorithm have a strong impact on ellipse detection. Another disadvantage is the lack of edge direction information incorporated in the algorithm which leads to ellipses that span over non-blood regions (compare Figure 6.3). Quite important as well is

Nr	ED slice manual	ED slice Hough	perfor-mance	Nr	ED slice manual	ED slice Hough	perfor-mance
1	41	50	~	17	21	34	-
2	22	3	-	19	41	63	-
3	31	46	-	20	31	37	-
4	42	41	+	21	42	55	~
5	41	46	-	22	41	31	+
8	41	31	+	23	51	60	~
9	31	14	-	24	51	59	-
10	41	46	-	25	41	10	-
11	41	42	+	26	51	59	-
12	41	61	+	27	32	24	-
13	31	54	~	28	41	71	-
14	31	32	+	29	41	5	-
15	31	54	-	30	31	46	-
16	51	51	+	31	31	42	+

Table 6.2: Patient data sets with ellipse detection results. The column 'ED slice manual' shows the reference image while 'ED slice Hough' is the end-diastolic image detected during the Hough transform algorithm. Further a subjective rating of the correctness is given for each data set with the following possible marks: '+','~','-'.

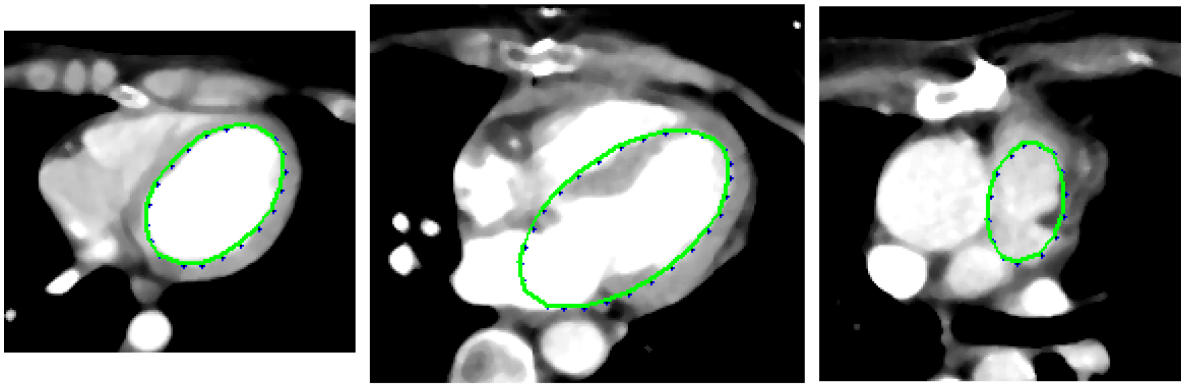

Figure 6.3: Some results of the ellipse detection algorithm. The left image shows data set 31 with an optimally found left ventricle chamber. The image in the middle is from data set 23 and shows the located ellipse (green) compared with the real ventricle contour (red). The right image (data set 29) illustrates the failing of the algorithm.

the fact that some end-diastolic projections of the left ventricle chamber do not have an elliptic shape making it impossible to find a correct result. Finally the algorithm's assumption that the ellipse with the largest area over all images of a data set has to be the left ventricle is not always valid.

6.2 Statistical Evaluation of Left-Ventricle Volumes

For all of the following evaluations a certain schema will be used. Each comparison of two models will be presented on a two-dimensional chart with the result vector of one model on the x-axis and the result vector of the other model on the y-axis. So we get a scatter plot of corresponding evaluation values. After fitting a straight line into this scatter plot in a least-squares manner we receive a so-called regression line y=k*x+d. As can easily be seen it would be optimal that k=1 and d=0. But this statistical measure is not sufficient, it is further necessary to calculate the cross-correlation of the two result vectors to statistically interpret the relation of the two vectors. The definition of the cross-correlation given two vectors $(x_1, ..., x_n)$,$(y_1, ..., y_n)$ and its mean values x', y' is:

$$r = \frac{\sum_{i=1}^{n}(x_i - x')(y_i - y')}{\sqrt{\sum_{i=1}^{n}(x_i - x')^2 \sum_{i=1}^{n}(y_i - y')^2}}$$

The regression line shows stochastic dependencies of the data vectors, making it possible to derive conversion rules between different models. The correlation is a measure of how similar the two data vectors are, the values for r lie between -1 and 1 representing optimal positive and negative correlation. The higher the correlation the more valid is a possibly derived conversion rule.

6.2.1 Volume via Manually Drawn Contours

The first evaluation being presented is the most simple one. Prof. Rienmüller's contours which were drawn into the images during the parametric model calculation sessions are used to define cropping regions in the images. After cropping the volumes are calculated. The results show good correlation with the parametric model (compare Figure 6.4). This result should not be overestimated due to the fact that Prof. Rienmüller drew the contours in the same sessions when the two-axis method for volume estimation was performed. So he had exactly the same kind of ventricle shape in mind during the session. This is theoretically optimal but it doesn't represent medical routine evaluations where different operators are working with the data at different points of time.

6.2.2 Volume via Thresholding

The segmentation tool embeds the thresholding algorithm in an environment which is able to automatically work on data sets. Therefore it loads data sets from a specified base directory and processes them one by one writing volume estimation results into an output file. First of all a window-level transformation (window: 300 HU, level: 50 HU) is applied to the images of each data set. Afterwards the images are median filtered (5x5 kernel) and the region of interest is extracted according to Sections 4.1.3 and 6.1.1. From the manual inspection of the image data sets it is known which images of the data sets contribute to the end-diastolic and end-systolic volume. This was specified during the manual drawing of the ventricle contours. This information is utilized to improve the performance of the thresholding segmentation by discarding those images which are not contributing to the volume. There is a parameter file from which this data is read during the automatic evaluation step. On the regions of interest of the remaining images the optimal-threshold-approximation algorithm is applied

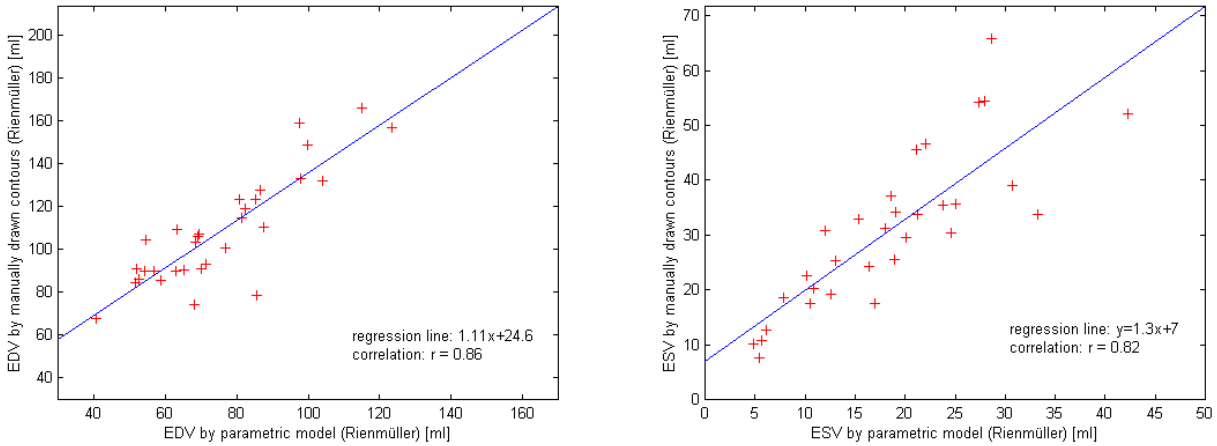

Figure 6.4: Evaluation of volume by means of manually drawn contours. End-diastolic and end-systolic images are selected in each of the 31 data sets. The volume is defined by manually drawing the contours of the left ventricle into the images and cropping the regions inside these contours.

(see Algorithm 3). The volume results are written to an output file and combined to form the evaluation shown in Figure 6.5. From this chart it can easily be verified that the performance of the threshold is very poor due to the reasons mentioned in Section 4.2.1. The main reason for the bad performance is that most image histograms are not tri-modal as assumed due to noise and aliasing artifacts.

Examples for a successful segmentation result and a very poor one are given in Figure 6.6. The upper row shows the algorithm's performance when the assumed tri-modal histogram really occurs. The lower row illustrates the failing of the algorithm due to a bi-modal histogram.

6.2.3 Volume via Active Contours

For the evaluation of the Snakes algorithm some parameters have to be set. Although the result of the evaluation would be better by tuning these parameters they remained fixed during the processing of each data set. On the one hand this is necessary for a meaningful evaluation and on the other hand the time required for processing a data set is an important factor. Similar to other evaluations in this work, the first step is applying a window-level transformation (window: 300 HU, level: 50 HU) to the images of each data set. This is followed by a 5x5 median filter to remove noise. Now the actual Active Contours algorithm is performed by placing an initial contour in each image and starting the iterative energy optimization.

To segment the images first of all the information which images of a data set are contributing to end-diastolic and end-systolic volumes is used. Only those images are presented to the operator. The operator who performed the Active Contours segmentation is the author of this work. This is problematic to a certain extent, due to the author's lack of radiological experience. Therefore every time the author was not sure about an initial contour location, the manually drawn contours from Prof. Rienmüller were taken as a reference for the initial location. Due to the fact that the exact segmentation is performed by the Active Contour algorithm and the operator only has to delineate the approximate location of the left ventricle, this assumption is appropriate for most cases.

The graphical user interface (GUI) of the segmentation tool first of all requires the operator to draw

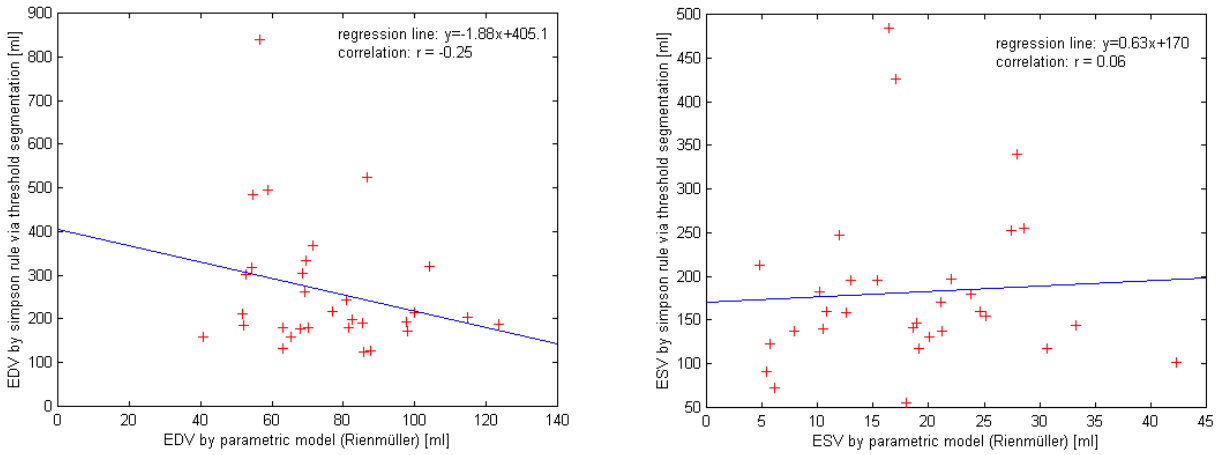

Figure 6.5: Evaluation of volume by means of optimal threshold approximation. End-diastolic and end-systolic results of the threshold algorithm applied to all 31 data sets compared with the results from the parametric model. The performance is very poor due to the reasons mentioned in Section 4.2.1.

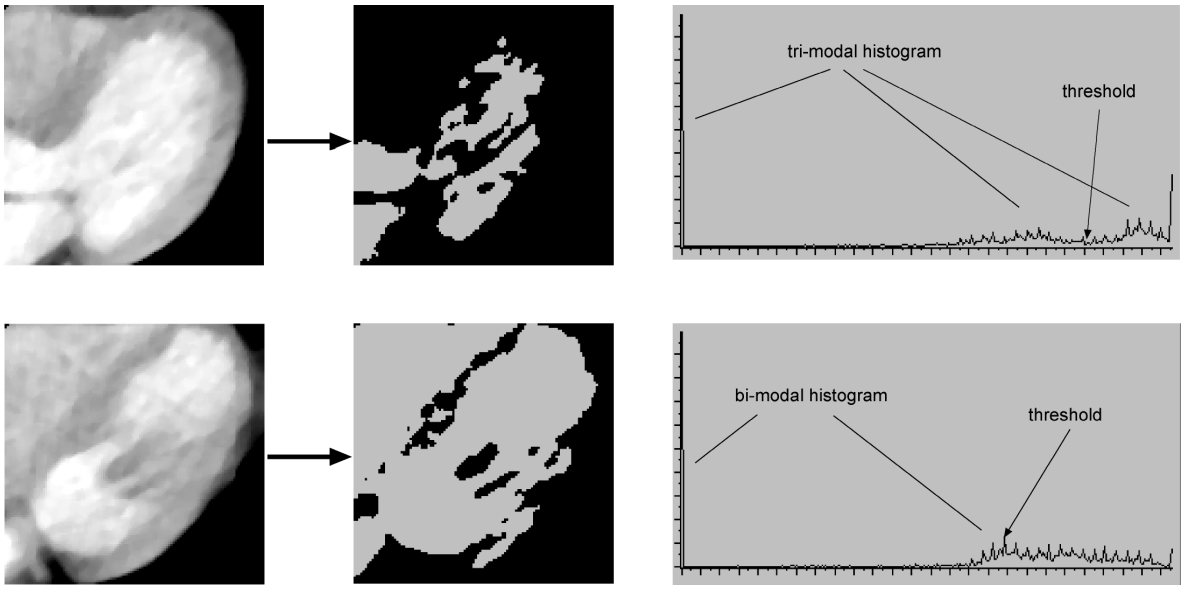

Figure 6.6: Examples for threshold segmentation. The upper row shows a region of interest before and after applying the optimal threshold approximation algorithm. The histogram of the original image is tri-modal, therefore the algorithm is quite successful. The lower row shows a ROI where the corresponding histogram is bi-modal, leading to the failure of the algorithm.

an initial contour into the images. This is shown in Figure 6.7. Each drawn point consists of its two integer-valued coordinates with respect to the local coordinate system of the underlying image and of some Snake parameters (ususally three to four parameters). Every time a contour point is placed in the image, the currently valid Snake parameters, which can be modified by some editable text fields of the GUI (compare Figure 6.7), are chosen. A further possibility is to fix a point by choosing zero for all Snake parameters. These two features remained unused during the evaluation. No fixed points were used, because no radiological expert was available to define the fixed locations, so the really difficult distinction between merging ventricle chambers is performed by the Snakes algorithm. Nevertheless this feature is important in an actual implementation for medical use. Furthermore the Snake parameters alpha, beta and gamma remained unchanged for all images of all data sets. This instance makes the results of the evaluation more comparable, nevertheless by fine-tuning the parameters the result of each evaluation could be enhanced. The Snake parameter values chosen for the evaluation are:

- alpha = 0.8
- beta = 1
- gamma = 1.3

This resembles that the gradient energy (gamma) of the underlying image is the strongest factor influencing the Snake iteration process. The bending term (alpha) of the internal Snake energy is the weakest term, due to the fact that it is not very important to converge to a contour with equally-spaced points. Nevertheless it is quite important to define an initial contour which is equally spaced and consists of a large number of points. For this purpose the segmentation tool application calculates additional contour points after the drawing of the initial contour is finished. If two initial points have a distance larger than a certain threshold value (specified by the configuration file key SNAKE-_DENSITY_THRESHOLD), the line between these points is subsequently halved until the distance is smaller than the threshold. The value of the density threshold used during the evaluation was 5 pixels. The effect of this feature is shown in Figure 6.8, which also illustrates a run of the Snakes algorithm on the specified contour in the right image. Its contour is constructed from the contour of Figure 6.7 by adding some more points and closing it.

Another unused feature of the Active Contours implementation is the curvature maximum determination which allows sharp contour corners to form more easily. This feature is not important in our case, because the left ventricle border is a smooth curve without sharp corners.

Now we will discuss the other parameters of the Active Contours algorithm. For calculating the gradient image a 3x3 Sobel operator is used. The two filter masks for x- and y-derivatives are convoluted with the image resulting in two floating-point gradient images S_x and S_y. They are combined to form the floating-point gradient magnitude image $S = \sqrt{S_x^2 + S_y^2}$. Finally this image is converted to a 16-bit unsigned integer image with values between 0 and 65535. This image is the basis for the Active Contours algorithm.

The size of the local neighbourhood (configuration file key SNAKE_NEIGHBOURHOOD_SIZE) used for searching local maxima was 7 pixels. The two termination criteria of the algorithm, minimum number of moved points during an iteration step (configuration file key SNAKE_MIN_NR_POINTS-_MOVED) and maximum number of iterations (configuration file key SNAKE_NR_ITERATIONS) per algorithm run were set to 2 points and 20 iterations respectively.

Two types of evaluation have been performed. First of all the results of the volume estimations by means of Snakes segmentation are compared with the volume estimations from the parametric model

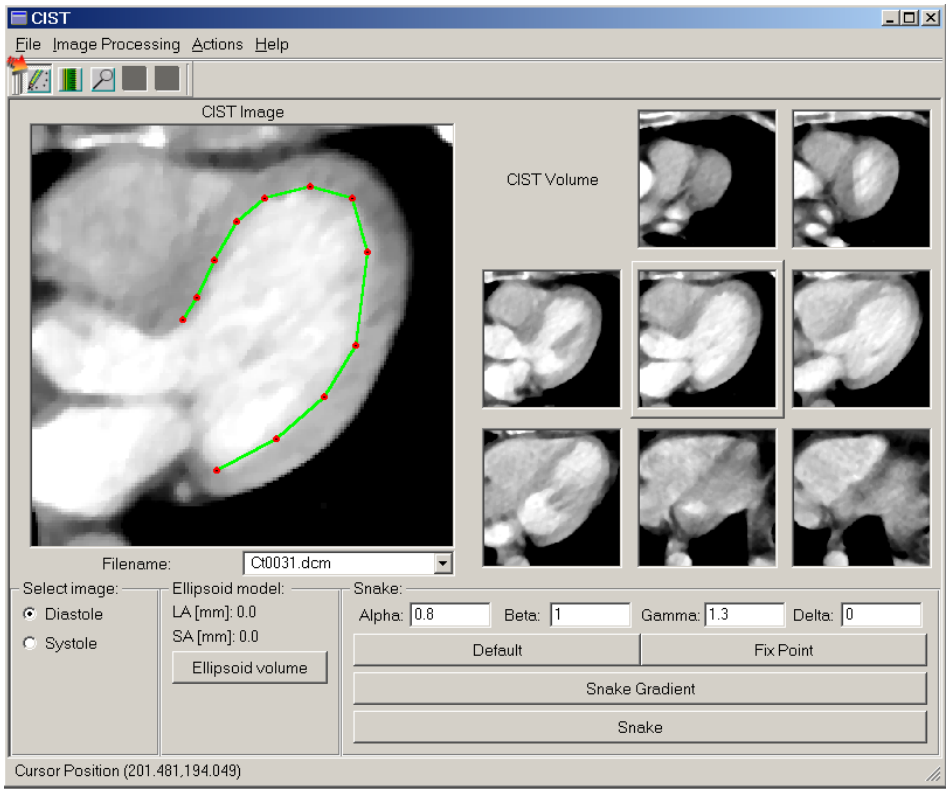

Figure 6.7: Example for drawing an initial contour. The main window shows the manually specified contour during drawing, note the pressed button in the top left corner of the window, which indicates the so-called Point-ROI mode. It is used too specify a region of interest by a contour. The bottom right section of the window holds the parameter boxes and buttons used for the Snakes algorithm.

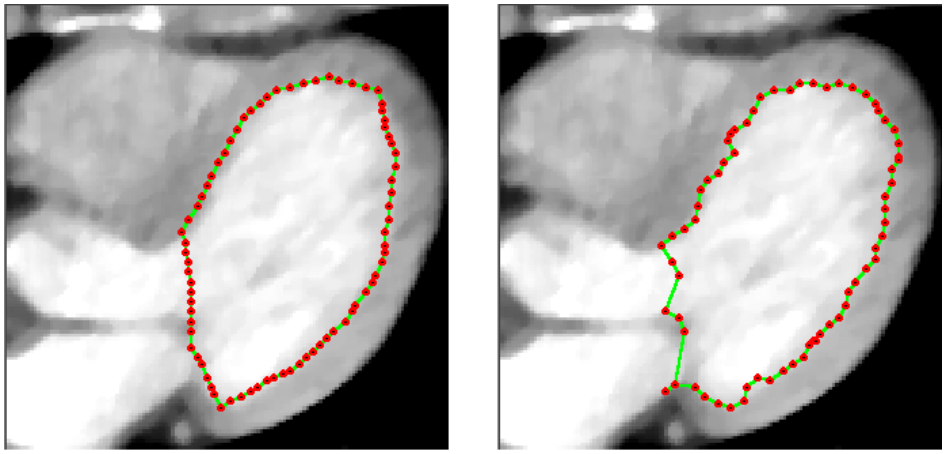

Figure 6.8: The left figure shows an initial contour drawn by an operator to provide a contour for the Snake algorithm. Please compare the contour of Figure 6.7 and note the automatically inserted contour points to get a more flexible initial contour. The right figure shows the same contour after performing the iterative Snake algorithm. The contour snapped onto the strong gradients of the image.

(see Figure 6.9) and with the volumes from the manually drawn contours (see Figure 6.10). The manually drawn contours and the Snake contours are excellently correlated. We found a correlation coefficient of 93% in the end-diastolic and 92% in the end-systolic case. This coefficient is higher than the actual results show (the evaluation of the measure for the accuracy of the contour overlap is presented in Figure 6.11), so we can conclude that we don't have enough data sets for a meaningful comparison of the Active Contour results. The many problems of the model are not correctly reflected in this high correlation. The parametric model shows a weaker correlation, there are more outliers and the regression line is farther away from the optimal 45-degree line. Nevertheless the result is quite similar to the result of the manually drawn contours compared with the parametric model (see Figure 6.4). This fact allows the conclusion that there is a connection between the two models which could be described by a formula.

The second evaluation concerns the accuracy of the contours found by comparing them with the manually drawn contours. This is achieved by comparing how much two corresponding contours (one from the Active Contour algorithm and one from the manual contour drawing) overlap. Therefore the interior pixels of both contours are regarded as pixel sets. The union of the two sets gives the area of both contour regions, the intersection of the two sets gives the region where both contours overlap. If we simply take the area of the overlapping region as a measure for the contour accuracy, we would depend on the size of the contour regions. To remove this effect the area of the the two pixel sets' intersection are related with the total area under both contours. In detail the performance measure looks like this:

$$contour\_performance = 1 - \frac{area(S_{man} \cup S_{snake}) - area(S_{man} \cap S_{snake})}{area(S_{man} \cup S_{snake})}$$

with $area()$ being a function to measure the area of a region by counting pixels, S_{man} and S_{snake} being the sets of pixels forming the interior of the manual and the Snake contour respectively and $\cup$ as well as $\cap$ being the logical operators union and intersection with sets of pixels as input. Figure 6.11 illustrates this performance measure. High $contour\_performance$ values indicate a better performance, the range of the values lies between zero and one.

The segmentation tool provides a testbed for calculating this performance value. Again, results are written to an output file for further investigation. Figure 6.12 shows some representative results. On the charts the "Slice Number" axis represents the 80 slices of a data set. A '+' indicates those slices which contribute to the end-diastolic image set (e.g. slices 11, 21, 31, 41, 51, 61 and 71 for data set 31) and a '*' indicates the slices of the end-systolic image set. On the "Contour Performance" axis the performance values from the comparison of Snake- and manually drawn contours are presented. Values near to one represent a better overlapping of the two contours.

The performance values have a tendency to be better on the inner slices than near the edge of the image set. The reason is that the outer slices show the left ventricle near its outer edges, where partial volume effects occur. The averaging of the pixel values in these regions due to partial volume effects, results in blurred edges (see Figure 6.13a), so all segmentation models depending on edge information are destined to produce inaccurate results. Further reasons for insufficient contour overlap are:

- Strong gradients in the vicinity of the initial Snake contour from the heart/lung border attract the contour (see Figure 6.13b).

- Some data sets do only have a weak contrast between blood and heart tissue (see Figure 6.13c) due to partial volume effects.

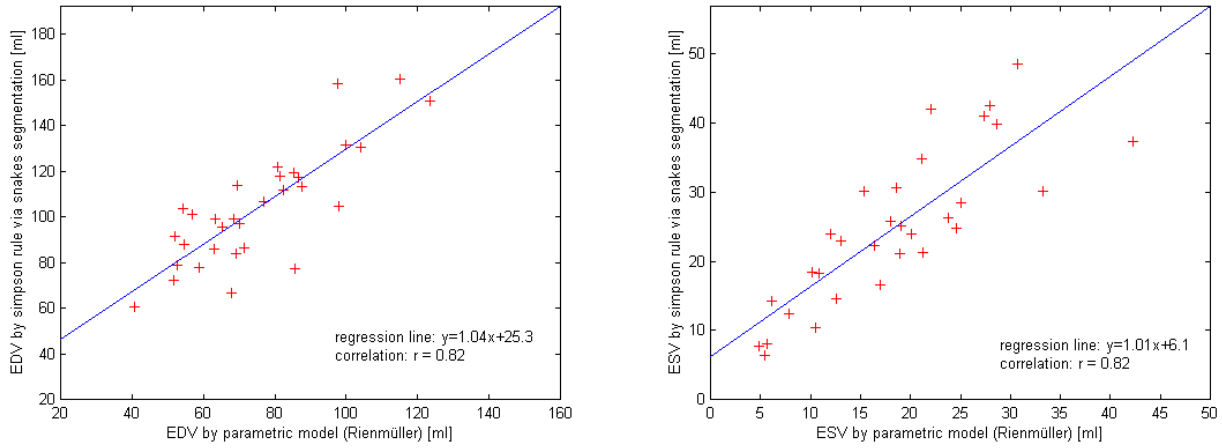

Figure 6.9: Evaluation of volumes by means of Active Contour (Snake) segmentation and parametric model. End-diastolic and end-systolic results of the Snake algorithm applied to all 31 data sets compared with the results from the parametric model. The performance of 0.82 is quite similar to the result of the manually drawn contours.

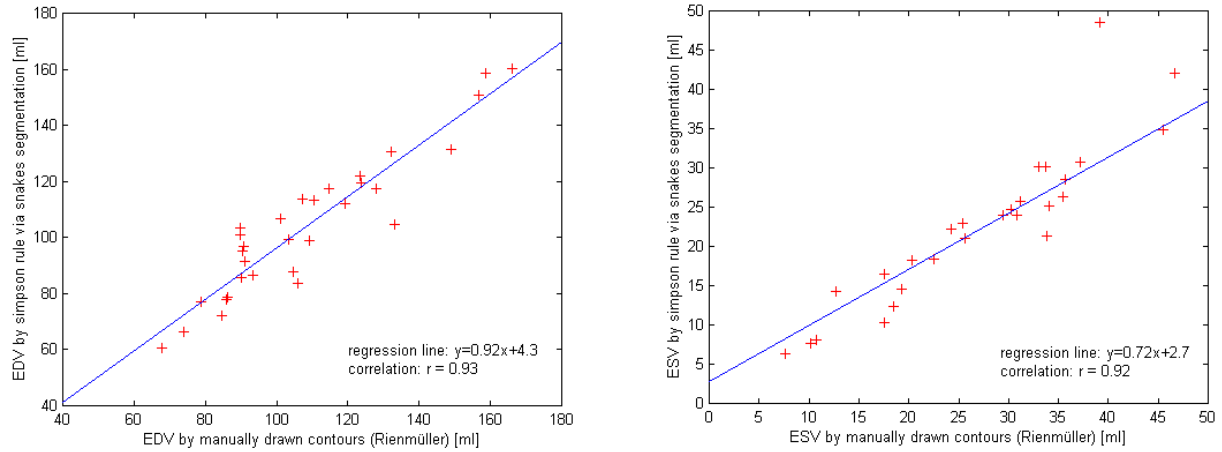

Figure 6.10: Evaluation of volumes by means of Active Contour (Snake) segmentation. End-diastolic and end-systolic results of the Snake algorithm applied to all 31 data sets compared with the results from the manually drawn contours. The correlation is very high but this doesn't really resemble the results from the contour overlap evaluation.

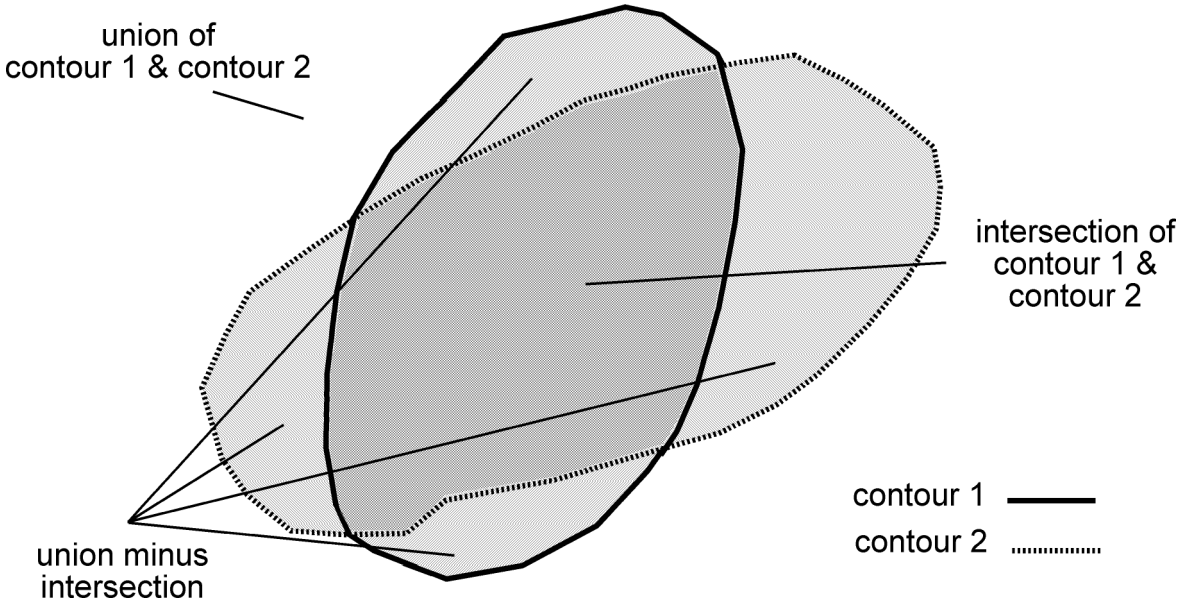

Figure 6.11: Illustration of contour comparison performance measure. Two overlapping contours define an intersection and a union region. The intersection is the interior part with the crossed lines as filling. If we take the area of union region minus intersection region and divide it by the union region, we get an appropriate performance measure for the degree of overlapping of the two contours.

- The Snake sometimes collapses to a line due to the chosen parameterization, this problem could be solved by fine-tuning the Snake parameters, but there is no general possibility to prevent this effect (see Figure 6.13d).
- The distinction between merged ventricle chambers is a very hard problem for the Snake because there are no strong gradients in such a region. This problem would vanish by specifiying fixed points on the border of the chambers (see Figure 6.13e).
- The papillary muscle is sometimes excluded from the segmented contour due to the fact that it consists of the same tissue as the heart muscle, so there is no gradient information for the Snake to adapt to this muscle. Again fixed Snake points would remove this drawback. Nevertheless if the initial contour is chosen appropriately this problem does not occur very often (see Figure 6.13f).

Summing up, the Snake tool shows good performance, but is far from being optimal for left ventricle segmentation. The partial volume effects and the weak contour information are problems that every technique based on the calculation of image gradients will show. But there are some further problems for which the Snake algorithm provides a sub-optimal solution. The parameterization of the Snake contour introduces a non-intuitive way to change the behaviour of the Snake. It is quite difficult to find a good parameterization given a certain kind of problem, there is no direct mapping from the problem to the parameterization. Further the introduction of fixed points to more accurately extract the papillary muscle and the ventricle borders leads to a high effort of the operator. If we directly compare the time-effort of parametric model and Snake-based segmentation model, it is more costly to find the images contributing to the desired volume, place initial contours on these images

additionally selecting some fixed points specifying border, apply the Snake algorithm and correct some points which were attracted to the wrong image feature. Additionally it could be necessary to apply further iterations of the Snake algorithm for a better segmentation. The time-effort of the Snake segmentation could be reduced by propagating contour results to adjacent slices of the data set. But there are two reasons why this method is not very accurate. First of all the local behaviour of the Snake implementation is very sensitive to misplaced initial contours and the slice thickness of our data is quite large with its 8 mm. So the difference in ventricle position between adjacent images is mostly too high.

A further disadvantage of the Active Contour model is that initial contours have to be placed near the ventricle chambers, but while placing contour points the operator gets no feedback of the expected algorithm performance. In order to be able to evaluate the results, all the contour points must have been placed and the algorithm must have been performed.

Some examples for the performance of the Snake segmentation algorithm are shown in Figure 6.14 and Figure 6.15. Both data sets resulted in accurate volume estimations due to well-placed contours. Problems with the papillary muscle didn't occur, and the contours were strong enough. Nevertheless it can be seen that a manual post-processing step would be useful to correct some misplaced contour points (e.g. slice 21 from data set 10 or slice 27 from data set 20).

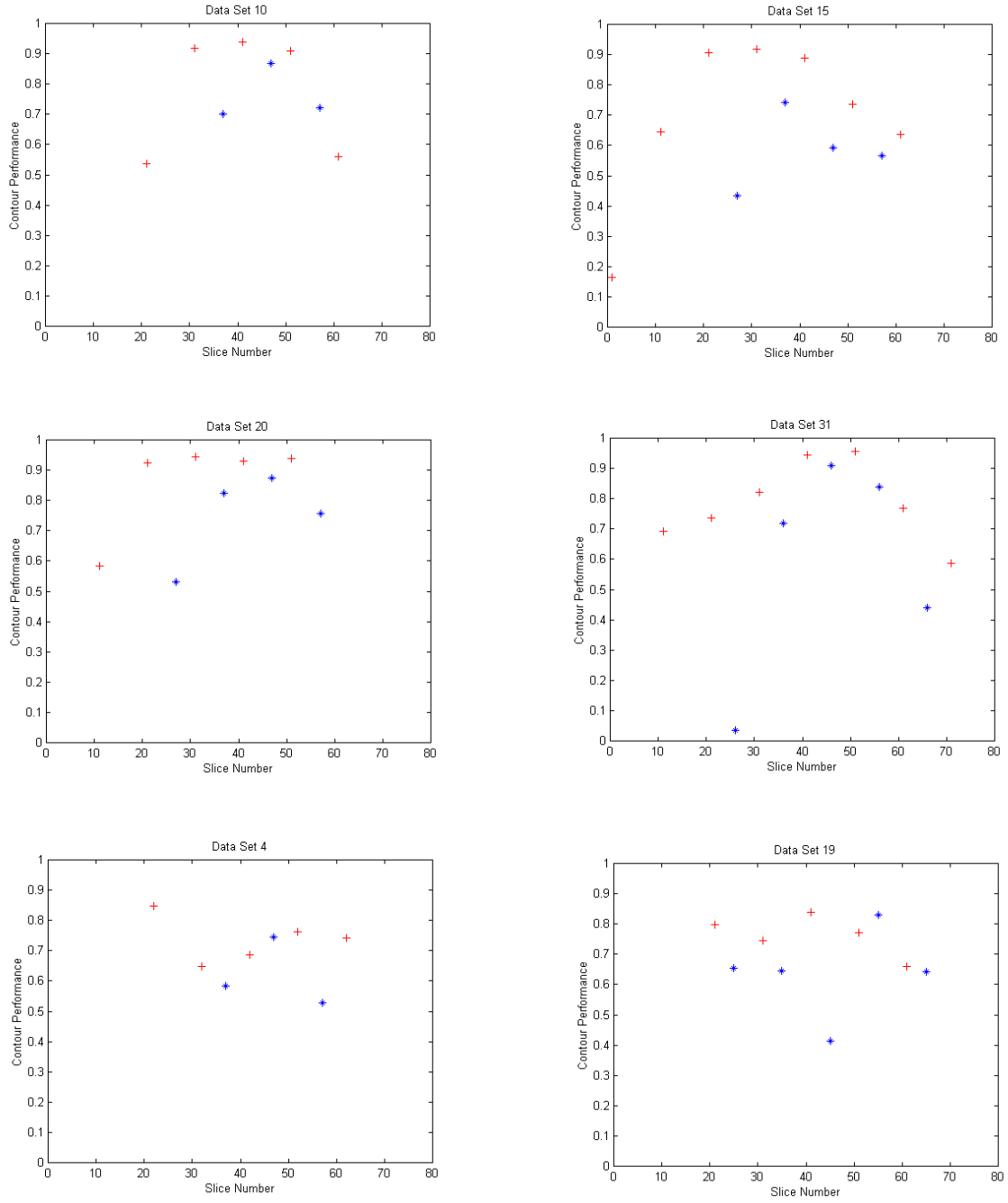

Figure 6.12: Comparison of manually drawn contours and Snake results. These charts show some representative results of the comparison of manually drawn and Snake contours. Data sets 10, 15 and 20 show some of the better results with high overlapping performance. Data set 31 shows an outlier at slice 26, where the two contours do not overlap at all. Data sets 4 and 19 deliver quite bad results. End-diastolic image slices are indicated by a '+', end-systolic ones by a '*'. Note the tendency of the performance values to be higher on the inner slices, where the gradient information is better. The outer slices are more difficult due to partial volume effects and less contrast.

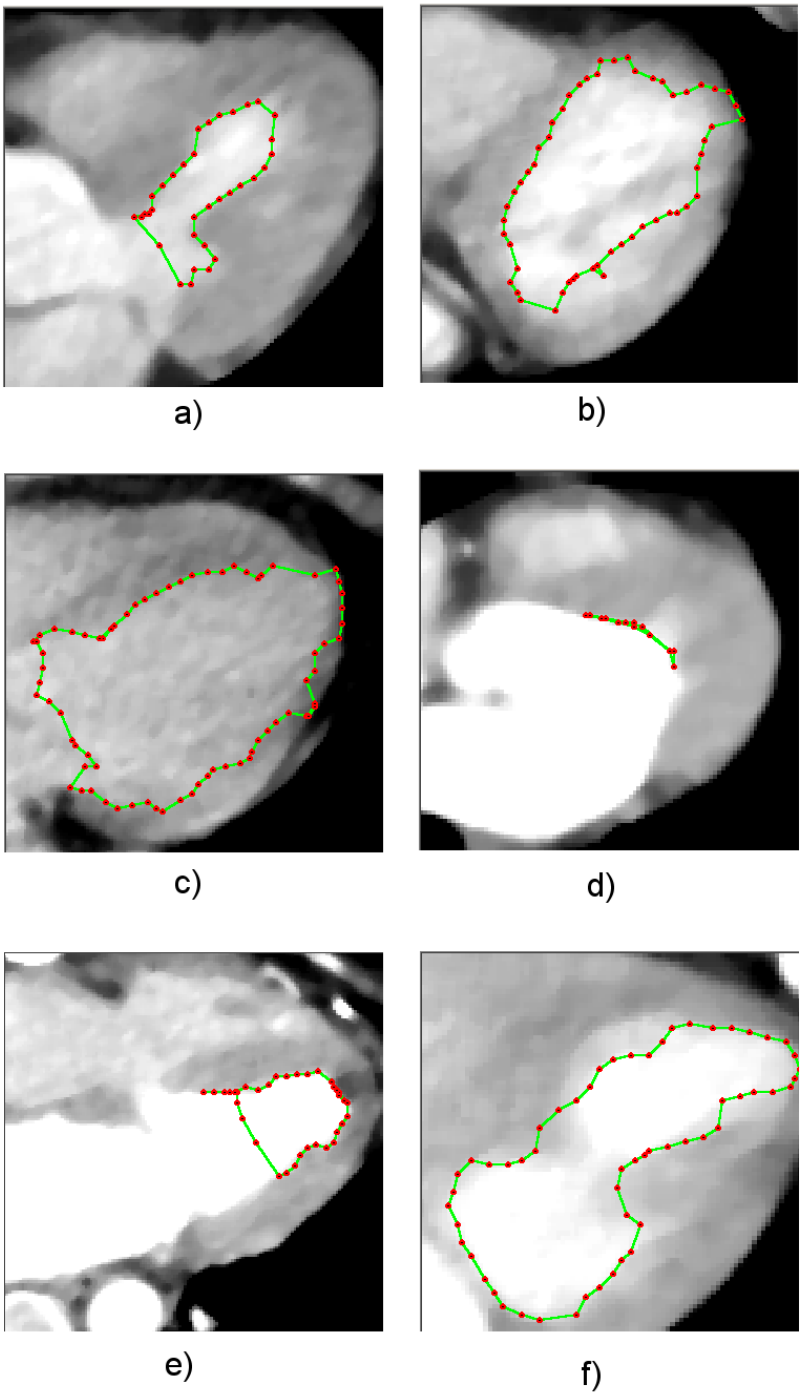

Figure 6.13: These figures show the problems when working with Snakes for segmentation. a) illustrates the partial volume effect at the top right region of the contour. b) shows the influence of a too strong gradient. c) demonstrates an image with very weak contrast. d) shows a collapsed Snake contour and e) a failing distinction between left ventricle and left atrium. Finally f) as well as a) illustrate the ignored papillary muscle.

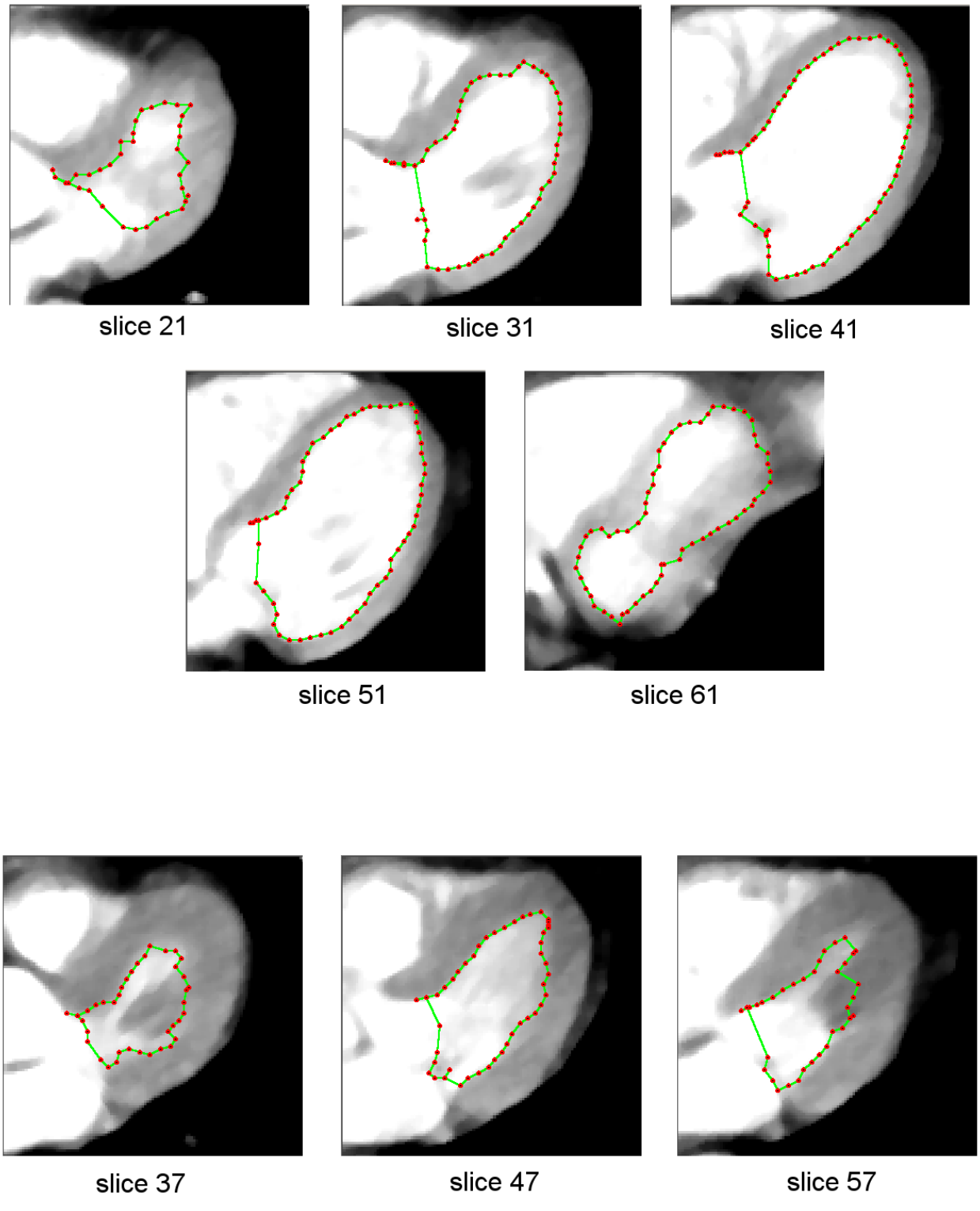

Figure 6.14: Snake result Data Set 10.

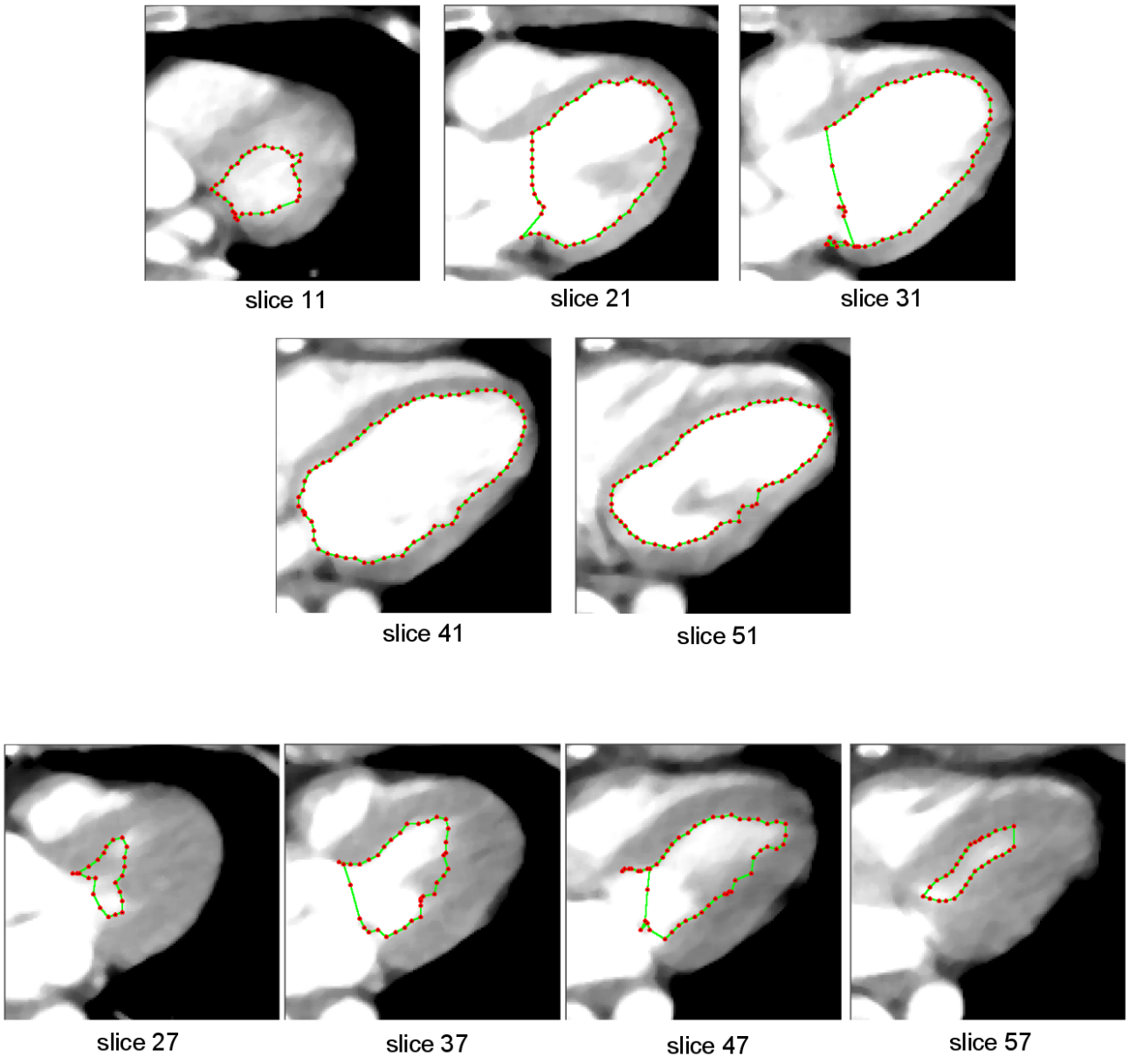

Figure 6.15: Snake result Data Set 20.

6.2.4 Volume via Intelligent Scissors (Live Wire) Segmentation

The evaluation of the Live Wire algorithm will be performed in a similar way as the Snakes evaluation. The most important reason for this is to be able to compare both techniques.

A very useful advantage of the implemented Live Wire segmentation technique is that there are few parameters to be set, most of them having straightforward definable values, which are suitable for a wide range of images. Nevertheless there are some steps that need to be performed. First of all the well-known window-level transformation (window: 300 HU, level: 50 HU) is applied, followed by a 5x5 median filter to remove noise. Now the actual algorithm is initialized by placing a start (seed) point in the image.

As explained in Section 4.4 the first algorithm step is to calculate the static local cost map of the image. Therefore we apply three different Laplacian of Gaussian convolutions and add them together to form a static cost function $f_M(q)$. We use a 5x5, a 9x9 and a 15x15 kernel, respectively, all of them being defined in floating point accuracy. The three convolution results are added with the following weights:

$$f_Z = 0.15 * LoG_5 + 0.45 * LoG_9 + 0.4 * LoG_{15}$$

This reflects the 5x5 kernel's tendency to produce a lot of noise, so its weight is lower compared to the other two kernels, which detect stronger edges quite successfully. A slight emphasis lies on the 9x9 kernel because the 15x15 kernel tends to find only real strong gradients like the border between heart and lung, with the result of attracting the Live Wire path onto this border.

Gradient magnitude and gradient direction cost functions were calculated by convoluting the image with the two 3x3 Sobel kernels for the gradients in x- and y-direction. They were calculated according to the description in the algorithm section.

The weights of the local cost functions are chosen like this:

ω_Z : 0.4

ω_G : 0.4

ω_D : 0.2

This resembles that the gradient location and magnitude informations are far more important than the smoothness of the boundary incorporated in the gradient direction function.

For the actual implementation of the minimum cost path map creation some other parameters have to be specified. The discretization value of the cost function M, which represents the discrete range of possible values the cost function can have, is chosen to be 128. As a consequence $NR\_BUCKETS$ is equal to $128 * \sqrt{2 * 360^2} = 65166$ (image width and height are 360, respectively). So we have an active nodes list with 65166 entries. Of course this number could be chosen smaller by implementing the active nodes list as a ring buffer like suggested in the paper, but there is no dramatic performance gain using this technique in the author's opinion.

After placing the seed point in the image, the operator has to point the mouse cursor along the border of the object to be segmented. Again the operator was the author, the same considerations as in the Snakes evaluation apply here, too. It is assumed that the error due to the author's lack of radiological experience does not have a great impact on the results, because the exact segmentation is performed by the Live Wire algorithm.

While pointing around with the mouse cursor and using the path cooling feature, a closed contour is defined. This contour is used to crop the left ventricle region and calculate a volume. The volumes

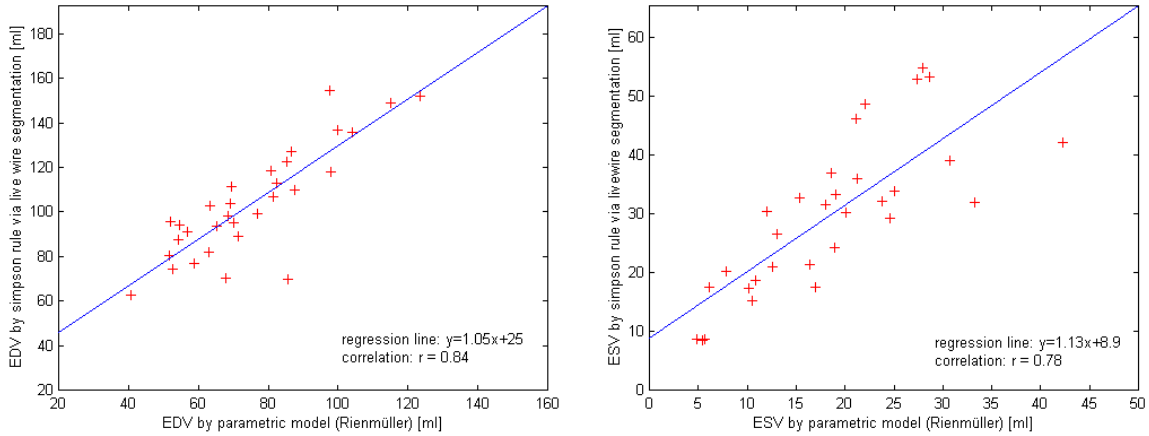

Figure 6.16: Evaluation of volumes by means of Live Wire segmentation and parametric model. End-diastolic and end-systolic results of the Live Wire algorithm applied to all 31 data sets compared with the results from the parametric model. The performance is quite good, it is comparable with the Active Contours and the manual contours evaluation.

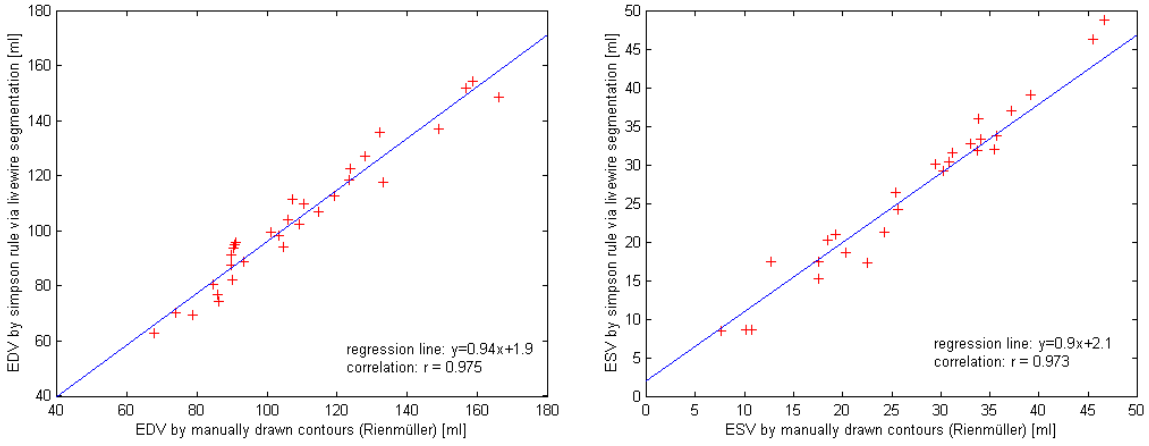

Figure 6.17: Evaluation of volumes by means of Live Wire segmentation. End-diastolic and end-systolic results of the Live Wire algorithm applied to all 31 data sets compared with the results from the manually drawn contours. The correlation is excellent leading to the assumption that the Live Wire tool would be a great enhancement in the hands of a radiologist.

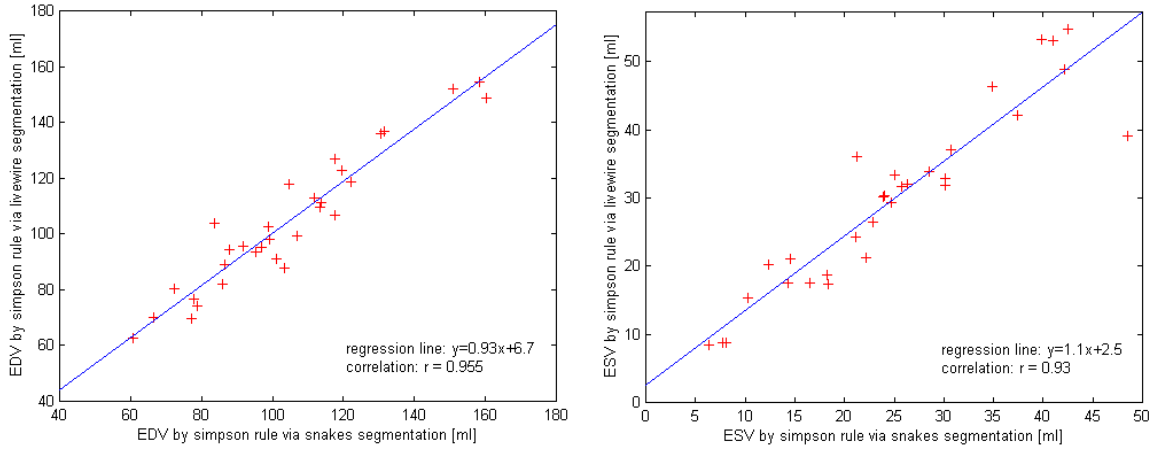

Figure 6.18: Comparison of volumes calculated after Live Wire and Snakes segmentation. End-diastolic and end-systolic results of Snake and Live Wire algorithms applied to all 31 data sets. The correlation is high proving that the results are well comparable.

of the Live Wire segmentation are compared with the volumes from the parametric model (Figure 6.16), the manually drawn contours (Figure 6.17) and the Snake segmentation (Figure 6.18).

What we can conclude from these evaluations is, that Snake and Live Wire segmentation behave quite similar with respect to the parametric model results. The regression lines and the correlation coefficients are nearly equal. Very The excellent behaviour of the Live Wire segmentation result compared with the contours drawn by Prof. Rienmüller is very interesting. It shows a high correlation of more than 97% in the end-diastolic as well as the end-systolic case. Furthermore we have a nearly linear regression. Obviously the consequence of this result is that it is possible to imitate the contour definition of an experienced radiologist by simply looking at his contours and drawing the Live Wire contour. In this context it should be noted again, that the operator who performed the Live Wire segmentation was the author of this work, who has no radiological experience. So it can be assumed that Live Wire is a very powerful tool if used by an experienced radiologist.

Nevertheless there is an obvious disadvantage, the time-consumption of the method. Although it is a lot faster than manual contour definition, processing of data still requires a considerable amount of time. We also realized that the time consumption of the Live Wire segmentation and the Active Contours segmentation is similar, with Live Wire being far more accurate. Another disadvantage is that an automatization of Live Wire is not possible, while well-defined and well-parameterized Snakes could be utilized in a more automatic schema.

The direct comparison of the Live Wire and the Snake segmentation provides no more interesting information, it only validates the similarity of both techniques by showing their good correlation.

Figure 6.19 shows some steps of interactively segmenting images utilizing the Live Wire mode of the implemented segmentation tool.

Again, similar to the Snakes evaluation, the accuracy of the computed contours is measured by comparing them with the manually drawn contours. The previously presented technique of overlapping the contour regions and calculating a relative *contour_performance* is applied. The results are illustrated in Figure 6.20. These results again show the excellent performance of the Live Wire segmentation, the accuracy of the contours is much higher compared to the Snakes contours. The figure shows the same data sets presented in the Snakes contour evaluation sections.

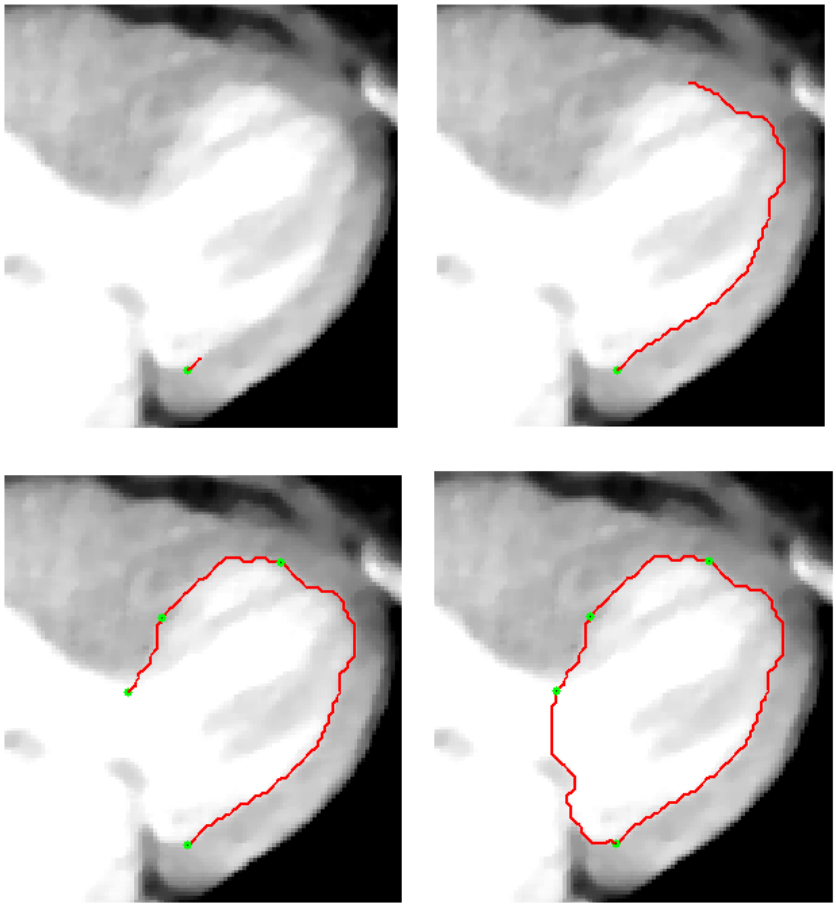

Figure 6.19: Example for drawing a Live Wire contour. The top left image shows the seed point, afterwards we drag the mouse cursor along the ventricle border. On the left side of the border it becomes necessary to set some additional start points (path cooling). It is easily possible to draw the line separating the left ventricle from the other blood-filled chambers. The papillary muscle can also easily be avoided.

Some results for good and bad ventricle segmentations are shown in Figure 6.21 and Figure 6.22. The overall performance of the ventricle contours is excellent, so we don't have images showing a critical failure of the algorithm. For comparison reasons we present the same data sets as in the Snake evaluation section, namely data sets 10 and 20. Further some images from a data set with very weak contours are shown, compared to the manually drawn contours (Figure 6.23). Here we can see the fine overlap of manually drawn contours and Live Wire contours even if the underlying images have low contrast. The livewire contour is drawn in red and green while the manual contour is blue-yellow. Finally Figure 6.23 shows a contour comparison provided a data set with good contrasts. We can see that the problems of defining a border between left ventricle and surrounding heart chambers as well as the inclusion of the papillary muscle are solved here.

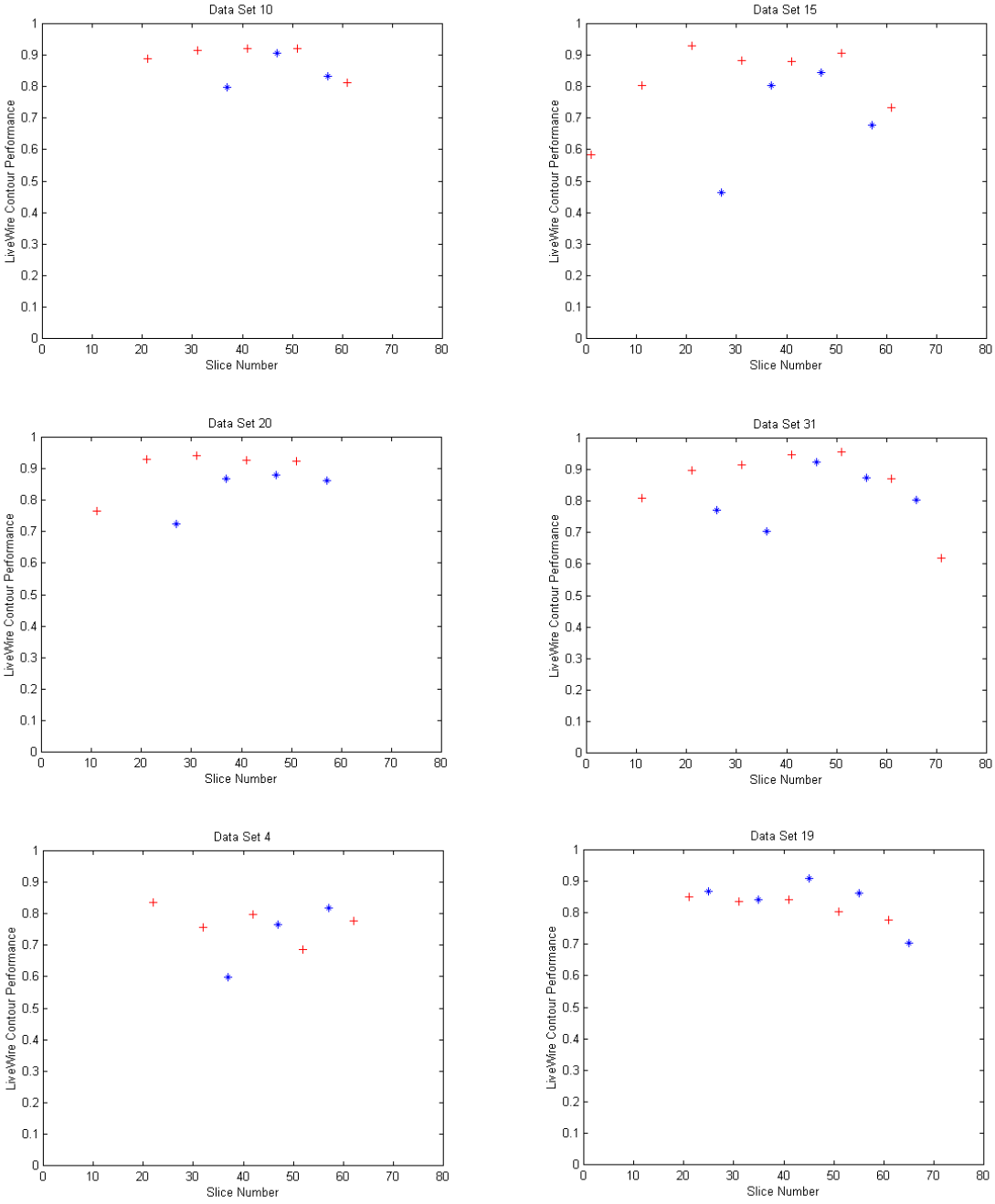

Figure 6.20: Comparison of manually drawn contours and Live Wire results. These charts show some representative results of the comparison of manually drawn and Live Wire contours. All results are better than the results after Snake segmentation, some are even dramatically better like e.g. data set 19. End-diastolic image slices are indicated by a '+', end-systolic ones by a '*'. Note again the tendency of the performance values to be higher on the inner slices, where the gradient information is better. The outer slices are worse due to partial volume effects and less contrast. Nevertheless these effects are not as strong as in Figure 6.12.

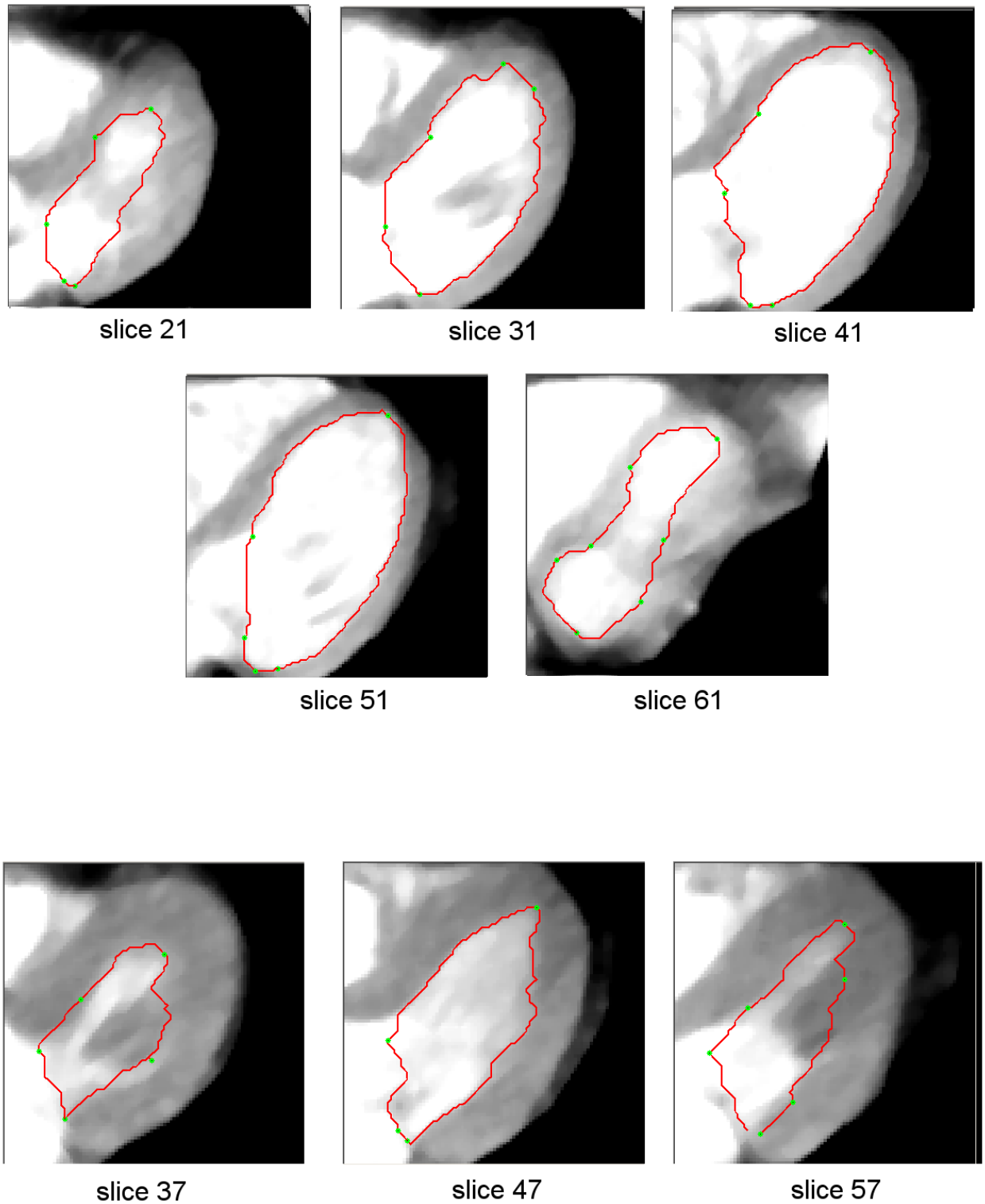

Figure 6.21: Live Wire result Data Set 10.

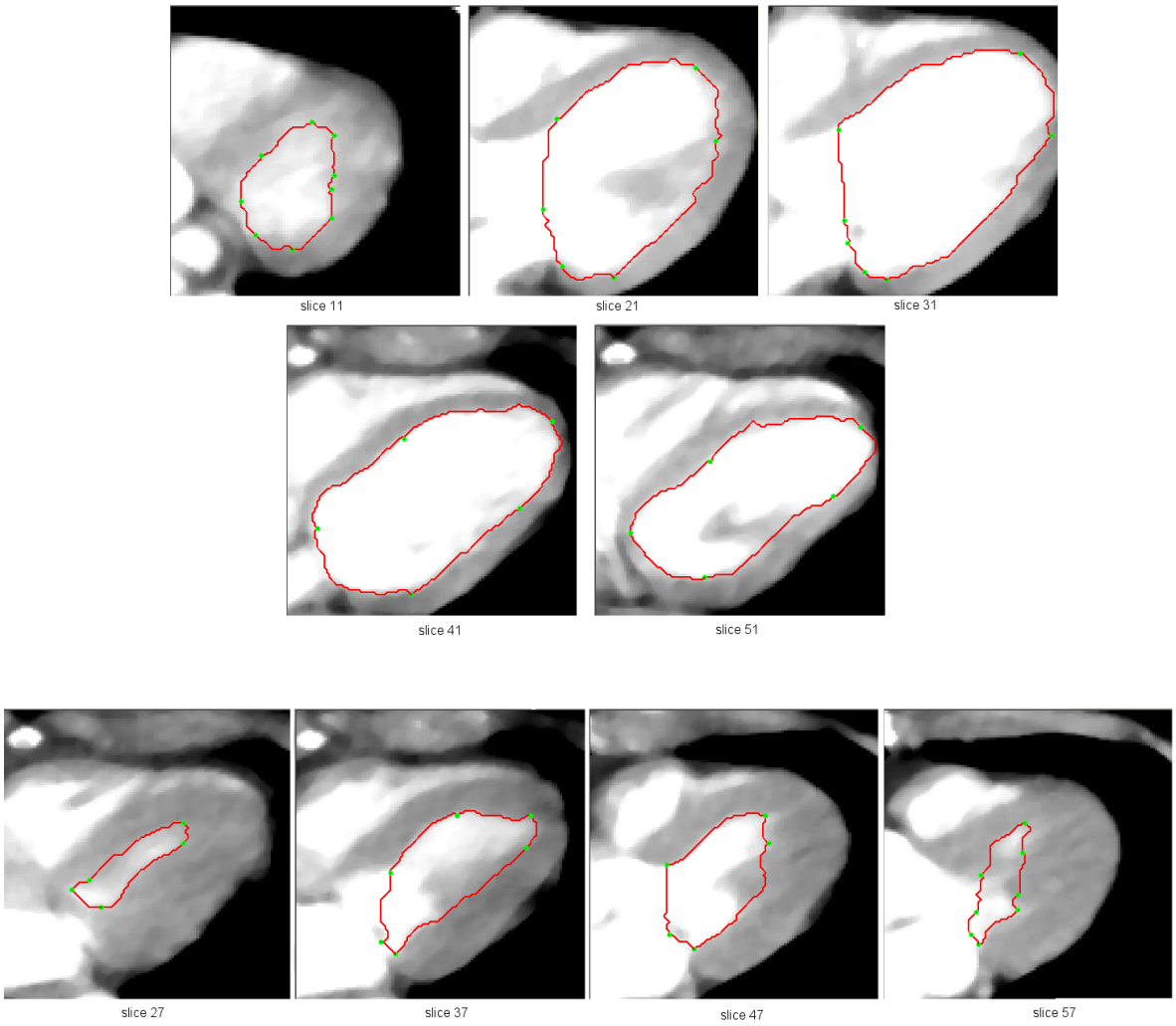

Figure 6.22: Live Wire result Data Set 20.

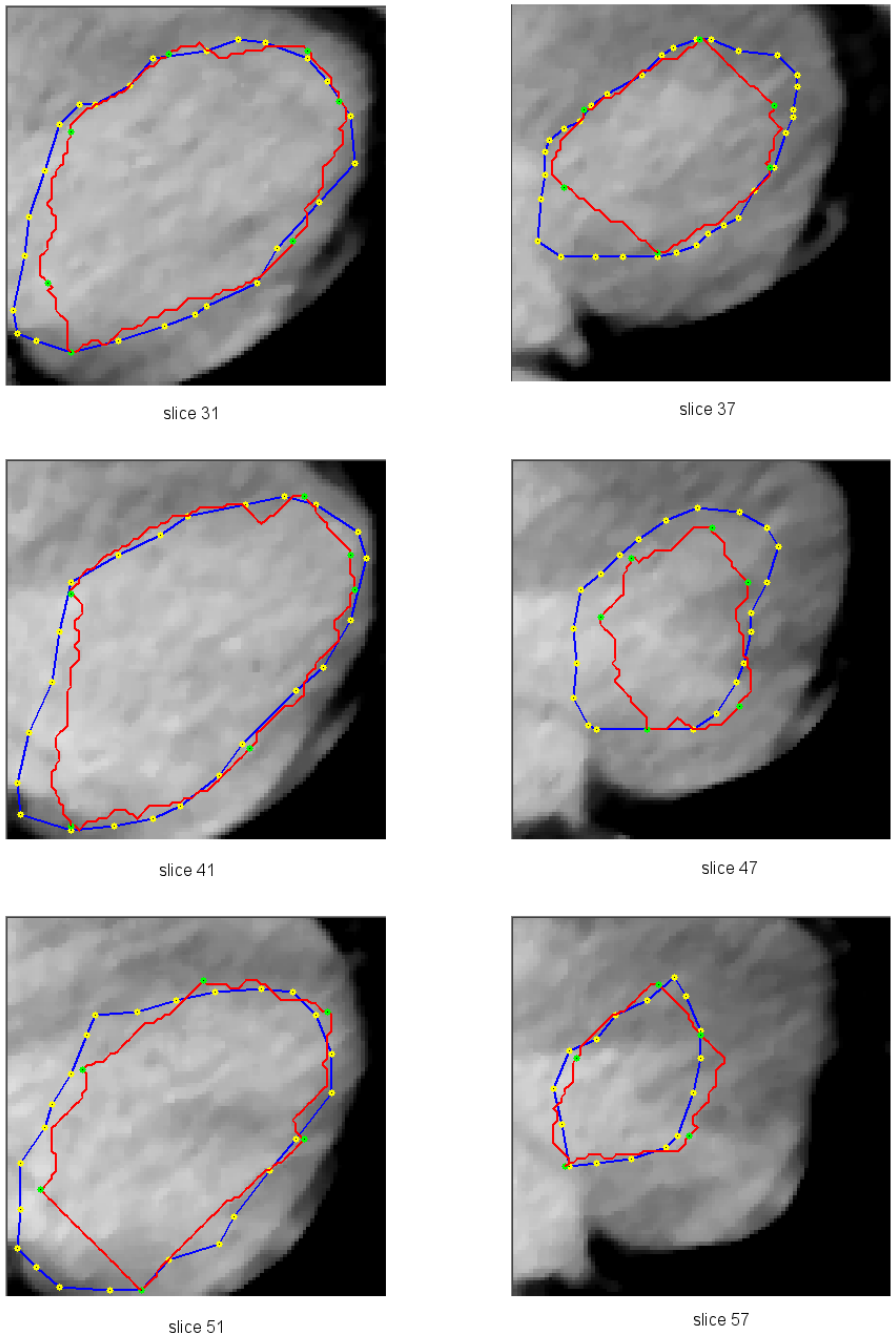

Figure 6.23: Live Wire result Data Set 13. A comparison between manually drawn contours and Live Wire contours on an image data set with low contrast. The Live Wire contour is red-green, the manual one blue-yellow.

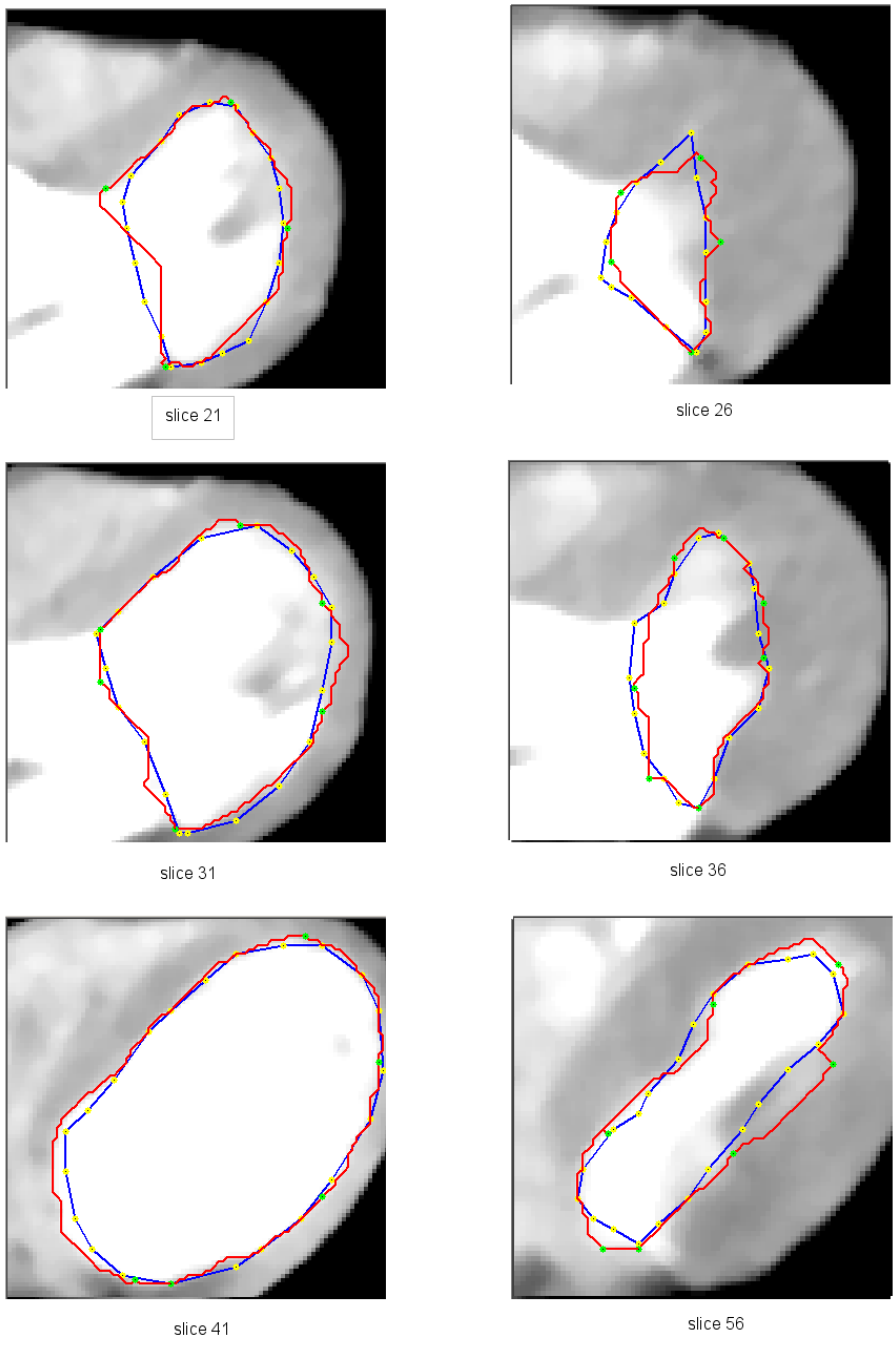

Figure 6.24: Live Wire result Data Set 12. A comparison between manually drawn contours and Live Wire contours. The livewire contour is red-green, the manual one blue-yellow. We can see that the definition of the border between left ventricle and other heart chambers is performed very well. The inclusion of the papillar muscle into the segmented volume also is no problem.

6.3 Summary and Discussion

In this chapter we presented the experiments that we performed on the medical image data sets. The algorithm for automatic ROI detection was found to be useful, while the ellipse detection algorithm didn't show appealing results. Furthermore we presented the evaluations of the end-diastolic and end-systolic volumes derived by the different segmentation techniques and the parametric model.

An important conclusion of the evaluations is that there are large variances in the application of the parametric model. If there are two different operators evaluating the same data set, we end up with a systematic error as well as a weak correlation. This shows the inter-operator variability which is due to different assumptions and experience with the medical and radiological aspects. The obvious implication is that the parametric model lacks in reproducibility.

The manually drawn contours of Prof. Rienmüller compared with the parametric model emphasize the consistency of his contours which correspond to a certain ventricle shape he has in mind during work. Prof. Rienmüller is a very experienced radiologist, therefore his results are constant and reproducible. If we compare our results from the Active Contour and the Live Wire segmentation with the parametric model we get similar results like from the comparison to the manually drawn contours. This validates that our methods are also reproducible. The accuracy of our models (and especially the Live Wire model) are validated in the direct comparison of our contours with Prof. Rienmüller's. The high correlation between these two models implies that we found a way to easily derive contours which lead to volume results similar to Prof. Rienmüller's results with the parametric model. The main difference is that we don't need a long learning curve to obtain these results but can find reproducible results with the help of the segmentation tool.

Nevertheless a certain degree of radiological experience is a prerequisite for the application of our tools, but much of the time-consuming low-level work is performed by the tool. The author simulated the medical experience by looking at the contours of Prof. Rienmüller before using the segmentation tools.

Especially the Live Wire algorithm was very useful in segmentation, so we can conclude that it would be a very powerful tool in the hands of a radiologist. With this tool it was possible to perform an important step in investigating comparability and reproducibility of parametric models and image segmentation (Simpson-Rule based) models.

Chapter 7

Summary

7.1 Conclusion

In this document we presented an image processing system to calculate the volume of left ventricles from computer tomographic image slices of the heart. The 31 medical image data sets which were to be investigated come from an Ultrafast CT Scanner. This scanner is fast enough to image a beating heart accurately. The images are available in the DICOM standard file format and show the left ventricle.

A parametric model and digital image processing techniques for volume estimation were investigated to enable a direct comparison. The parametric model is called *Two-Axes Method by Greene*, this is the model radiologists at the Department for Radiology use to determine ventricle volumes. It makes use of the assumption that the left ventricle has an approximate ellipsoid contour and calculates the volume of this ellipsoid by seeking the slice with a typical projection of the ventricle and measuring ellipse parameters in this projection.

The digital image processing algorithms which were applied to the data sets are an automatic region-of-interest finding algorithm using the temporal information in the images, a Hough transform for ellipse detection to automatically locate the projection needed for the parametric model, and three different segmentation techniques to calculate volumes by separating the left ventricle area from the rest of the images and sum up the number of pixels of the separated regions multiplied by their voxel size over the slices. The region-of-interest algorithm showed good performance. In nearly all of the data sets a meaningful ROI could be found. The ellipse detection algorithm did not work very well due to different reasons. One of them is the algorithm's failing robustness if the ellipses to be found are not exact or too much additional information is incorporated in the images. Another one is the large number of parameters that have to be tuned in our implementation of the ellipse Hough transform.

The main part of the work was concerned with volume estimations. First of all a reference estimation was produced by comparing the volumes of the parametric model with the volume derived from manually drawing left ventricle contours into the images. These manual contours came from an experienced radiologist. Unfortunately it is necessary to be content with this kind of reference estimation due to the fact that there are no absolutely correct volume estimations available to compare with. Calculating the volume of in vivo (living) ventricles would only be possible by a complicated and dangerous operation and the measurement of dead hearts is not really representative, because a CT scan of a non-beating heart does not show imaging artifacts like e.g. artifacts due to motion aliasing.

The segmentation techniques utilized for the comparisons were *Thresholding*, *Active Contours (Snakes)* and *Intelligent Scissors (Live Wire)*, based on graph-theory. The following table shows the

results of the segmentation techniques with respect to certain problem areas:

problem area	Thresholding	Snakes	Live Wire
accuracy of the estimated volumes	very weak	good	very good
accuracy of the segmented contours	-	good	very good
total time for segmentation	very fast	slow	slow
sensitivity to partial volume effect	high	high, hard to take into account	high, easy to take into account
sensitivity to weak image contrast	high	high, hard to take into account	high, easy to take into account
dependency on algorithm parameters	no	yes	no

The table shows that the Live Wire segmentation technique is the most useful one. It has the advantage that it supports a manual segmentation by an operator in an optimal way by specifiying an exact delineation of a contour while the operator is responsible for the recognition of the object to be segmented. The problems which appeared with the Snakes algorithm like tendency to be drawn to high gradients, laborious parameter tuning and problems with region separation and papillary muscles can easily be avoided with the Live Wire tool. We think that in the hands of an experienced radiologist this tool is a very powerful one to segment images, especially in situations where full-automatic segmentation techniques fail.

7.2 Future Work

Some possible areas for future work are:

- Testing 3D segmentation techniques like a 3D region growing algorithm. The third dimension incorporates new information which can be used for an effective segmentation. A manually defined cutting plane could be used to separate the different heart chambers. Nevertheless this approach will fail in including the papillary muscle and the problem of the low z-resolution of the image slices (only 8 image slices can be used for the 3D visualization of the heart) is to be considered.

- Incorporate learning and classification algorithms like an Active Appearance Model. By providing more similar data sets it would be possible to establish a training set of left ventricles and build a classificator, e.g. by an Active Shape or an Active Appearance Model which is able to automatically segment image data sets. Here either a more powerful and more robust way to automatically locate end-diastolic and end-systolic images has to be found or this choice has to be incorporated in a separate classification schema.

- Test other algorithms for ellipse detection. Perhaps a generalized Hough transform would be capable of automatically finding the interesting projections for the parametric model.

Bibliography

[1] Raj Acharja, Richard Wasserman, Jeffrey Stevens, and Carlos Hinojosa, *Biomedical Imaging Modalities: An Overview*, in Singh et al. [53], pp. 20–44.

[2] A.A. Amini, S. Tehrani, and T.E. Weymouth, *Using Dynamic Programming for Minimizing the Energy of Active Contours in the Presence of Hard Constraints*, International Conference on Computer Vision, 1988, pp. 95–99.

[3] Marek Belohlavek, *Quantitative Three-Dimensional Echocardiography: Image Analysis for Left Ventricular Volume*, Ph.D. thesis, The Mayo Graduate School, August 1996.

[4] Andrew Blake and Michael Isard, *Active Contours*, Springer, 1998.

[5] CA Brooks and G. DiChico, *Theory of Image Reconstruction in Computed Tomography*, Radiology **117** (1975), pp. 561–572.

[6] V. Caselles, F. Catte, T. Coll, and F. Dibos, *A Geometric Model for Active Contours*, Numerische Mathematik **66** (1993), pp. 1–31.

[7] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic Active Contours*, International Journal of Computer Vision **22** (1997), no. 1, pp. 61–79.

[8] Y.C. Cheng and S.C. Lee, *A New Method for Quadratic Curve Detection Using K-Ransac with Acceleration Techniques*, Pattern Recognition **28(1)** (1995), no. 5, pp. 663–682.

[9] W.P. Choi, K.M. Lam, and W.C. Siu, *An adaptive active contour model for highly irregular boundaries*, Pattern Recognition **34** (2001), no. 2, pp. 323–331.

[10] L.D. Cohen, *On Active Contour Models and Balloons*, CVGIP: Image Understanding **53** (1991), no. 2, pp. 211–218.

[11] T.F. Cootes, G.J. Edwards, and C.J. Taylor, *Active Appearance Models*, Pattern Analysis and Machine Intelligence **23** (2001), no. 6, pp. 681–684.

[12] T.F. Cootes, A. Hill, C.J. Taylor, and J. Haslam, *Use of Active Shape Models for Locating Structure in Medical Images*, Image and Vision Computing **12** (1994), no. 6, pp. 355–365.

[13] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, MIT Electrical and Computer Science Series, MIT Press, 1992.

[14] Intel Corporation, *Open Computer Vision Library homepage*, http://www.intel.com/research/-mrl/research/opencv/, 2001.

[15] M. Dulce, G. Mostbeck, K. Friese, G. Caputo, and C. Higgins, *Quantification of Left Ventricular Volumes and Function with Cine MR Imaging: Comparison of Geometric Models with Three-Dimensional Data*, Radiology **188** (1993), pp. 371–376.

[16] N. Duta and M. Sonka, *Segmentation and Interpretation of MR Brain Images: An Improved Active Shape Model*, IEEE Transactions on Medical Imaging **17** (1998), no. 6, pp. 1049–1062.

[17] A.X. Falcao, J.K. Udupa, and F.K. Miyazawa, *An Ultra-Fast User-steered Image Segmentation Paradigm - Live Wire on the Fly*, IEEE Transactions on Medical Imaging **19** (2000), no. 1, pp. 55–62.

[18] A.X. Falcao, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, and R.D.A. Lotufo, *User-steered Image Segmentation Paradigms - Live Wire and Live Lane*, Graphical Models and Image Processing **60** (1998), no. 4, pp. 233–260.

[19] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos, *Dynamic-Programming for Detecting, Tracking, and Matching Deformable Contours*, Pattern Analysis and Machine Intelligence **17** (1995), no. 3, pp. 294–302.

[20] Samuel M. Goldwasser, Anthony R. Reynolds, Ted Bapty, David Baraff, John Summers, David A. Talton, and Ed Walsh, *Physician's Workstation with Real-Time Performance*, IEEE Computer Graphics and Applications **5** (1985), no. 12, pp. 44–56.

[21] Ardeshir Goshtasby and David A. Turner, *Segmentation of Cardiac Cine MR Images for Extraction of Right and Left Ventricular Chambers*, IEEE Transactions on Medical Image Processing **14** (1995), no. 1, pp. 56–64.

[22] N. Guil and E.L. Zapata, *Lower Order Circle and Ellipse Hough Transform*, Pattern Recognition **30** (1997), no. 10, pp. 1729–1744.

[23] A. Gupta, T. O'Donnell, and A. Singh, *Segmentation and Tracking of Cine Cardiac MR and CT Images Using a 3-D Deformable Model*, IEEE Conf. Computers in Cardiology, 1994, pp. 661–664.

[24] A. Gupta, L. von Kurowski, A. Singh, D. Geiger, C-C. Liang, M-Y. Chiu, L.P. Adler, M. Haacke, and Wilson D.L., *Cardiac MR Image Segmentation Using Deformable Models*, IEEE Conf. Computers in Cardiology, 1993, pp. 747–750.

[25] J. Gutierrez, I. Epifanio, E. de Ves, and F.J. Ferri, *An Active Contour Model for the Automatic Detection of the Fovea in Fluorescein Angiographies*, Proc. of the International Conference on Pattern Recognition, vol. IV, 2000, pp. 312–315.

[26] Chun-Ta Ho and Ling-Hwei Chen, *A Fast Ellipse/Circle Detector Using Symmetry*, Pattern Recognition **28(1)** (1995), no. 1, pp. 117–124.

[27] P.V.C. Hough, *A Method and Means for Recognizing Complex Patterns*, US Patent 3,069,654, 1962.

[28] Johannes Hug, Christian Brechbuhler, and Gabor Szekely, *Tamed Snake: A Particle System for Robust Semi-automatic Segmentation*, Medical Image Computing and Computer-Assisted Intervention, 1999, pp. 106–115.

[29] IMATRON, *Longitudinal View of Imatron Scanner*, http://www.imatron.com/engineering.htm, 2001.

[30] Intel, *Intel Image Processing Performance Library homepage*, http://developer.intel.com/-software/products/perflib/ipl/index.htm, 2001.

[31] Tim D. Jones and Peter Plassmann, *An Active Contour Model for Measuring the Area of Leg Ulcers*, IEEE Transactions on Medical Imaging **19** (2000), no. 12, pp. 1202–1210.

[32] M. Kass, A. Witkin, and D. Terzopoulos, *SNAKES: Active Contour Models*, International Journal of Computer Vision **1(4)** (1988), pp. 321–332.

[33] Pat Kerrisk, *Cardiovascular Lecture*, http://www.healthsci.utas.edu.au/anatomy/pat__kerrisk/-lectures/cardiovascular/index.htm, 2001.

[34] J. Liang, T. McInerney, and D. Terzopoulos, *United snakes*, IEEE Proceedings on Computer Vision, vol. 2, 1999, pp. 933–940.

[35] Paul R. Lichtlen, *Beiträge zur Kardiologie: Band 11: Koronarangiographie*, Perimed Verlag Dr. med. D. Straube, Erlangen, Deutschland, 1979.

[36] Liana M. Lorigo, Olivier D. Faugeras, W. Eric L. Grimson, Renaud Keriven, and Ron Kikinis, *Segmentation of Bone in Clinical Knee MRI Using Texture-Based Geodesic Active Contours*, Medical Image Computing and Computer-Assisted Intervention, 1998, pp. 1195–1204.

[37] R. Malladi, J.A. Sethian, and B.C. Vemuri, *Shape Modeling with Front Propagation: A Level Set Approach*, Pattern Analysis and Machine Intelligence **17** (1995), no. 2, pp. 158–175.

[38] T. McInerney and D. Terzopoulos, *A Dynamic Finite Element Surface Model for Segmentation and Tracking in Multidimensional Medical Images with Application to Cardiac 4D Image Analysis*, Computerized Medical Imaging and Graphics **19** (1995), no. 1, pp. 69–83.

[39] R.A. McLaughlin, *Randomized Hough Transform: Improved Ellipse Detection with Comparison*, Pattern Recognition Letters **19** (1998), no. 3-4, pp. 299–305.

[40] E.N. Mortensen and W.A. Barrett, *Interactive Segmentation with Intelligent Scissors*, Graphical Models and Image Processing **60** (1998), no. 5, pp. 349–384.

[41] NEMA, *DICOM Specification*, http://medical.nema.org/dicom.html, 2001.

[42] Timothy S. Newman and Hong Yi, *Compound Extraction and Fitting Method for Detecting Cardiac Ventricle in SPECT Data*, Proc. of the International Conf. on Pattern Recognition, 2000, pp. 328–331.

[43] K.P. Philip, E.L. Dove, D.D. McPherson, N.L. Gotteiner, M.J. Vonesh, W. Stanford, J.E. Reed, J.A. Rumberger, and K.B. Chandran, *Automatic Detection of Myocardial Contours in Cine-Computed Tomographic Images*, IEEE Transactions on Medical Image Processing **13** (1994), no. 2, pp. 241–253.

[44] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in C*, 2nd ed., Cambridge University Press, Cambridge, England, 1993.

[45] Surendra Ranganath, *Contour Extraction from Cardiac MRI Studies Using Snakes*, IEEE Transactions on Medical Image Processing **14** (1995), no. 2, pp. 328–338.

[46] RB Rehr, CR Malloy, NG Filipchuk, and RM Peshock, *Left Ventricular Volumes Measured by MR Imaging*, Radiology **156** (1985), pp. 717–719.

[47] R. Rienmüller, J. Lissner, A. Kment, J. Bohn, B.E. Strauer, D. Hellwig, E. Erdmann, J. Cyran, G. Steinbeck, D. Höss, and B. Höfling, *Das enddiastolische Volumen des linken Ventrikels in der Computertomographie im Vergleich zur Herzkatheterventrikulographie*, Computertomographie 1 **1** (1981), no. 2, pp. 62–67.

[48] Hilmi Rifai, Isabelle Bloch, Seth A. Hutchinson, Joe Wiart, and Line Garnero, *Segmentation of the skull in MRI volumes using a deformable model and taking the partial volume effect into account*, SPIE Medical Image Analysis 4, 2000, pp. 219–233.

[49] Thomas B. Sebastian, Huseyin Tek, Joseph J. Crisco, Scott W. Wolfe, and Benjamin B. Kimia, *Segmentation of Carpal Bones from 3D CT Images Using Skeletally Coupled Deformable Models*, Medical Image Computing and Computer-Assisted Intervention, 1998, pp. 1184–1194.

[50] RC Semelka, E Tomei, S Wagner, J Mayo, C Kondo, J Suzuki, GR Caputo, and CB Higgins, *Normal left ventricular dimensions and function: interstudy reproducibility of measurements with cine MR imaging*, Radiology **174** (1990), pp. 763–768.

[51] J.A. Sethian, *Level Set Methods and Fast Marching Methods*, 2nd ed., Cambridge University Press, 1999.

[52] Florence Sheehan, David C. Wilson, David Shavelle, and Edward A. Geiser, *Echocardiography*, in Sonka and Fitzpatrick [54], pp. 609–674.

[53] Ajit Singh, Dmitry Goldgof, and Demetri Terzopoulos (eds.), *Deformable Models in Medical Image Analysis*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[54] Milan Sonka and J. Michael Fitzpatrick (eds.), *Handbook of Medical Imaging: Volume 2 - Medical Image Processing and Analysis*, SPIE - The International Society for Optical Engineering, 2000.

[55] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image Processing, Analysis and Machine Vision*, 2nd ed., Brooks/Cole Publishing Company, Pacific Grove, CA, USA, 1999.

[56] L.H. Staib and J.S. Duncan, *Boundary Finding with Parametrically Deformable Models*, Pattern Analysis and Machine Intelligence **14** (1992), no. 11, pp. 1061–1075.

[57] William Stanford and John A. Rumberger, *Ultrafast Computed Tomography in Cardiac Imaging: Principles and Practice*, Futura Publishing Company, Inc., Mount Kisco, NY, USA, 1992.

[58] Alexander M. Taratorin and Samuel Sideman, *Constrained Detection of Left Ventricular Boundaries from Cine CT Images of Human Hearts*, IEEE Transactions on Medical Image Processing **12** (1993), no. 3, pp. 521–533.

[59] D.R. Thedens, D.J. Skorton, and S.R. Fleagle, *Methods of Graph Searching for Border Detection in Image Sequences with Applications to Cardiac Magnetic Resonance Imaging*, IEEE Transactions on Medical Imaging **14** (1995), no. 1, pp. 42–55.

[60] R. Toledo, X. Orriols, P. Radeva, X. Binefa, J. Vitria, C. Canero, and J.J. Villanueva, *Eigensnakes for Vessel Segmentation in Angiography*, Proc. of the International Conf. on Pattern Recognition, vol. IV, 2000, pp. 340–343.

[61] Trolltech, *Qt homepage*, http://www.trolltech.com/, 2001.

[62] John Weng, Ajit Singh, and M.Y. Chiu, *Learning-Based Ventricle Detection from Cardiac MR and CT Images*, IEEE Transactions on Medical Image Processing **16** (1997), no. 4, pp. 378–391.

[63] D. Williams and M. Shah, *A Fast Algorithm for Active Contours and Curvature Estimation*, CVGIP: Image Understanding **55** (1992), no. 1, pp. 14–26.

[64] Marco Wiltgen, *Digitale Bildverarbeitung in der Medizin*, Shaker Verlag, Aachen, Germany, 1999.

[65] W.Y. Wu and M.J.J. Wang, *Elliptical Object Detection by Using its Geometric Properties*, Pattern Recognition **26** (1993), pp. 1499–1509.

[66] R.K.K. Yip, P.K.S. Tam, and D.N.K. Leung, *Modification of Hough Transform for Circles and Ellipses Detection Using a 2-Dimensional Array*, Pattern Recognition **25** (1992), pp. 1007–1022.

[67] J.H. Yoo and I.K. Sethi, *An Ellipse Detection Method from the Polar and Pole Definition of Conics*, Pattern Recognition **26** (1993), pp. 307–315.

[68] S.C. Zhu and A. Yuille, *Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation*, Pattern Analysis and Machine Intelligence **18** (1996), no. 9, pp. 884–900.