TAMPERE UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING

Joutsen, Atte

Volume Visualization of Calculated Bioelectric Fields

MASTER OF SCIENCE THESIS

Subject approved by the department council 13.03.2002

Supervisors: Prof. Jari Hyttinen
Prof. Jaakko Malmivuo

Preface

The research for this thesis has been carried out at Ragnar Granit Institute, Tampere University of Technology. The world of visualization has been truly amazing and I would like to thank several people who have illuminated the path and given me support.

At first I wish to express my gratitude to the institute's director and co-supervisor of this thesis Jaakko Malmivuo, PhD, for giving me a change to start as a research assistant in the summer of 2000, which later led to this thesis. My supervisor Jari Hyttinen, PhD, provided me with invaluable guidance during the course of this study. His expertise in FDM and models gave insight in seeing the wholeness of the visualization process, and for that I am very grateful. I owe my skills in programming to Noriyuki Takano, LicTech, who tirelessly explained the art of software engineering and how to make the most of the resources. *Doumo arigatou gozaimashita*. I also wish to thank my colleagues at the institute for the laid-back motivation boosting conversations we have had.

A very special thanks to my cousins in the new continent, Jenna Tarkela, BA, and Mika Tarkela for proofreading this thesis. To my dear girl friend Minna my profound appreciation for her efforts towards the completion of this thesis, and endless support. Finally my deepest gratitude especially to my parents; and all the family for their support and care from the beginning.

Tampere, May, 2002

Atte Joutsen

Vaajakatu 5 J 193

33720 Tampere

+358503270314

Table of Contents

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

Ragnar Granit Institute

Joutsen, Atte: Volume Visualization of Calculated Bioelectric Fields

Master of Science Thesis, 81 pages, CR-ROM

Examiners: Prof. Jari Hyttinen, Prof. Jaakko Malmivuo

Funding: Ragnar Granit Institute

June, 2002

Keywords: visualization, file converter, modelling, FDM

Visualization is a very useful tool in our modern information-laden society. It can be applied in various fields to bring light into masses of otherwise incomprehensible data. In this Master of Science thesis the visualization process of bioelectrical fields is examined and improved. The process starts by acquiring magnetic resonance images (MRI) of patients. The gray-scale images are segmented and a computer model is formed of the patient's body. The bioelectric fields are solved from the model using finite different method (FDM) applied in Noname Bioelectric Field Software (NBFS). FDM gives electric potentials in all points of the volume. From these potentials current vectors and potential gradients can be calculated. These quantities and the model anatomy are visualized using Elmer Post.

Two different programs are rarely compatible, thus there was a need for a converter between NBFS and Elmer Post. A file conversion program was developed using a high-level Python programming language. The Converter from NBFS to Elmer (Cone) operates as a Python script file. Cone reads an input file, queries specifications about the output file from the user, forms elements for surface and volume visualization, and finally writes an

output file, which can be directly opened in Elmer Post. The visualized results showed functional surface creation and field variable presentation, enabling researchers to make deductions about the simulated bioelectric field phenomena.

Tiivistelmä

TAMPEREEN TEKNILLINEN KORKEAKOULU

Sähkötekniikan osasto

Ragnar Granit instituutti

Joutsen, Atte: Laskennallisten biosähköisten kenttien volyymivisualisointi

Diplomityö, 81 sivua, CD-ROM

Tarkastajat: Prof. Jari Hyttinen, Prof. Jaakko Malmivuo

Rahoittaja: Ragnar Granit instituutti

Kesäkuu, 2002

Avainsanat: visualisointi, tiedostomuuntaja, mallinnus, FDM

Visualisointi on hyvin käytännöllinen työkalu nykyisessä informaationtäyteisessä yhteiskunnassa. Sitä voidaan soveltaa monilla aloilla valottamaan hankalasti ymmärrettäviä tietomääriä. Tässä diplomityössä tarkastellaan ja täydennetään biosähköisten kenttien visualisointiprosessia, joka alkaa potilaan magneettiresonanssikuvauksesta. Harmaasävykuvat segmentoidaan ja potilaan kehosta rakennetaan tietokonemalli. Biosähköiset kentät ratkaistaan mallista Noname Bioelectric Field Software:lla (NBFS), joka käyttää finiitti differenssi menetelmää (FDM). FDM antaa sähköiset potentiaalit volyymin jokaisessa pisteessä. Näistä potentiaaleista voidaan laskea virta- ja gradienttipotentiaalivektorit, joita yhdessä mallin anatomian kanssa visualisoidaan Elmer Post:ssa.

Kaksi eri ohjelmaa ovat harvoin yhteensopivia, joten NBFS:n ja Elmer Post:n välille piti suunnitella muuntaja. Korkeatasoista Python-ohjelmointikieltä käytettiin Cone-tiedostomuunto-ohjelman kehittämiseen. Cone toimii juontotiedostona Python-tulkille. Cone lukee syötetiedoston, kysyy käyttäjältä täsmennyksiä tulostetiedostosta, rakentaa elementtejä pinta- ja volyymivisualisointia varten, ja lopulta kirjoittaa tulostetiedoston, joka

voidaan välittömästi avata Elmer Post:ssa. Pinnanmuodostus on toimivaa esitettyjen tulosten perusteella. Kenttämuuttujien visualisointi auttaa tutkijoita tekemään päätelmiä simuloiduista biosähköisistä kentistä.

Nomenclature

$\overline{B}_0$	external magnetic induction
$\overline{B}_1$	resonant frequency magnetic induction
$\overline{c}$	bioelectric lead vector
F	potential
^1H	hydrogen atom
Hz	frequency
μ	attenuation coefficient, nuclear magnetic moment
$\overline{M}$	net magnetization
$\overline{M}_{xy}$	transverse component of magnetization
$\overline{M}_z$	longitudal component
n0…n7	apices of elements
ne	number of elements
nf	total number of degrees of freedom
nn	number of nodes
nt	number of time steps
$\overline{p}$	electric current dipole moment
R, r	resistance
?	spin density
S	signal
T_1, T_2	characteristic decay times
TE	pulse echo time
TR	pulse repetition time

?	gyro magnetic ratio
V	voltage
f	phase
?	angular frequency
x, y, z	three independent components of the Cartesic coordinate system.
2D	two Dimensional
3D	three Dimensional
BEM	Boundary Element method
CAD	Computer Assisted Design
Cone	Converter from Noname Bioelectric Field Software to Elmer
CSC	CSC - Scientific Computing Ltd.
CSF	Cerebrospinal Fluid
CT	Computed Tomography
DynAMo	Dynamic Adaptive Modelling
ECG	Electrocardiogram
EEG	Electroencephalogram
FDM	Finite Difference Method
FEM	Finite Element Method
FOV	Field Of View
HTTP	Hyper Text Transfer Protocol
IARD	Image enhancement, Amplitude segmentation, Region growing, Decision trees
IEEE	Institute of Electrical & Electronics Engineers
MRI	Magnetic Resonance Imaging/Image
NMR	Nuclear Magnetic Resonance

NSF	National Science Foundation
TUT	Tampere University of Technology
RAM	Random Access Memory
RF	Radio Frequency
RGI	Ragnar Granit Institute
SIGGRAPH	Special Interest Group on Graphical Display
SGI	Silicon Graphics
VCMT	Volume Conductor Modelling Tools
VRC	Virtual Reality Center

1. Introduction

This thesis comprises the process of bringing numerical simulated bioelectrical data to images through advanced visualization techniques. The data is calculated from computational models based on the magnetic resonance images (MRI) of patients.

The modern computer world is flooded with information. Satellites, supercomputers, digital sensors and such data acquisition systems acquire, generate and transmit data at prodigious rates. The ever-expanding amount of information available has yielded a need to present the vast data sets in an efficient and comprehensible manner. This is where visualization comes into play. Without effective methods of presenting the information, most of the data would sit unseen on computer discs and drives. With visualization we can extract the important information hidden in the data.

1.1 Introduction to visualization

Visualization is a fairly new form of science. The term scientific visualization was first used in 1987 by B. H. McCormic in NSF article *Visualization in Scientific Computing* [McCormic 1987]. Since then the field has grown rapidly due to new technology and major conferences such as IEEE Visualization and SIGGRAPH.

Visualization is a part of everyday life. Weather maps, market charts and rendered commercials are all applications of visualization. The idea is to present the desired data in the most understandable way to individuals -by taking advantage of the natural abilities of the human vision system. We rely on our vision system heavily and pictures truly tell us more than thousand words.

Visualization is the transformation of data into pictures. It is used in many fields such as science, economics, meteorology and graphic design. In modern medicine visualization has helped to see inside the human body by processing and presenting the information provided by X-ray Computed tomography (CT) and MRI equipment. Both give information about the internal anatomy of a living patient as cross-sectional images. CT-imaging uses a gantry with an X-ray transmitter and receiver that rotates around the patient to acquire different projections while MRI uses 3 magnetic coils and gradient coils with pulsed radio frequency (RF) waves to obtain the data. Mathematical techniques are then used to back project and solve slice-planes of the volume. By giving the numerical matrices gray-scale values the anatomical structure emerges from the numbers. More impressive results can be obtained by using the slices to create 3D structures to view the brain, the heart, or skeletal system of a patient.

When rendering an object the surfaces of the object and their interactions with light are determined and projected to a 2D view that is visible to the camera. 3D rendering can be split into surface and volume rendering. In surface rendering only the surface is employed assuming that it is opaque thereby leaving the interior undepicted. Volume rendering allows us to see the inhomogeneity inside the matter by calculating the interaction of light rays with the interior.

Common visualization techniques are used every day. Different pie and bar charts and graphs describe finances and stock markets. 1- to n-dimensional plots illustrate functions, and maps convey for example height information through colors. More sophisticated techniques such as 3D surfaces, solids, isosurfaces, slices, translucency, stereopsis, and animation

transmit significant information even more efficiently. A good visualization conveys the important features of the data in an unambiguous and distinct manner and omits everything nonessential. It must be faultless for sufficient quantitative evaluation *i.e.* it must obey the Lie factor rule. Visualizations must not offend viewer's senses as moiré patterns do, and they must be adjustable to serve multiple needs.

1.2 The goals of this study

This visualization project is a part of bioelectrical research projects conducted at Ragnar Granit Institute (RGI) in Tampere University of Technology (TUT). First MR images are taken of a patient. The resulting image slices are segmented to separate different tissues from others. From the segmented material a finite difference method (FDM) body model consisting of voxels is formed. This is done by regarding the slice stack itself as the third dimension and calculating a resistor network between nodes placed in the apices of the voxels. Takano has developed a computer application, Noname Bioelectric Field Software (NBFS) [Takano 2001], to reduce the computational load required in electrocardiogram (ECG) simulations in forward problem. NBFS uses FDM to solve the forward problem.

As a result of calculations NBFS produces numerical data, which consists of scalar electrical potentials, gradient potential vectors and current vectors at the nodes. When this nodal information is combined with data of the model *i.e.* tissues codes, coordinates of nodes, etc., one output file of all the nodes of thorax can contain hundreds of thousands of nodes resulting in a text file of tens of megabytes. This data is not in an informative format for an individual; therefore visualization is applied to view the results.

The visualization software of choice was Elmer, developed at CSC - Scientific Computing Ltd. (CSC) in Espoo. Elmer is a computational tool designed for multi-physics problems. It includes physical models of fluid dynamics, structural mechanics, electromagnetics and heat transfer. These

are described by partial differential equations, which are solved using the finite element method (FEM). Elmer is not a one large program, but modular software comprising of three different parts. Geometry, boundary conditions and physical models are defined in Elmer Front. The resulting problem definition is solved by Elmer Solver. Finally the results are visualized by Elmer Post. These programs work like a pipeline but any of the three parts can function individually for processing inputs of other programs.

The goal of this study is to develop a converter program using a high level programming language, Python, to convert the output files of NBFS to input files of the third part of the Elmer package, Elmer Post, which is then utilized to produce images of the simulation results. Visualizing potential divisions and current fluxes of the calculations will help in understanding what is happening inside and on the surface of the body during the heart's activation cycle.

2. Material

The motive of this research was to form a connection between two key elements in the visualization process. The available data for visualization had to be converted to a form accepted by the visualization computer program. In this case visualization is used to bring different scalar and vector fields of bioelectromagnetism into images.

2.1 Introduction to material

The whole process, which ends in completed visualizations, begins with acquisition of data. This data can be anything depending on the matter, but in the research conducted at RGI, it is the anatomical data of patients. The data is usually attained by making cross section images of the patient using of MRI or CT. 2D gray-scale CT or MRI images are segmented *i.e.* every pixel is labelled as some predefined tissue such as heart muscle or lung. Those segmented slices are used to construct a 3D resistor mesh model of the body. Different tissues represent different conductivities in the resistor mesh. Sources and sinks are placed into the model and FDM equations are formed. Both the model and the equations are fed to FDM solver, which calculates potentials at every node of the model. From those solved potentials gradient potentials and current vectors can be obtained as

well. These field variables with source and sink potentials are the subjects of visualization.

Python was chosen as the programming language for this file conversion project because of its high-level nature favouring beginners but still having efficient tools for program development. Because Python is an Open Source language it was very well documented in the Internet, therefore useful solutions to problems that came up during the development of Converter from NBFS to Elmer (Cone) were quite easy to find. Python's versatility is also an advantage in further program development and implementations in other environments.

Cone was used to transform data from NBFS to Elmer. Most visualization techniques require some surface or volume to be introduced to the visualization software for certain visualization types, thus Cone also produces 3 different types of elements of the nodes inputted to it. Elmer reads the files written by Cone and after that the user can freely visualize the features of his/her interest. The name 'Cone' was chosen for the program for its double meaning. First, it is an abbreviation of its main function, and second, it has the same function as a cone shaped funnel – to facilitate input of some matter from one object into another.

2.2 Visualization

Visualisation is defined in Webster's Ninth New Collegiate Dictionary, as *the process of interpreting in visual terms or of putting into visual form*. Visualization is often confused with imaging and computer graphics. By definition imaging or image processing is the study of 2D images. It includes transformations, information extraction, image enhancement and analysis. Computer graphics is creation of images using a computer. This includes 2D drawing and 3D rendering techniques. Visualization is exploring, transforming and viewing data as images or other sensory forms to gain understanding of the data. It is an interactive process of turning data into comprehension where the user is the last link in the chain. This includes both

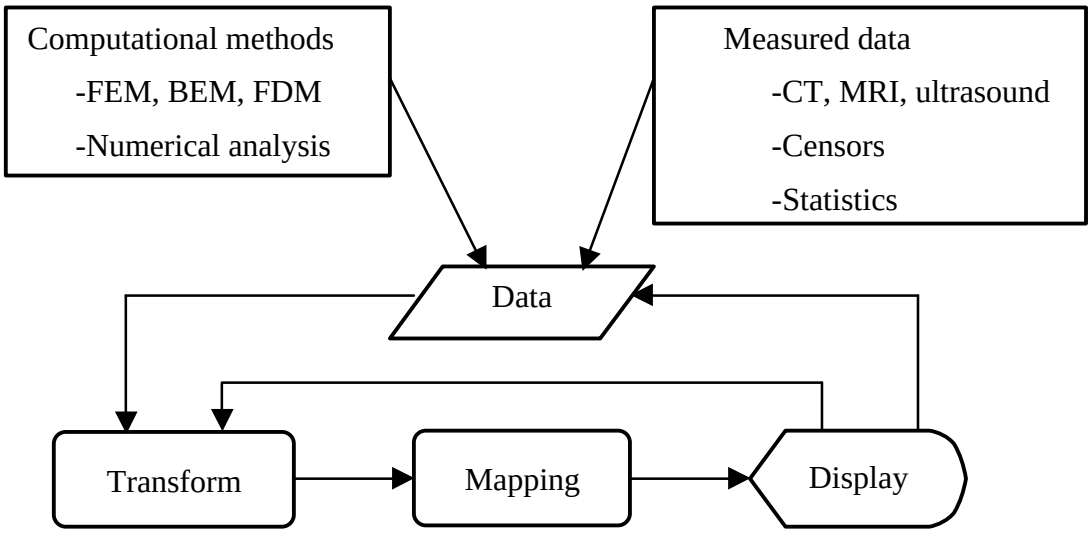

Figure 2-1 Visualization process [Schroerder et al. 1998]

imaging and computer graphics as well as data processing and filtering, user interface methodology, computational techniques and software design.

The visualization process focuses on data (Figure 2-1). Data is acquired from sources, which can be computational methods or some measured data. The raw data is then transformed to suit the process. This can be anything from data conversion to image enhancement to data processing. Then the transferred data is mapped using appropriate presentation to the user. This means making potential maps of discrete scalar potentials or setting three components of vector to one and coloring it according to its length; in short, presenting the data in the most appropriate form to the user. Finally, the data is rendered completing the process. The process can be repeated, as the data is better understood. Novel data can be introduced or the same data can be represented in some different manner revealing other points of interest. This is referred to as analysis steering. It is a fundamental part of visualization, because it enhances the interactivity of the overall process.

2.3 Bioelectromagnetism

All living beings consist of cells. Cells are specialized in their anatomy and physiology to perform different tasks. All cells exhibit a voltage difference across their cell membrane, but only nerve and muscle cells are

excitable. Their cell membrane can produce electrochemical impulses and conduct them along the membrane. This is the genesis of bioelectromagnetic signals. The nervous system, which consists of nerve cells, receives, processes, and transmits information to muscle systems, consisting of muscle cells, making them contract. All these actions produce innumerable action potentials, which are the results of cell excitations. When all these action potentials cumulate temporally a varying bioelectromagnetic signal known as an ECG (~$10^{-3}V$) or electroencephalogram (EEG) (~$10^{-6}V$) can be acquired.

Bioelectromagnetism is a discipline that examines the electric, electromagnetic and magnetic phenomena, which take place in biological tissues. These phenomena include:

- The behavior of excitable tissue (signal sources)
- The electric currents and potentials in the volume conductor (the body)
- The magnetic field at and beyond the body
- The response of excitable cells to electric and magnetic field stimulation
- The intrinsic electric and magnetic properties of the tissue

Bioelectromagnetism has a close relation to electromagnetism, because until the middle of the nineteenth century, the history of electromagnetism was also the history of bioelectromagnetism. This can be easily understood because most of the electromagnetic phenomena observed in that time were taking place in living tissue.

The taxonomy of bioelectromagnetism (Figure 2-2) is governed by two cornerstone concepts of science: Maxwell's equations and the principle reciprocity. Maxwell's equations connect time varying electric fields to magnetic fields such that bioelectric fields implicate the presence of biomagnetic fields and vice versa. This connects the 3 horizontal subdivisions (A, B, and C) together. The vertical connector between measurements and medium properties (I, II, and III) is done by the principle of reciprocity, which states that the sensitivity distribution in the detection

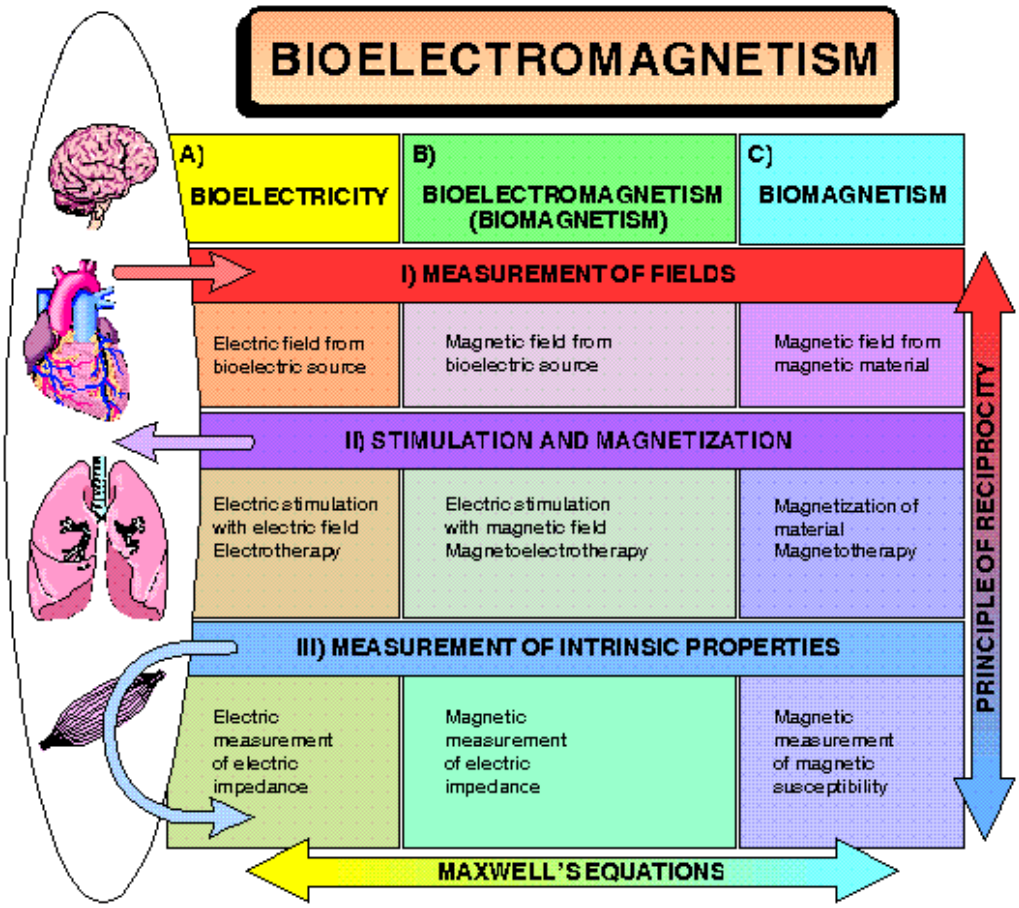

Figure 2-2 Organization of bioelectromagnetism into its subdivisions [Malmivuo et al. 1995]

bioelectric signals, the energy distribution in electric stimulation, and the sensitivity distribution of electric impedance measurements are the same [Malmivuo et al. 1995].

The bioelectromagnetic phenomena are studied by making *in vivo* and *in vitro* measurements and observations. One popular method to bring light to the occurrences in living tissue is to construct a model. Basic properties of the subject at hand are modeled and the results obtained from the model are compared with measurements from the living subject. From that deductions are made and as knowledge of the subject grows the model can be enhanced. Models can be divided into physical (tanks models, analog models) and *in silico i.e.* computer models. Nowadays *in silico* models have superseded cumbersome and impractical physical models with their superior precision and versatility.

The activity in living tissue produced by excitable cells can be measured with sophisticated equipment. The brain consists of 10^{10} to 10^{11} interconnected nerve cells. All these convey information in the form of action potentials, which are the result of a complex ion exchange through the cell's membrane, propagating in the cells' branches. In the heart a group of specialized cells spontaneously produce an action potential that starts to propagate to heart muscle tissue first contracting the muscle around the atriums and then the ventricles producing a pumping action. When models of these kinds of systems are constructed the bioelectromagnetic phenomena are reduced to simple sources inside the heart or brain or any tissue under investigation. The sources can be sets of electrical dipoles (one dipole, moving dipole, multiple dipole, multipole or double layer) producing an electric field throughout the volume conductor. The volume conductor can be simple homogenous infinite or finite, or more complex inhomogeneous infinite or realistically shaped finite model with components of appropriate resistivity representing organs. During the visualization process (Figure 2-1) the scalar potentials are solved from the inhomogeneous finite model.

2.3.1 Forward and inverse problem

Problems in bioelectromagnetism can be divided into two sections, these are forward and inverse problems. Both problems treat the relations of bioelectromagnetic sources, volume conductors and bioelectromagnetic fields (Figure 2-3). In the forward problem the source and the conducting medium are known and from these the resulting field is solved. The forward problem has a unique solution that is only limited by the accuracy of the source and the conductor. *In vivo* this is easily obtained by simply measuring the field as ECG or EEG. In the inverse problem the field and the conductor are known and from those the source must be solved. This has also clinical importance, as physicians need to know the source of the bioelectromagnetic phenomena to localize epileptical foci or cardiac ischemia to make diagnostic decisions. The inverse problem, however, does not have a unique solution. This is due to the discrete locations of measurements. All of the field cannot be measured, but a sample. Also noise

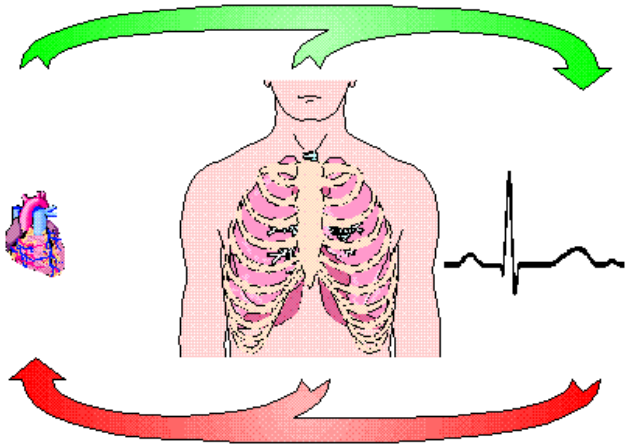

Figure 2-3 Forward and inverse problem in electrocardiology [Malmivuo et al. 1995]. In the forward problem electromagnetic field is solved on the basis of the source and the volume conductor. The inverse problem is the reverse of this operation solving the source from the knowledge of the volume conductor and the measured field.

complicates the solution. Nevertheless, there are methods to overcome this dilemma by limiting the solutions. One of these is using lead field theory in conjunction with modelling.

2.3.2 Lead vector and lead field

Lead field is a formation of lead vectors over every possible source point in the volume conductor. Lead vectors form a unique sensitivity distribution for a particular lead for measuring activity of the sources. Lead vector, lead field and the theorem of reciprocity are applicable in bioelectricity, bioelectromagnetism and biomagnetism alike. Lead vector was described as transfer impedance between the source and the measuring point by Geselowitz in 1971. In other words, it is Ohm's law applied in 3D. The vector is a transfer coefficient of three components in the Cartesic coordinate system for a fixed lead to measure a dipole source (Figure 2-4). The theory can be formulated as:

$$\Phi_P = c_x p_x + c_y p_y + c_z p_z \qquad [\,2\text{-}1\,]$$

where c_x, c_y and c_z are components of the lead vector, p_x, p_y and p_z components of dipole source vector and F_p the potential of the lead measuring the dipole. This can be also rewritten as a scalar product:

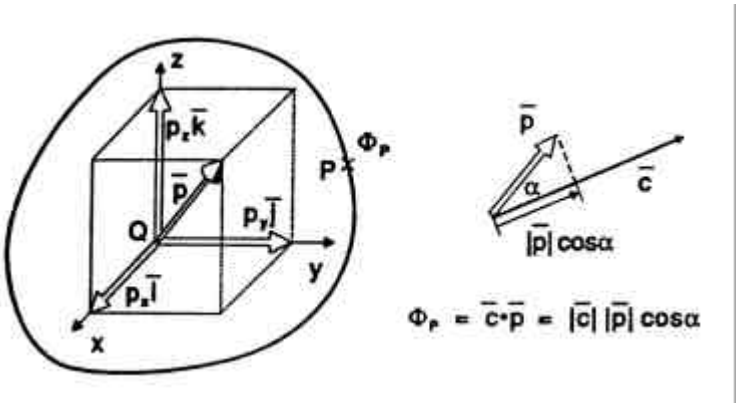

Figure 2-4 Lead vector [Malmivuo et al. 1995]. Left: The lead vector and its components in Cartesian coordinate system. Right: Potential F_P is a scalar product of the lead vector $\bar{c}$ and the dipole $\bar{p}$.

$$\Phi_p = \bar{c} \cdot \bar{p} \qquad [\ 2\text{-}2\ ]$$

where $\bar{c}$ is the lead vector, $\bar{p}$ is the dipole source vector and F_p the potential of the lead.

The lead field is an extension of the idea of the lead vector. Again a fixed lead is assumed and for that the behavior of the lead vector is examined over the entire volume conductor placing dipole sources into every point in the volume and measuring the potential at the lead. This procedure gives the sensitivity distribution for the entire volume. The contribution of all sources to one lead is by the principle of superposition:

$$V_L = \sum_k \bar{c}_k \cdot \bar{p}_k \qquad [\ 2\text{-}3\ ]$$

where V_L is the voltage at the lead, $\bar{c}$ is a lead vector and $\bar{p}$ is a dipole source. In lead vector and lead field theory the volume conductor is assumed to be linear. The principle of reciprocity applies in the lead field such that the lead field is *exactly* the same as the field of current flow when a reciprocal current is applied to the lead (Figure 2-5). Theory of reciprocity was conceived by von Helmholz in 1853, and can be summed as: it is possible to exchange the source and the detector without any change variation in the detected signal.

From the previous the solution of the inverse problem can be adjusted. Presumptions about the activity of the source can be made based on the sensitivity distribution. This approach depends on each lead detecting

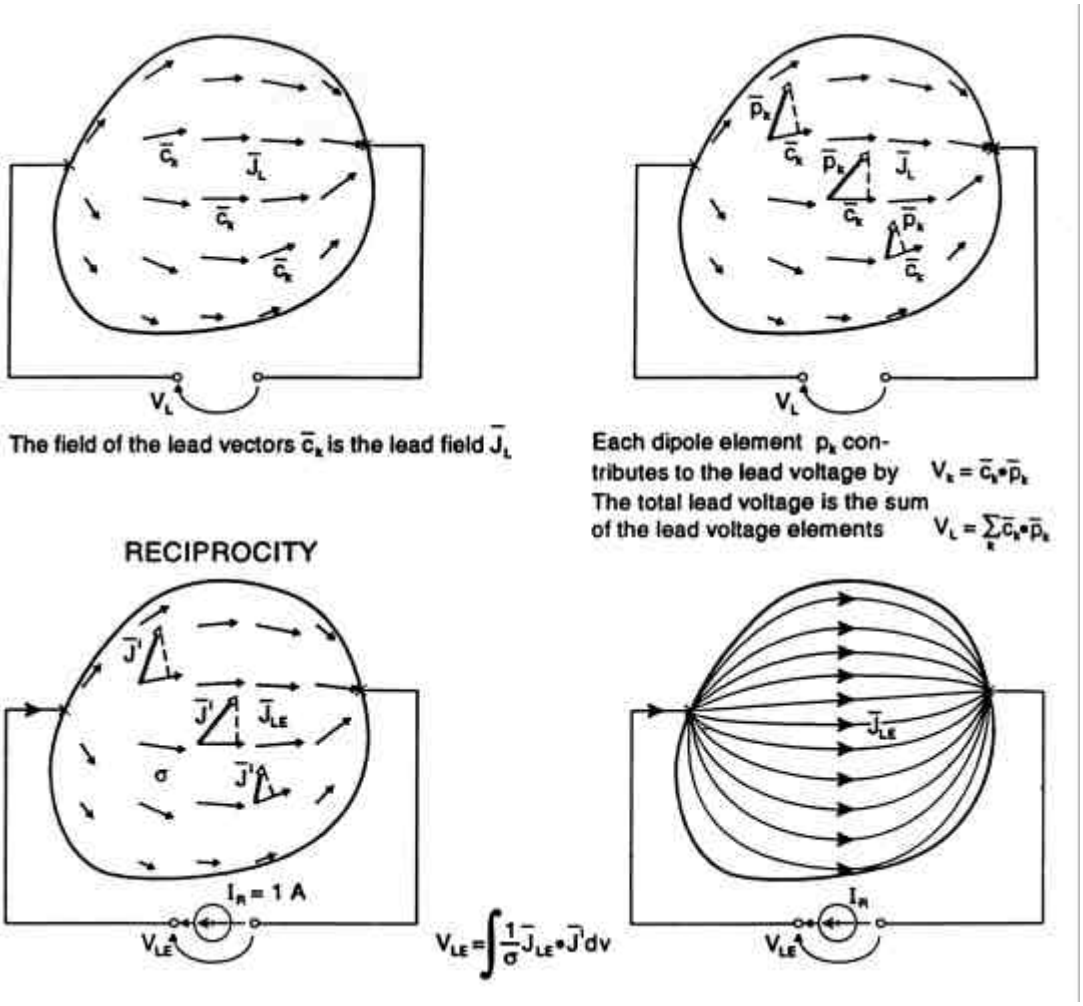

Figure 2-5 Lead field and reciprocity [Malmivuo 1995]. Top left: To form a lead field $\mathbf{J_L}$ *for a fixed electrode pair the behavior of the lead vector* $\bar{c}$ *is examined at every point in the volume. Top right: The total lead voltage* $\mathbf{V_L}$ *is a sum of all the voltage elements derived from dipoles* $\bar{p}_k$ *and corresponding lead vectors* $\bar{c}_k$*. Bottom left: The lead field* $\bar{J}_L$ *is the same as the current field* $\bar{J}_{LE}$ *raised by feeding a reciprocal current* I_R *to the lead. The lead voltage* V_{LE} *due to a volume source is obtained by integrating the scalar product of the lead field current density* $\bar{J}_{LE}$ *and the source density* $\bar{J}^I$ *over the volume conductor. Bottom right: The lead field can also be represented as current flow lines.*

the components of the source dipoles that are pointing to the direction of the lead's sensitivity vector.

2.3.3 Application of lead field theory in modelling

When FDM is used to solve potentials in volume conductor, a resistor network is formed to model the electrical properties of the conductor. The resistor network is piecewise homogenous and linear, thus lead field theory can be applied point wise. The potential distribution of the resistor network must be solved for every source configuration. The possible source location

equals the number of nodes in the network, whereas the measurements are limited only to the electrode locations. Using reciprocity the source, and the measurement locations may be swapped without affecting the result. A reciprocal current is fed to the electrodes, which flows through the network creating potentials at every node. From the knowledge of the resistors and currents lead vectors can be calculated for node locations. The length of the current vector is directly proportional to the length of the lead vector, while the direction is identical. The magnitude of the lead vector describes the effect of the dipole to the measurement lead under investigation. A 3D array of lead vectors forms a lied field for the electrode pair. The current filed and the node potentials in the volume can be visualized. Because the current vector and lead vector relate to each other, the visualization of the current vector field is also the visualization of the lead vector field multiplied with some coefficient relating to the input current, assuming that the spacing in the model grid is equal i.e. every resistor describes an equal volume.

2.4 Medical imaging methods

MRI and CT are computerized imaging methods in which first the information is gathered from a patient by using RF electromagnetic waves or radiation respectively. The obtained mass of information is converted to digital form and processed using back projection algorithms by which cross sectional slice planes of the patient are solved. The planes are traditionally presented with shades of grey ranging from white to black. The medical staff can then view them on computer screens or illumination boards.

2.4.1 MRI

MRI is an efficient means to produce cross section images of the human body with a superior soft tissue contrast compared to other imaging methods available. The concept of using nuclear magnetic resonance (NMR) to detect tumours in patients was first proposed by Raymond Damadian in 1972 [Damanian 1972]. A year later Lauterbur formulated the fundamental imaging concept used widely by MRI equipment.

NMR is a quantum mechanical phenomenon occurring in atoms that have an odd number of protons or electrons. Such atoms have a nonzero nuclear magnetic moment μ. Because of that moment the atoms will precess about an external magnetic field $\overline{B}_0$ with a frequency of

$$? _0 = ? \overline{B}_0 \qquad [\ 2\text{-}4\ ]$$

where $?$ is the gyro magnetic ratio and $? _0$ the resonant frequency. Many NMR prone isotopes can be used in MRI but ^1H is the most conventional for its high inherent sensitivity and abundance in living tissue. In human body the strongest NMR signals are obtained of ^1H in water and fat.

When placed in $\overline{B}_0$ ^1H nuclei align their spins either parallel or antiparallel to $\overline{B}_0$. A slight excess of the nuclei is in lower energy parallel state and creates a net magnetization $\overline{M}$. The precessing $\overline{M}$ can be rotated out the equilibrium by exposing the tissue to a changing magnetic field B_1 using RF energy at the resonant frequency $? _0$. By selecting the right parameters rotation angles or 90° and 180° are achieved. Following the excitation the transverse component $\overline{M}_{xy}$ and the longitudinal component $\overline{M}_z$ of magnetization return to equilibrium state. During the excited state and the relaxation a current is inducted in a coil of wire around the patient due to the changing magnetic flux created by the precessing nuclei. This signal is called the free induction decay (FID) (Figure 2-6). The characteristic decay time for the $\overline{M}_{xy}$ is T_2 and for the $\overline{M}_z$ T_1. These in addition to proton density in the tissue influence the received signal. The signal is proportional to:

$$Sar \cdot \left(1 - e^{-\frac{TR}{T1}}\right) \cdot e^{-\frac{TE}{T2}} \qquad [\ 2\text{-}5\ ]$$

where TR and TE are the pulse repetition and echo time of the pulse respectively and $?$ is the spin density. Optimum image contrast depends on TR and TE timing in order to produce $T1$, $T2$ weighted or proton density images. In liquids the local magnetic fields respond rapidly; they have less spin-spin interferences and dephasing is relatively slow, thus their $T2$ times

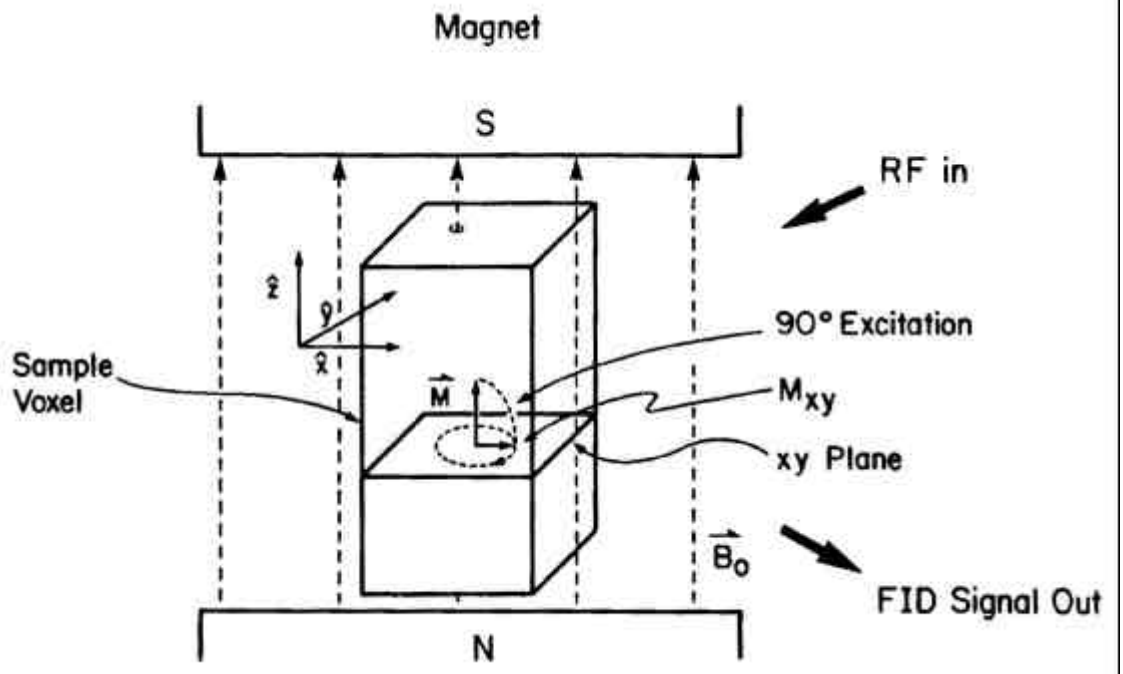

Figure 2-6 The principle of MR-imaging [Webster 1988]. The protons in a sample tissue align with the magnetic field $\overline{B}_0$ to produce a net magnetization $\overline{M}$ in the z-direction. A 90° RF pulse rotates $\overline{M}$ into the xy-plane. $\overline{M}_{xy}$ precesses about the $\overline{B}_0$ direction at the resonant frequency due to the torque on $\overline{M}$ caused by $\overline{B}_0$. A FID-signal is inducted in a coil around the sample due to the changing magnetic flux.

are consequently longer. Solids have a fixed molecular structure and neighbouring protons strongly influence FID phase relationships producing a shorter *T2*.

The T1 weighted images show dark cerebrospinal fluid (CSF) and the gray matter is darker than the white matter. In the T2 weighted images the CSF has a higher signal than the gray or white matter (Figure 2-7). This is ideal for visualizing CSF or oedema. Considerable contrast between different tissue types enables precise segmentation and labelling. A detailed segmentation is vital when models of the body are developed.

The MR image is achieved by spatially encoding the signal with frequency and phase to determine its source in space. Because the resonance frequency depends on the applied magnetic field, a linear gradient field created across the patient will make the ^1H nuclei precess at different frequencies and enable a linear relationship between frequency and spatial location. By using a narrow band RF pulse matching the frequency of

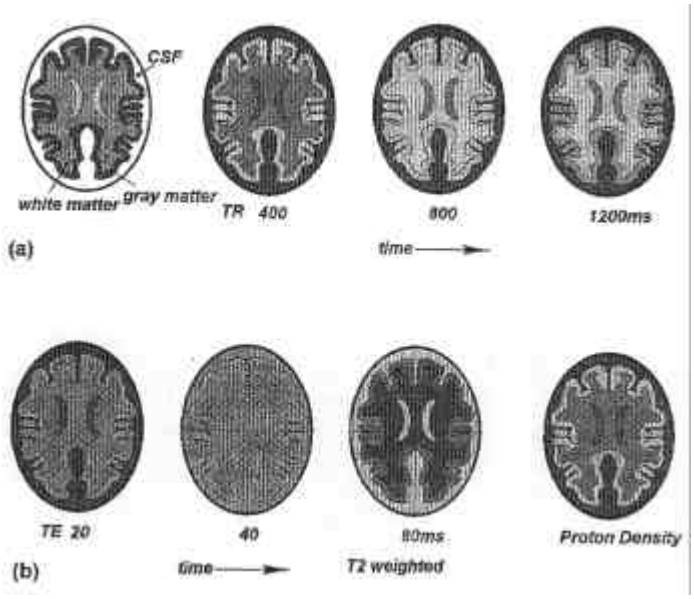

Figure 2-7 T1 (a) and T2 (b) weighted MR-images [Dowsett et al. 1988]. Different repetition and echo times produce different results of the same subject.

the z-gradient in a wanted region, an exact cross-section of the patient is excited. In that section x-gradient relates frequency to x-position and y-gradient phase shift to y-position for exact slice. A 2D Fourier transform is applied to the complex signal detected by the receiver coil to locate the signal source voxels by separating them on the basis of received signals frequency and phase (Figure 2-8). A second Fourier transform yields signal strength information, which is stored as a gray scale value in a pixel display matrix.

The raw data consists most often of 256x256 data points with 16 bits of amplitude resolution. The signal strength calculation cuts the amplitude information down to 15 bits of resolution. The magnitude data is expanded to

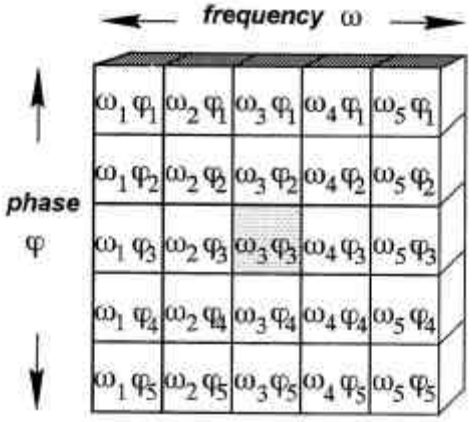

Figure 2-8 The MRI slice plane matrix obtained with Fast Fourier Transform (FFT) of the FID-signal [Dowsett et al. 1988]. Each voxel has a unique pair of phase f and frequency ? .

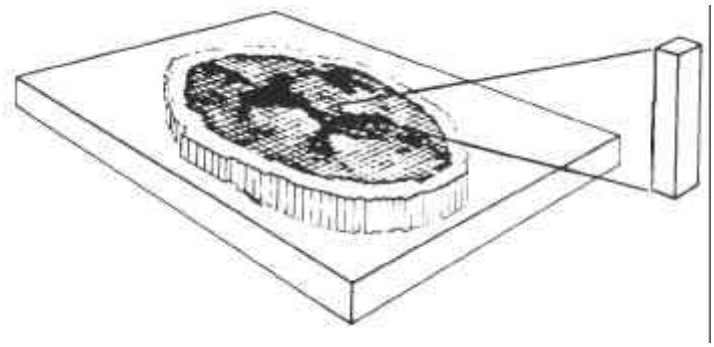

Figure 2-9 MR-image [Webster 1988]. The image consists of an array of pixels that represent the NMR properties of a group of volume elements, voxels, within the body of the patient.

a data matrix of desired size by either pixel interpolation or pixel replication. The image is typically displayed with an eight bit video display. This means there are 256 possible gray levels with which to display the 32768 possible data values from the 15 bits of magnitude information. A linear look-up table is typically used to assign contrast and brightness to the image by selecting size and a middle value of a window in the image histogram.

The matrix dimension *m*, and the field of view (FOV) with the slice thickness determine the individual voxel size. For a 50*cm* FOV, which is a typical value, occupying a 512^2 matrix and a 2*mm* slice thickness, the voxel size would be (FOV/*m*) approximately 1mm^2 by 2 *mm* deep (Figure 2-9).

2.4.2 CT

CT was developed from conventional X-ray imaging by replacing the film and the X-ray tube with a gantry fitted with an X-ray transmitter and a fan of detectors to get high contrast volume information of the patient as opposed to lower contrast regular X-ray images. The gantry rotates about a patient taking multiple projections of the patient and calculating the cross-sections there after. The first CT machine was built by Hounsfield in 1973.

There are 5 generations of CT machines all utilizing a different number of emitters and receivers. The 3rd generation scanners, which utilize a large fan array of detectors and one emitter completely encompassing the patient, are the most common. The X-ray tube is quite similar to general radiographic tubes with a tube potential ranging from 80 to 140*kV*. The

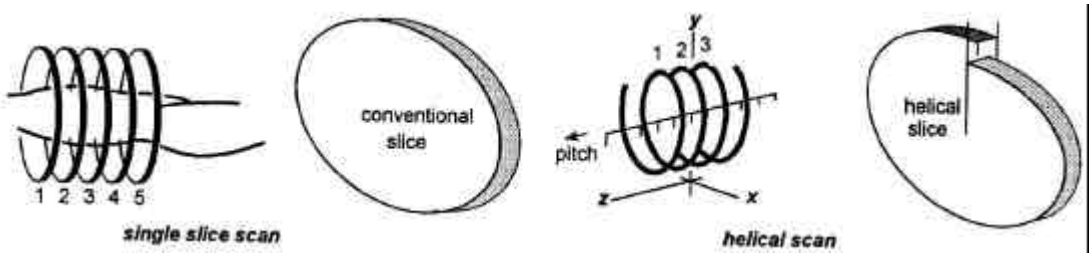

Figure 2-10 Scanner patterns [Dowsett et al. 1988]. Left: Incontinuous separate slice acquisition and the resulting complete slice periphery. Right: Spiral continuous acquisition gives incomplete slice periphery.

detectors are one-dimensional efficient and fast photon counters, usually xenon gas filled chambers.

The scanner moves in a single slice or helical slice pattern 360° around the patient taking projections (Figure 2-10). 500 projections are sufficient for a 256x256 matrix with 1*mm* resolution and 1000 projections for 512x512 matrix with 0,5*mm* resolution. A high resolution can only be achieved using a thin slice. In data collection only ray-sums, which relate to the total attenuation coefficient of the volume through which the ray has traversed, are obtained (Figure 2-11). From attenuation figures a matrix with individual coefficients using back projection algorithm is solved. The matrix is high pass filtered to remove low frequency starburst patterns.

The absorption coefficient μ depends on the *kV*. The μ of the tissue is related to that of water at the same *kV* to obtain a CT number insensitive to *kV*. The CT number range goes roughly from –1000 (air) to 0 (water) to 3000 (bone). The solved matrix is an array of CT-numbers, which is a

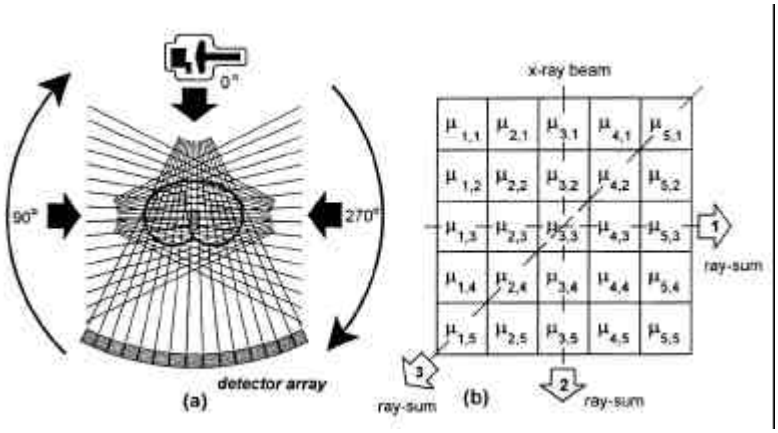

Figure 2-11 Projections and the resulting matrix [Dowsett et al. 1988]. Leftt: Image matrix projections for a fan-beam showing construction of matrix cells during fan-beam rotation. Right: A small section of the final matrix showing individual attenuation values combined as a ray-sum.

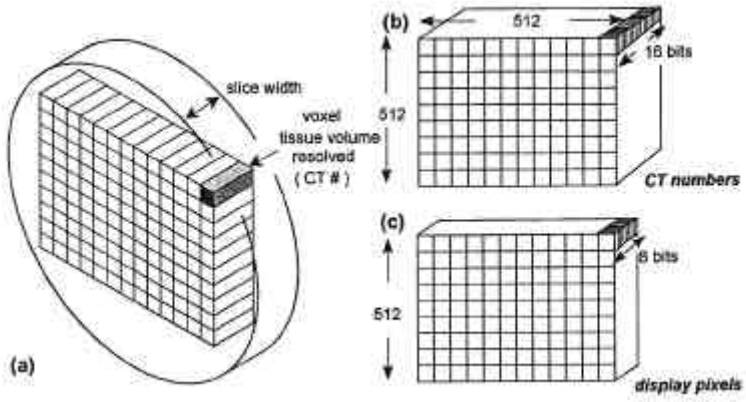

Figure 2-12 A slice plane represented as CT numbers. The voxel information is stored in memory as a 512x512 matrix with a 12 to 16 bit precision. The displayed pixels are usually presented with 8 bits.

volume slice element or a voxel. The primary storage matrix for the computed CT numbers must be able to handle the whole value range and a +/- sign, which requires a 12+1 bit representation. Usually two bytes are used. The entire CT information cannot be displayed simultaneously since video monitors have a limited dynamic range because of 8-bit D/A converters enabling 256 gray levels (Figure 2-12). Windowing is used to select a section of the –1000 to 3000 range and present there values by 8 bits. Greater contrast can be obtained from narrower windows. Each CT number is referenced to a look up table in the computer display memory.

Slice thickness varies between 1*mm* and 5*mm*. Soft tissue contrast is lost if the slices are too thin but angular structures in thick slices cause volume artifact. Resolution enables differentiating high contrast objects close to each other. The size of the display matrix and FOV influences the resolution. The size of the display pixel can limit image resolution if it is larger than the intrinsic resolution. Typical values of display pixel size are 0,15*mm* for a 512x512 matrix representing a restricted FOV of 7,5*cm* in a head scan and 0,3*mm* for a 512x512 matrix with a 15*cm* FOV in a body scan.

2.5 Segmentation

Segmentation is the process of extracting desired objects from a background in an image or a volume. Image segmentation is useful in many applications. From the segmentation results, it is possible to identify regions

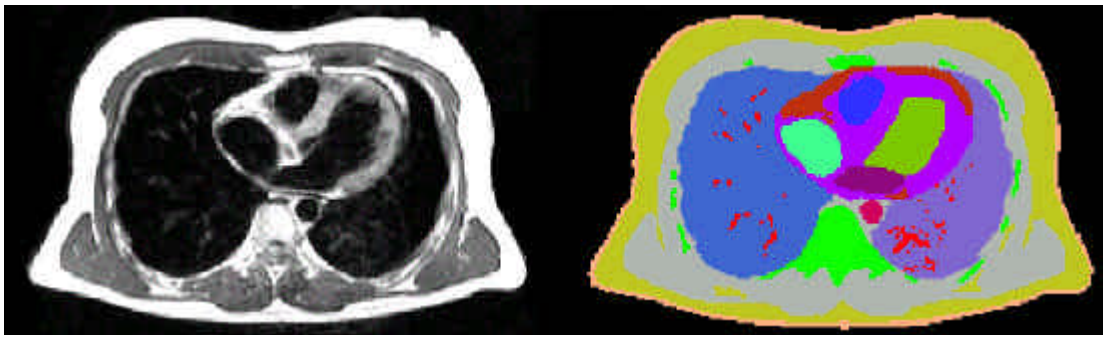

Figure 2-13 A gray-scale MR-image of the thorax of a patient has been segmented. Different colors define different tissues within the thorax. In further processing the tissues are given properties to produce a piecewise homogenous linear volume conductor model.

of interest and objects in the scene, which is very beneficial considering subsequent image analysis (Figure 2-13).

Segmented MR-images can be rendered in 3D enabling efficient analysis of the MRI data, as anatomical structures are more easily understood than just studying 2D slices. Visualization can be applied in 3D volume visualization and 3D surface reconstruction. With the knowledge of the tissues one can decide to visualize only the regions of interest and make the other tissues partly opaque or transparent enabling a look inside the volume. Volumetric analysis can be made of anatomical disorders alleviating treatment planning. Reconstructed surfaces can be color mapped with measured or calculated data producing efficient multimodal visualizations.

The most simple segmentation method is to manually draw the desired borders directly onto the raw image. This would take too much time and be prone to errors, especially due to fatigue. To alleviate the workload a variety of semi-automatic and automatic computer segmentation methods have been developed. In biomedical engineering the material is usually MRI cross-sections with 256x256 resolution. Every pixel is a gray-scale value presented with 8 bits enabling values from 0 to 255.

Thresholding is a method of separating tissues by similar pixel intensity. This can be done by selecting the coefficients manually or obtaining them directly from the histogram. The image is scanned pixel by pixel and pixels belonging to some defined intensity range are labeled. Edge-based methods center around contour detection. Contours between different areas are detected and the enclosed areas are defined as homogenous tissue. In region-based

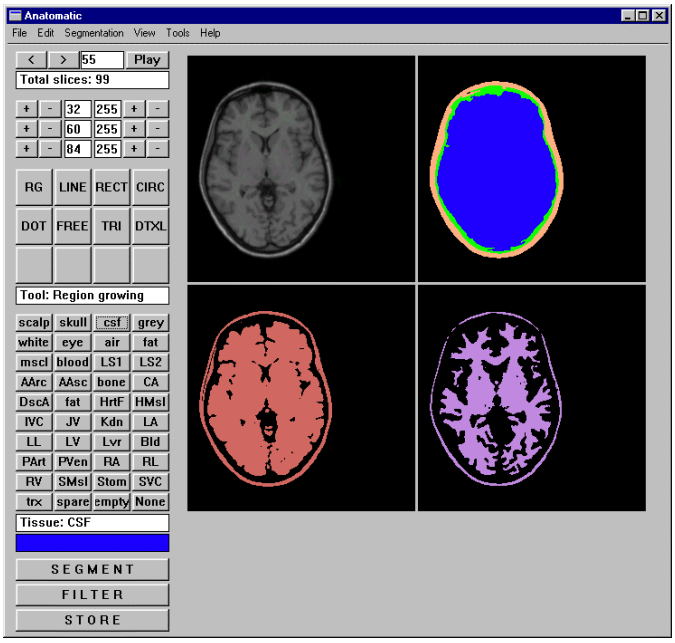

Figure 2-14 Graphical user interface of Anatomatic segmentation software. An MR-image of the head of a patient has been segmented for further processing. Different tissues e.g. scalp, skull, cerebrospinal fluid, gray- and white matter in the head are clearly presented using different colors.

methods the image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels. Adjacent regions are then merged under some criterion involving perhaps homogeneity or sharpness of region boundaries. More elaborate methods are connectivity-preserving relaxation methods such as active contour model, where the main idea is to start with some initial boundary shape represented in the form of spline curve, and iteratively modify it by applying various shrink/expansion operations according to some energy function. Other connectivity-preserving methods are stochastic model based approaches, morphological watershed based region growing, energy diffusion, and graph partitioning. Quantitative evaluation methods have also been suggested.

Due to the difficult nature of the problem, there are few automatic algorithms that can work well on large and detailed data such as the human anatomy. The problem of segmentation is difficult because of image texture. If an image contains only homogeneous color regions, clustering methods in color space are sufficient to handle the problem. Blurring and random noise in the images complicates the situation for automated methods. This can lead to false decisions in the segmentation, which might lead to false results in the further processing. Nowadays the best method to segment medical

images is semi-automatic, where the user gives parameters to the program and supervises the process.

Anatomatic [Heinonen et al. 1998] segmentation software has been developed by Heinonen at RGI. The software operates semi-automatically and utilizes the *IARD* (Image enhancement, Amplitude segmentation, Region growing, Decision trees) [Heinonen et al. 1997] segmentation algorithm. The source data can be any 3D medical images, such as MR images and CT images. Anatomatic was used in processing the MR-images (Figure 2-14) for building a model of the human body later to be used in FDM simulation to solve potentials in the volume generated by heart action.

2.6 Finite Difference Method

Finite element methods have been used for numerous applications in different fields of engineering since the use of computer models has become popular. Finite element methods can be split to 3 different methods. These are finite difference method (FDM), finite element method (FEM) and boundary element method (BEM).

FDM is the easiest and the most straightforward method for implementing complicated structures of the human body in detail and it gives solutions to all the elements of the model. The increase in the accuracy (i.e. the number of elements in the model) only increases the number of equations, not the complexity of the equations, thus inhomogeneities and complex geometry can be easily modelled. At RGI, FDM has been applied to solve electric fields in realistic human body models [Hyttinen 1994][Kauppinen 1999][Laarne 2000]. At the relatively low frequencies of bioelectric signals the human body can be considered a resistive system and, because of segmentation, also piecewise homogeneous. The potential at a mesh node is expressed in terms of the potentials of the neighbouring nodes by the Poisson's equation, which is the governing equation of the electrical properties of the body as a volume conductor. In the orthogonal Cartesian coordinate system the equation is:

$$\frac{\partial}{\partial x}\left(s_x \frac{\partial \Phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(s_y \frac{\partial \Phi}{\partial y}\right)+\frac{\partial}{\partial z}\left(s_z \frac{\partial \Phi}{\partial z}\right)=-I_v(x,y,z) \qquad [2\text{-}6]$$

where F is potential and s_x, s_y and s_x are the conductivities in the x, y and z directions respectively and $-I_v(x,y,z)$ is the representation of the source currents in the volume conductor. Computational elements for the FDM are formed by first dividing the volume conductor to an array of elements representing the human body geometry and having properties of the tissue type assigned to it. This is done when different tissues are separated from MRI or CT images. The geometry of the element array is defined by the nodes in the corners of the blocks. These nodes are connected to each other and a resistance value for a connector between two nodes is calculated from the conductivities and distances of the nodes (Figure 2-15). Ohm's and Kirchov's laws are applied to this network of resistors to solve potentials at every node. The potential at a node is presented as function of its neighbours' potentials by:

$$f_n=\left(\frac{1}{r_a}+\frac{1}{r_b}+\ldots+\frac{1}{r_f}\right)^{-1}\left(\frac{f_a}{r_a}+\frac{f_a}{r_b}+\ldots+\frac{f_a}{r_f}\right) \qquad [2\text{-}7]$$

where r_a to r_f are the resistances between node n and the neighboring nodes and $?_a$ to $?_f$ the potentials of the respective nodes. If node n is a source node then $?_n$ is the potential of the source (Figure 2-16). To solve potentials

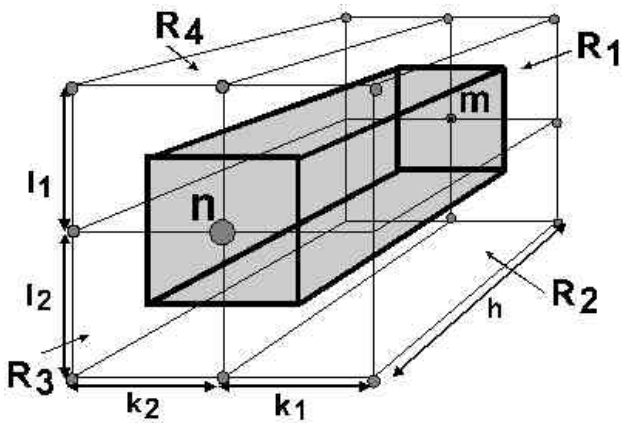

Figure 2-15 FDM computational element. Edge resistors are placed to the edges of the voxels to connect nodes with a resistor value inherit to the voxel. Edge resistors are formed of the knowledge of the voxel dimensions k_n,I_n, h and the conductivity of a tissue in the voxel. A network resistor between n and m is formed by calculating the parallel connection of the four edge resistors R_1, R_2, R_3, R_4 of the surrounding voxels.

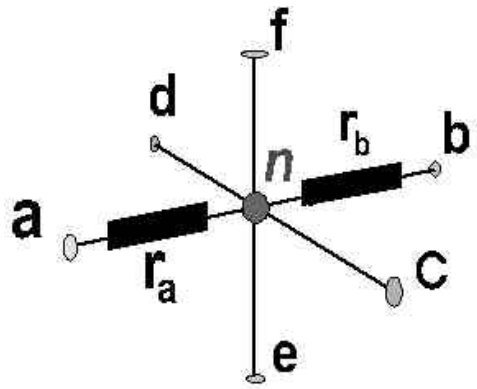

Figure 2-16 Network resistors. Resistors r_a and r_b calculated from the edge resistors are used as computational elements in FDM solving. The impressed current flows through the 3D resistor mesh causing potential differences between nodes [a,f] and n. Current and potential gradient components can be solved when the potentials of the nodes are known.

at all nodes in the volume conductor iterative methods *e.g.* successive over relaxation (SOR) method or Conjugate Gradient – Incomplete Cholesky Preconditioner (CG-ICP), are used. The iteration is stopped when desired accuracy for the potentials is reached.

In the visualization process the FDM solution software, NBFS, is used to solve potentials in the volume conductor. Of those potentials new information can be derived, which includes three components of gradient potential vector and three components of current vector. The solutions are combined with the model anatomy and the results are transformed using Cone into a suitable format for visualization and mapped with Elmer for inspection.

2.7 Python

Python was chosen as the programming language for the Cone for it is easy to learn and Python programs are relatively simple to combine with other systems. Python [Python 2002] is an interpreted, high level programming language originally developed at the university of Amsterdam by Guido van Rossum. It is easy to learn for beginners and has high-level data structures and a simple but effective approach to object-oriented programming. Python is an Open Source software and it is developing rapidly because of growing popularity towards Open Source systems such as

the operating system Linux. Python comes with wide selection of ready-made program modules.

Python's syntax, because of its high level nature, is very readable, easing maintainability and encouraging program modularity and code reuse. Indentation is used to create block structures further alleviating development. Python does not demand a compilation step and the edit-test-debug cycle is fast. A bug or bad input will never cause a segmentation fault. When the interpreter discovers an error, it raises an exception. When the program misses the exception, the interpreter prints a stack trace.

The Python implementation is portable and it runs on many types of UNIX, on Windows, DOS, OS/2, Mac, Amiga and any system that has a C compiler available. Python is best suited as a "glue" language between different components written with other low-level implementation languages. It can also be used for prototyping components before implementing them in other languages.

2.8 Noname Bioelectric Field Software

NBFS is software for computing bioelectric fields for FDM applications. This software package contains separate libraries and programs working in a pipe to bring FDM solutions from segmented anatomical data [Takano 2000]. The solution program originated at the university of Tasmania where Walker & Kilpatric preformed forward and inverse ECG calculations [Walker et al. 1987]. Later the solver was amended at RGI and used for bioelectrical calculations by Hyttinen [Hyttinen 1994], Kauppinen [Kauppinen 1999] and Laarne [Laarne 2000]. New programs have been contrived around the solver by Takano to form a software package [Takano 2001]. Takano is further developing and refining the different components of NBFS.

NBFS is composed of programs to make an FDM model (`befFdmMk`, `befMtfIf.py`), to solve the model (`befFdmSl`), to combine solutions

(befFdmCm), to pick up nodes (befFdmNdPk) and to report model and solution information (befFdmNdRp). befFdmMk, befMtfIf.py, befFdmCm are made for specific purposes whereas the others can be used for manifold purposes.

2.8.1 Process flow of Noname Bioelectric Field Software

The process begins by introducing IARD [Heinonen et al. 1990] segmented anatomy files of VCMT [Kauppinen et al. 1999] file format to befMtflf.py, which converts it to text format readable for befFdmMk making a model for subsequent programs use. The resolution of the anatomy file can also be reduced before the model is made. Usually data resolution is

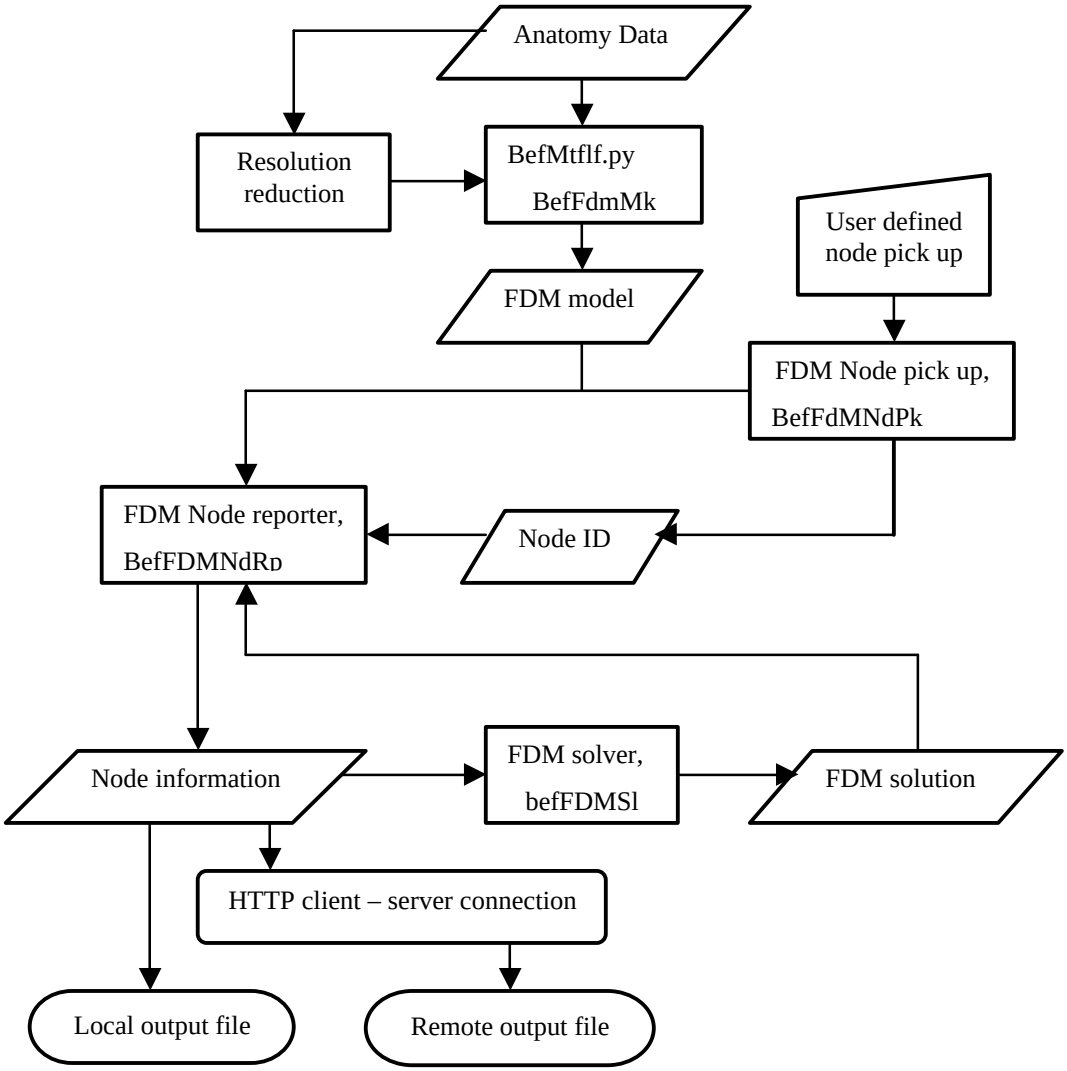

Figure 2-17 Flow chart of data and processes inside Noname Bioelectric Field Software.

reduced at the edges of the anatomy, for those regions contribute little to the electric fields when the heart is being studied. Sources and sinks are specified and added to the model and `befFDMSl` solves potentials at every node according to user definitions. Nodes or tissues of interest with additions (*e.g.* current vector, gradient potential at nodes) can be collected using `BefFdMNd` and `BefFDMNdRp`. The node file can be saved as a local file or transferred by HTTP client – server connection to a remote machine (Figure 2-17).

2.9 Elmer

Elmer is a multipurpose software parcel for simulations and visualization [Elmer 2000a, Lyly 2000]. It has been developed at CSC in cooperation with Finnish universities, research centers and industry. Elmer is not an single monolithic program, but a modular system consisting of three different segments (Figure 2-18) each able to work independently or in community to solve complex problems from building a model for the problem domain to visualizing the calculation results.

The Elmer package is available from CSC (*www.csc.fi*) and different versions work on UNIX (SGI, DEC, Linux) and Windows NT environments.

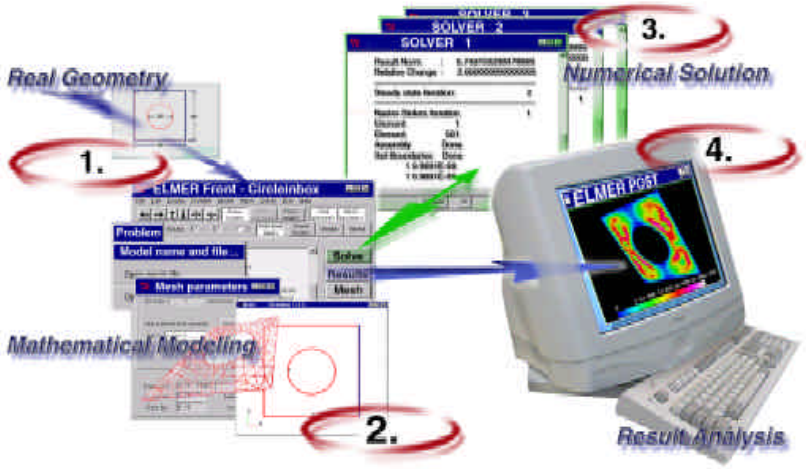

Figure 2-18 Elmer workflow [Elmer 2000]. As Elmer has been started, Elmer Front opens. The user can read in the file containing the model geometry (1.) , and start operating with the model shown on a separate Model window (2). After all definitions have been, Elmer Solver can be started (3.). Solver output is used in Elmer Post for visualization and results analysis (4.).

2.9.1 Elmer Front

The problem solving begins in Elmer Front by defining the domain to be used. Front communicates with external software producing geometrical data such as CAD files and computational meshes. Elmer Front generates its own finite element meshes, allows the user to build mathematical models graphically, and finally produces input data for Elmer Solver. The mesh generation techniques of Elmer Front are based on Delaunay triangulation algorithms, Voronoi diagrams, and advancing front methods.

2.9.2 Elmer Solver

Elmer Solver processes the physical models described in Elmer Front into a discrete form. The solver is based on FEM and solves stabilized finite element discretizations and weakly coupled solutions of transport (diffusion-advection-reaction equations), fluid flow (incompressible and compressible Navier-Stokes equations, elasticity (deformations and displacements, modal analysis), Acoustic scattering (Helmholz equations) and electric and magnetic fields (Maxwell equations) [Ruokolainen et al. 2000b].

An example of Elmer's application in biomedicine is a model of pulsatile flow inside aortic arch (Figure 2-19) related to Dynamic Adaptive Modelling (DynAMo) project conducted by the academy of Finland.

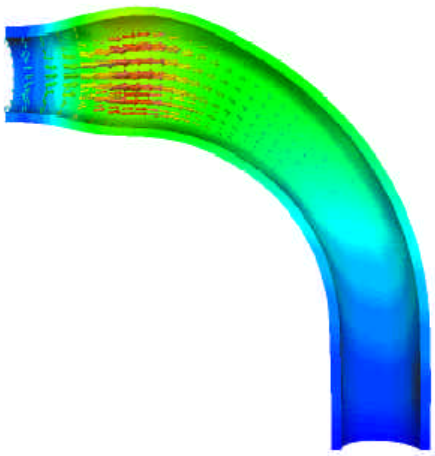

Figure 2-19 Flow inside aortic arch [Elmer 2001]. FSI-module of Elmer can be used to simulate the pulsatile blood flow in elastic blood vessels, which helps in understanding the effects of e.g. stenosis and bifurcations.

Problems can be solved as time dependent or time independent. Geometry and solution can be presented in different coordinate systems. The element types for the solver are Lagrange's element family. It includes lines, triangles, rectangles, tetrahedrons and cubes in linear, quadric or third degree base functions. Elmer is also an Open Source program, which enables the users to write more features to the solver that better meet their needs.

2.9.3 Elmer Post

The third segment of Elmer, Elmer Post, visualizes the numerical results produced by Elmer Solver and other finite element programs. Post is used through a window based user interface. Selecting a means of visualization produces a pop-up window for further selections for the presentation manner.

Post uses the same types of elements as Elmer Solver aside from third degree base functions. Post includes the typical visualization tools such as scaling, translation and rotation of the object. In addition to this, results can be presented in various fashions [Ruokolainen et al. 2000a] available in the *Display*-menu (Table 2-1). A few examples of these are using contour surfaces when solving isosurfaces of potential and colouring them according to the absolute value of current passing through the surface. In Figure 2-19 the walls of aortic arch are coloured by scalar pressure and vectors show the direction of blood flow with length and colour variable being the absolute value of velocity. In addition to different visualization methods colour map can be manipulated to meet different needs and a colour scale bar can be added. Also images can be combined as animations. Multiple cameras can be added for simultaneous views of the object. Elmer Post contains a math module MATC, which includes common matrix and vector operations, derivate operations and other standard functions and the basic structures of a programming language. The variables of the read model are available for use by MATC and the user can create new variables for visualization *e.g.* calculating the gradient field from potentials `gradient=grad(potential)`.

Table 2-1Data displaying in Elmer Post.

Colour Mesh	Shows the element mesh coloured with any field variable. The user can select between showing the mesh or filling the mesh to form surfaces.
Contour Lines	Draws isocontours of the chosen field variable for surface elements. In volume elements the isocontours are drawn on the elements' face.
Contour Surfaces	Draws 3D isosurfaces of the chosen field variable. The isosurface can also be coloured according to any variable.
Vector	Vector orientation is given by any selected vector field variable. The colour and length of the visualized vector itself can vary according to any desired scalar or absolute variable.
Particles and Streamlines	In time dependent cases it draws trajectories of given particles in velocity field. In time independent cases streamlines of a selected vector filed can be drawn
Spheres	The given field variable can be shown as different size and colour spheres at nodes.

3. Methods

3.1 The visualization process

The subject of this Master of Science thesis requires the understanding of the whole visualization process (Figure 3-1) from the beginning to the end to fully make use of different components of the system. The process does not necessarily stop after the end of post processing has been reached. The user might want to highlight different aspects of the data and thus the process starts again from the point where some thing needs to be changed. Usually this happens only in the post processing, but as the data is better understood also some components in the former parts may be changed.

The visualization process here starts with data acquisition, where MR-images of patients are taken. These MR-images are 2D cross-sections of the patient presented with gray-scale, where every pixel in the image corresponds to some tissue. In pre-processing the MR-images are processed with IARD algorithm implemented in Anatomatic [Heinonen 1999]. Anatomatic is a semi-automatic segmentation program developed specially for separating and labelling different tissues from gray-scale images. In the next stage a model is constructed from the segmented data. The resolution of the data can be reduced before a model is made of it by NBFS. The

produced model is fitted with current sources and sinks representing the bioelectromagnetic sources present in the living tissue, and FDM solution is calculated. In post processing the user can extract tissues of interest from the solution and model files. The file containing information of the nodes is loaded into Cone, which produces a text file in a format suitable for the visualization segment of Elmer, Elmer Post. In Elmer Post user can choose

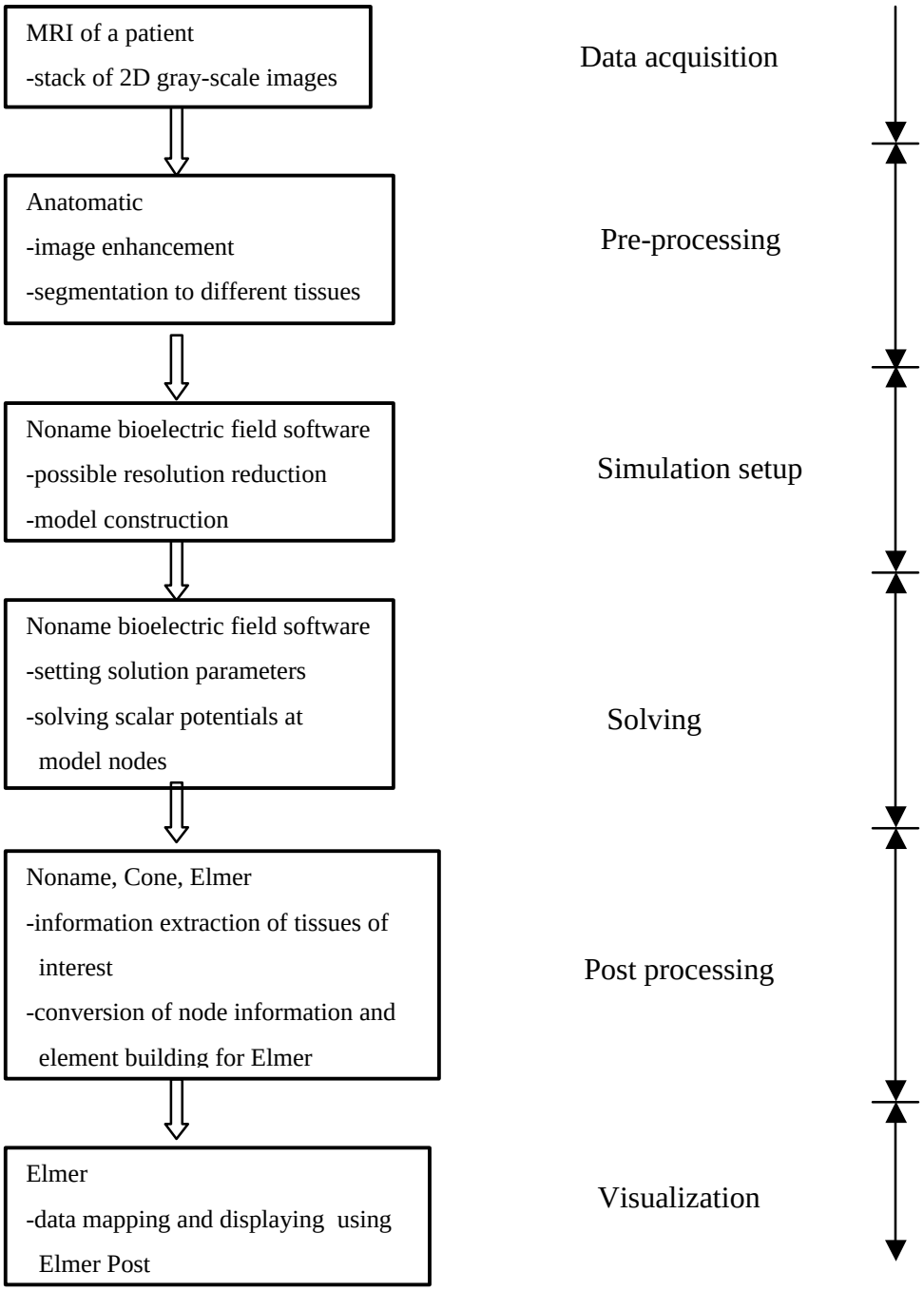

Figure 3-1 Visualization process from data acquisition to final images.

which field variables he/she wants to visualize. From those results decisions for further processing (if any) are resolved.

3.2 Material for Cone

Cone is a conversion program between NBFS and Elmer. Before it there was no means to visualize the simulation results of NBFS, thus there was a need for visualization software. Elmer is a quickly advancing software developed at CSC and it is available free to universities in Finland. Product support is also available on-line. Naturally the output format of NBFS and the input format of Elmer differ quite a bit, thus a converter program needs to be advanced.

The specifications for Cone were that it should be able to read text files produced by NBFS, efficiently produce elements of the nodes and write node coordinates, element information and features of every node to a text file for Elmer. In addition to this, new demands rose during the development process. These were not very visible to the end user, but affected the program structure a lot.

The program development itself was done at RGI during 2001 and 2002. The process demanded skills for programming the Python language and insight in visualization. The skills were acquired in the beginning of the thesis project. As knowledge of the data, the environment, and visualization grew, the program was implemented with new features and optimisations. Python programming language was chosen for the project for reasons stated in the previous chapter. The programming platform was Windows 2000 build on Windows NT system by Microsoft (*www.microsoft.com*) running on a PC workstation fitted with an Intel Pentium II at 350MHz and 384Mb of RAM memory. Programs used in the development were 4NT version 3.02B, Ultraedit-32 version 8.10A, Python interpreter version 2.0, Elmer release 2.0 and Matlab release 12. 4NT is a command line tool developed by JP-software (*www.jpsoft.com*) to make the "C:\" prompt more effective and easy to use. New commands have been added and improved command line editing alleviates prompt-based work. Ultraedit-32 is a text/HEX editor from

IDM Computer Solutions Inc. (*www.ultraedit.com*) with efficient tools for program code editing. Its disc based editing system allows effective handling of large files without using a lot of RAM, which was an important feature when browsing in- and output files of many megabytes. Ultraedit-32 was used for writing and editing the source file for Cone and the files were run as a script from the 4NT command line to the Python interpreter. Python interpreter checks the source file for syntax errors and reports problems. If no errors were found the program is run. The output files from Cone were used for input files to Elmer Post. In Elmer the result was examined and tested by visualizing different aspects of the data (Figure 3-2). Matlab was used in a small role in fitting a curve to findings about the resulting files.

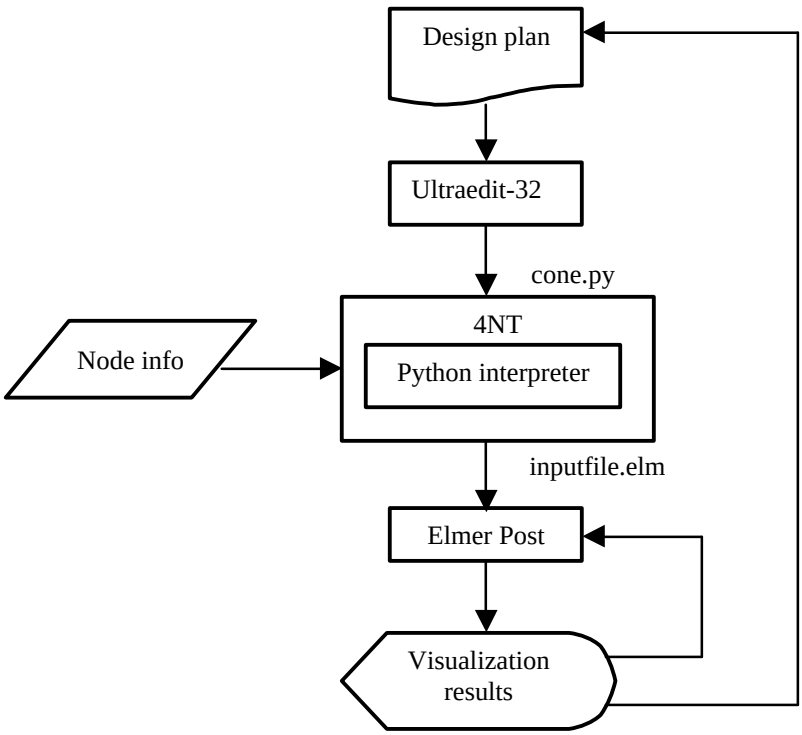

Figure 3-2 Design flow in the development of Cone. The process starts by sketching a layout of the program. Ultraedit-32 editor was used to produce code. Python interpreted was operated form 4NT command line. Input and outputfiles were defined at command line. Visualization results were produced with Elmer Post. The whole process is recursive.

3.3 Overview of Cone

As explained earlier Cone reads a node information file. The file is formed by extracting nodes of interest from the FDM solution and FDM model files using programs in NBFS. The resulting text file is saved on a local disc for further processing. Depending on the amount of nodes in the file the size of it can be from kilobytes to several megabytes. One line per node is used to present the data associated with that particular node. The data and its order on one line is the following:

- Node ID (A linear numbering for the nodes in the model)
- Node IDs of the six neighboring nodes (Figure 3-4)
- The index of the node in Cartesian coordinate system (x, y, z)
- The physical location of the node in millimeters (x-, y- and z-distance)
- The tissue codes of eight surrounding elements
- Scalar potential at the node
- Three components of gradient potential vector
- Three components of current vector
- Impressed current (*i.e.* the current fed into or drained from the node)

These 35 items read on every line are stored in the computer's memory for further processing. Nodes and the location information about them are sorted according to the tissues they are in contact with. After sorting tissues the number of nodes associated to them are printed for the user for a decision about the output file. The user inputs the desired tissue

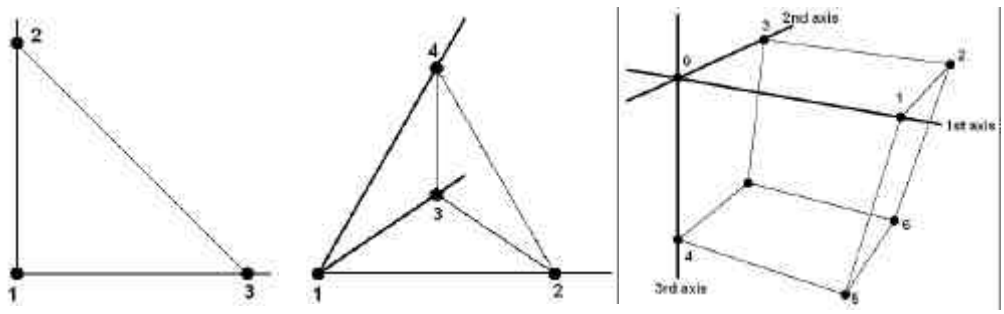

Figure 3-3 The three elements of the Lagrange family that are used in Cone: Linear triangle, linear, tetrahedron and linear cube.

codes into Cone along with the codes for elements that are wanted to represent the tissues. The input file contains just nodal data; therefore elements must be created from those nodes. Elements are needed to for surface and volume renderings. The elements can be triangle, tetrahedron or cube. Any one element can be given to any tissue and one file can contain multiple element types.

The elements belong to the basic element set of Elmer Post, which consists of the linear and quadric elements of the Lagrange family [Elmer 2000]. Altogether there are eight different elements in Elmer Post

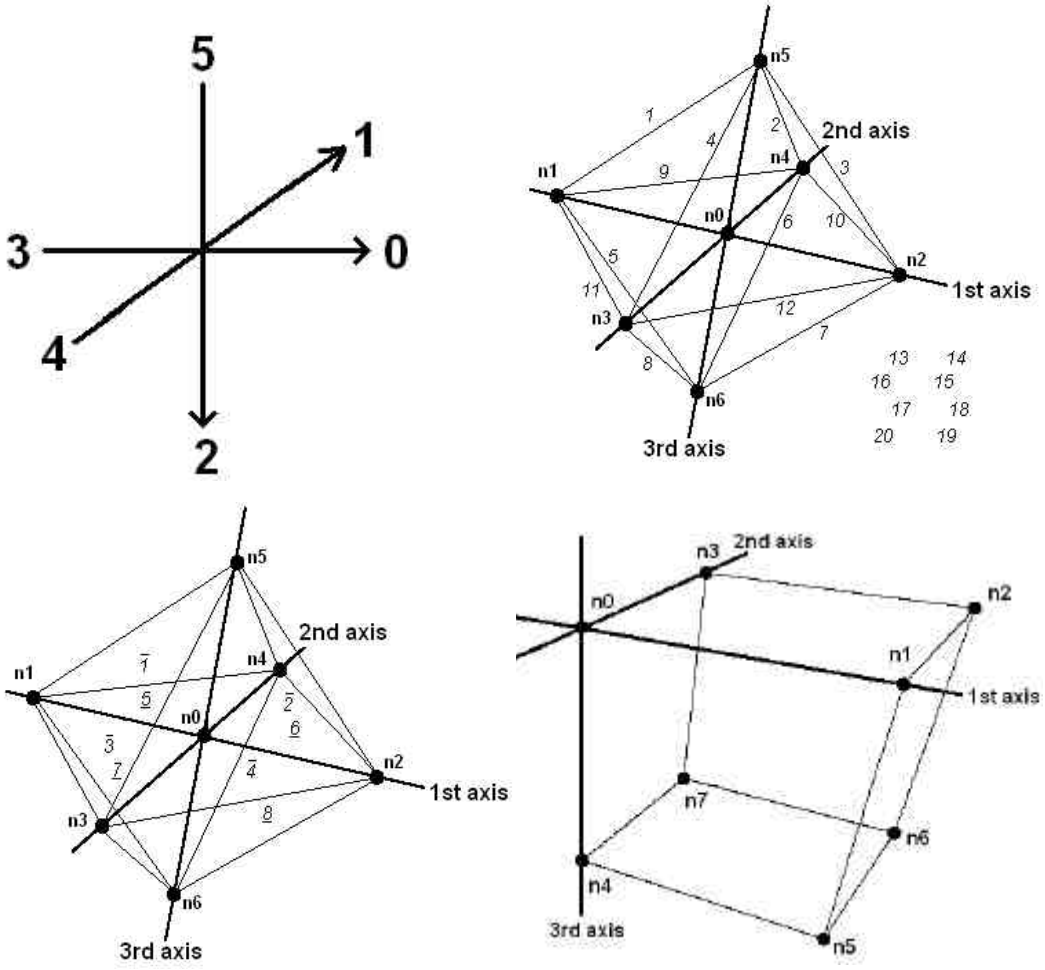

Figure 3-4 Elements produced by Cone. Top left: In the origin a node and its immediate neighbors (i.e. adjacent nodes from 0 to 5) in the order they are given in the input file. Top right: All the triangles (1 to 20) that can be formed with a node and its adjacent nodes. Triangles 13 to 20 are the exterior walls. The corresponding numbers are placed beside the figure in the order they would appear in it. Bottom left: All the tetrahedrons (1 to 8) that can be formed. The numbers with a dash over them are the upper elements and the underlined numbers are the lower elements. Bottom right: The cube element with its eight apices. n1, n3 and n4 are n0's neighbors while n2, n5 and n7 are n6's

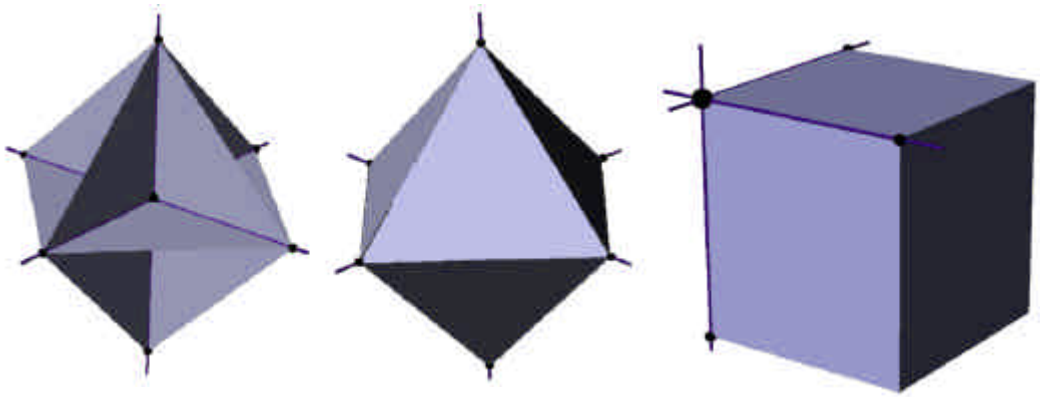

Figure 3-5 3D rendered elements of Cone. Left: The triangles. Only triangles 1 to 12 are visible. Triangles 13 to 20 are the exterior walls, which would make the representation look identical to the one in the middle. Middle: The tetrahedrons. Right: The cube.

[Ruokolainen 1997], but only three linear elements are handled by Cone (Figure 3-3). The other types proved to be unsuitable for the task at this point.

For one node and its immediate neighbours 20 triangles can be formed. With tetrahedrons the same number is 8. For a cube the node in the other end of the cube's diagonal must be searched. Three immediate neighbours of these two are used to form a cube (Figure 3-4, Figure 3-5). Because it is not certain that all the adjacent nodes actually exist in the input file, they must be searched and verified. After the element generation has completed the nodal information and the elements are written to an output text file.

The program itself, Cone, is a `*.py` –file *i.e.* a Python script. The program is run on command line in Windows and needs a Python interpreter to operate. On the command line the command "python" must be typed (ensure that Python is in the search path of the shell) to invoke the interpreter. After that the path to the `cone.py` –script file must be written. Additional arguments thereafter are passed to the script in the variable sys.argv, which is a list of strings. The first argument, `argv[1]` (`argv[0]` is the script name), is the input node file and the second, `argv[2]`, a user determined name for the output file. The output file name can be any, but `*.elm` is encouraged to separate the Elmer Post specific file from other text files. The resulting `*.elm` –file can be directly opened in Elmer Post, and visualization can commence straightaway.

The program code was designed to have a function-based structure (Figure 3-6). The main program only consists of function calls and glue structures between them. The `cone.py` is a text file containing a script written for the Python interpreter. The file itself is divided to four sections. First there

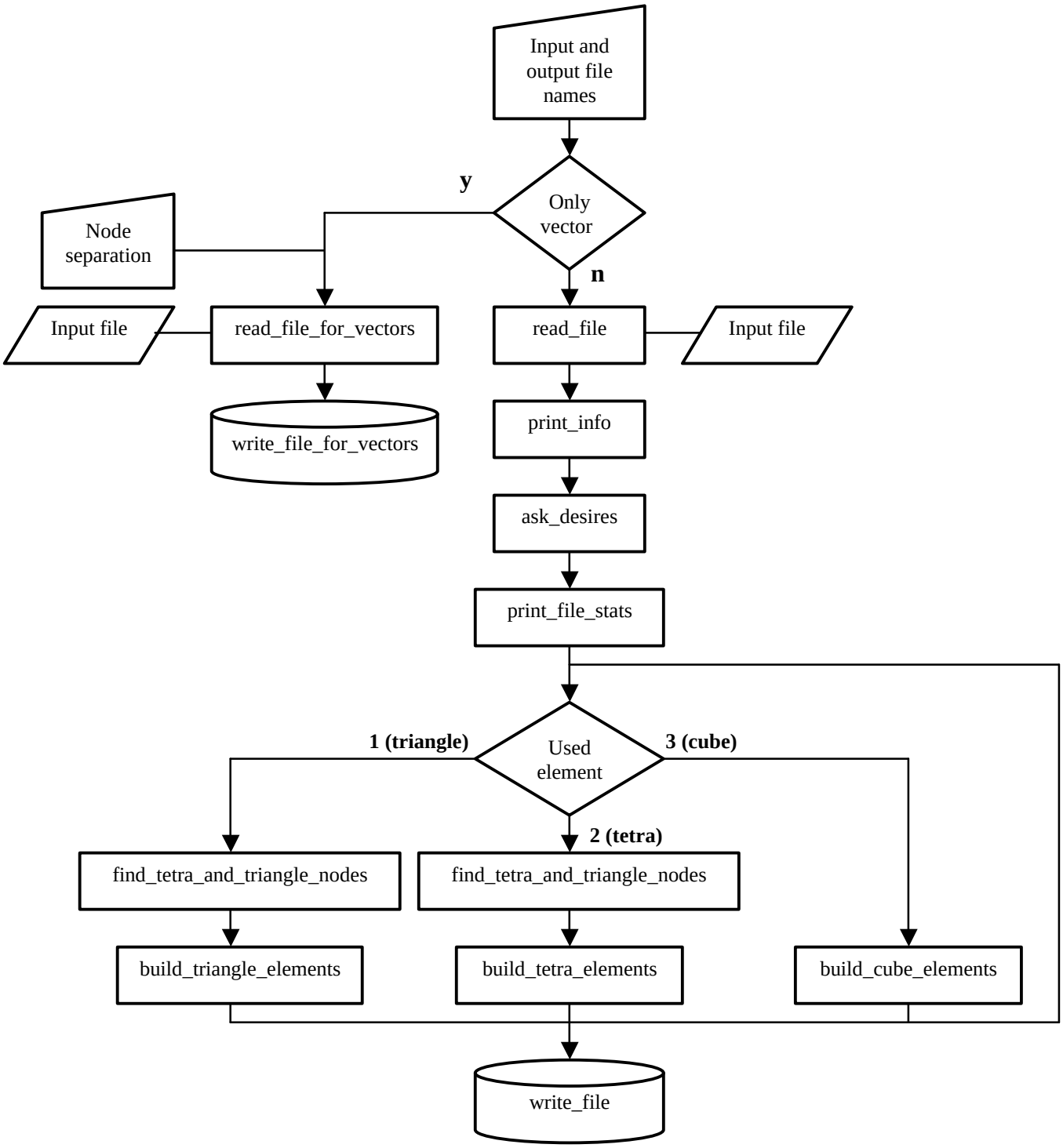

Figure 3-6 The functions inside Cone and their relations

is a header shortly reporting information about the program and its purpose. The second section is a set of modules. These are files containing Python definitions and statements that are not available in the interpreter by default, but must be imported in the beginning of the script file if they are needed during the execution of the script. The third section contains definitions of the functions that are called from the main program, which is the fourth and last section. In the program code readability is emphasized by choosing descriptive function and variable names. In all function names words are separated with underline characters such as `read_file_for_vectors` in distinction to variable names, where the words are written in combination like `allCellCodes`.

3.4 Header and imported modules

The header is made up of two parts. The first is general information about Cone and the second is imported modules to provide some additional functions to the program. All the lines in general information begin with '#', which is a comment sign in Python. The interpreter ignores the following text on that line.

The imported modules are `time`, `string` and `sys`. `time` module provides various time-related functions. From that `clock` is used. `string` module defines some constants useful for checking character classes and some useful string functions. From that `split` is used. `sys` module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

3.5 Functions

The functions appear in the `cone.py` in the same order as they are explained here. These functions contain simple Python commands which are executed each time the function is called somewhere in Cone.

- `read_file`

  -argument: `inFnam` (the input file name from the command line)

  -return: `nodeInfo, nodeStats, IDArray`

The function basically contains a `while`-loop where the input file is read line by line until the `inFile.readline()` –function reads a zero length string signifying the end of file. The read strings are split into items, which are used to build two dictionaries (hash table or hash in other languages), `nodeInfo` and `nodeStats`, and one list, `IDArray`. `nodeInfo` and `nodeStats` contain node id, adjacent nodes, index in 3D, surrounding cell codes and physical location, potential, gradient, current vector, impressed current respectively for every node.

- `read_file_for_vectors`

  -arguments: `inFnam,n`

  -return: `nodeInfo`

The action of this function is in principle the same as the previous except that it reads from the input file only every n^{th} node. This is to prevent vectors in vector visualization from occluding one another and obscuring the visualization.

- `print_info`

  -argument: `nodeInfo`

  -return: `allCellCodes, nodesSrtdByCllCds`

This goes through all nodes and appends all unique cell codes that are found in a list, `allCellCodes`. Then a new list, `nodesSrtdByCllCds`, containing as many empty lists as the number of unique cell codes, is formed. Again all nodes are browsed through and the nodes are appended to every list in `nodesSrtdByCllCds` corresponding to the cell codes with which it is in contact. Next the results are printed for the user. The print out is of the format:

```
Tissue type n: m nodes
```

Where *n* is the tissue code and *m* is the nodes in contact with that tissue.

- `unique_sort`

  -argument: `c`

  -return: `rtn`

  The function simply receives a list, `c`, as an argument and searches for unique values in the list and forms a new list, `rtn`, of them. The resulting list is then returned.

- `sort_lists`

  -arguments: `allCellCodes, nodesSrtdByCllCds`

  -return: `aCC, nSBCC`

  This receives the lists containing unique cell codes and the nodes sorted by cell codes. Note that these lists are of the same length therefore they are linked by an index, however these lists are in arbitrary order and need to be sorted. `sort_lists` sorts first all cell codes to ascending order, `aCC`, and then sorts the list containing nodes to match the index, `nSBCC`, as in the lists before.

- `ask_desires`

  -arguments: `allCellCodes, nodesSrtdByCllCds` (note that these are now sorted)

  -return: `desires, elmntsFrTsses`

  Here the user is asked for the cell codes (*i.e.* tissues) he/she wants to include in the output file. The desires are recorded to a list, desires. If an asked code is not available an error message is produced. Then elements are chosen for the tissues. Available elements are triangle; selection 1, tetrahedron; selection 2 and cube; selection 3. An invalid selection results in an error message.

- `print_file_stats`

  -arguments: `desires, elmntsFrTsses, allCellCodes, nodesSrtdByCllCds, nodeInfo`

  -return: `none`

This calculates information about the output file. When handling large input files and choosing triangle as the element for representation can result in surprisingly large files. The function of this function is to function as a promoter of the overall system functionality by printing out the approximate amount of lines and characters in the resulting file, and thus give information about the process, that is going to take place, to the user.

- `search_IDs`

  -arguments: `IDArray,n`

  -return: `x`

  Here binary search is utilized to search a specific node from a list of nodes. In the binary search the search area is halved until the area is only 1 element long. If the desired node is not in the list, None (*i.e.* false) is returned.

- `find_tetra_and_triangle_nodes`

  -arguments: `j, IDArray`

  -return: `n0LNr, n1LNr, n2LNr, n3LNr, n4LNr, n5LNr, n6LNr`

  `j` is an element from `nodeInfo` containing position and ID information for a node. The adjacent nodes for the node and their existence in the input file are cleared. This needs to be done because usually the input files are only extractions of the full model. In here `search_IDs` is called to search for specific nodes from `IDArray`. `nxLNr` (x=[0,6]) is a line number of the node in the input file. This is needed when elements are written into a file.

- `triangle`

  -arguments: `a,b,c`

  -return: `triangle`

  This simply examines if `a, b` and `c` are true (i.e. not false, such as `None`). If so, it makes a list containing these line numbers and returns it.

- `build_triangle_elements`

  -arguments: `n0LNr, n1LNr, n2LNr, n3LNr, n4LNr, n5LNr, n6LNr`

-return: `triangleCache`

The function goes through all the triangles that are possible to form of one node and its six neighbours without making a tetrahedron. Every triangle naturally consists of three apices and triangle is called to determine if they are available. The result is appended into `triangleCache`.

- `tetra`

- argument: `a,b,c,d`

- return: `tetra`

This function is essentially the same as `triangle`, but it is applied to tetrahedron.

- `build_tetra_elements`

-arguments: `n0LNr, n1LNr, n2LNr, n3LNr, n4LNr, n5LNr, n6LNr`

-return: `tetraCache`

Again this serves the same purpose as `build_triangle_elements`, but for tetrahedrons.

- `form_array`

-arguments: `partialInfo`

-return: `subIDArray`

The function forms a partial array of node IDs of the nodes associated with chosen tissue. This shortens the search time inside `build_cube_elements`.

- `build_cube_elements`

-arguments: `i, Info, IDArray, subIDArray`

-return: `n0LNr, n1LNr, n2LNr, n3LNr, n4LNr, n5LNr, n6LNr, n7LNr` or `0,0,0,0,0,0,0,0`

This is the equivalent of `triangle` and `build_triangle_elements`, and `tetra` and `build_tetra_elements` for the cube element. Here the starting node is `i` (0 in Figure 3-3), and three edges of the cube can be defined by

information in `i`. The node in the other end of the cube's diagonal must be searched after resolving its ID by calculating its index in 3D using coordinate information in `i`.

- `write_file`

-arguments: `outFnam, nodeStats, results`

-return: `elements`

The purpose of this function is to write the processed information to a text file for Elmer Post. First a one-line header is written. For this the number of nodes and the number of elements must be resolved. The header consists of:

```
nn ne nf nt scalar: potential vector: gradient vector: current
scalar: impressed_current
```

where *nn* is the number of mesh node points, *ne* is the total number of elements in the mesh, *nf* is total the number of degrees of freedom (add three for a vector and one for scalar quantity), *nt* is the number of time steps stored in the file and *scalar: name*, *vector: name* are a list of variable types and names respectively. Here *nn* and *ne* are calculated by `write_file`, *nf* and *nt* are always 8 and 1 respectively. The types and names are as they are presented above.

After that the coordinates (x, y, z) for every node are written. Then comes an optional separator, `#group all`, between the coordinates and element information. Element information is written as:

```
groupName elementTypeCode 0 1 2…
```

where *groupName* is the name of the element group (tissue code and element type e.g. `85cube`) , *elementTypeCode* is a numeric code giving the element type (Figure 3-3), and [0,N] are indices to the node coordinate array in the beginning of the file. These make the vertices of an element. Following these there is again an optional separator, `#endgroup all`. After that comes time step information:

```
#time n t time
```

where *n* is the order number of the time step written in this file, *t* is the simulation time step number and *time* is the current simulation time. In here these are 1, 1, 0 respectively, because the FDM solution used is quasistatic. Finally come the nodal values of the degrees of freedom, one node at a time, and in the order given by the keywords, scalar: and vector: in the header.

- `write_file_for_vectors`

-arguments: `nodeStats`

-return: `None`

This is the same operation as the previous function, but for vector visualization. The written data is the following: header, coordinates for nodes, separator, one element, separator, time step, node quantities. The one element is merely a dummy for Elmer Post, for it gives an error message if there is no element. In practise this poses no trouble, since only vectors are visualized.

3.6 The main program and its operation

The main program begins with argument assignment from the command line. These are the names of the input file and the output file. Following that is the major crossroads of the program (Figure 3-6). The user can select between handling only vectors or generating also volume elements. The amount of nodes is usually so large, that visualizing vectors of the full set leads to a muddled result. This is because the thousands of arrows occlude each other and hide the important aspects that the user wants to see. In vector only visualization the user can select the separation of the nodes that are read form the input file, thus clarifying the end result, because each node contains vector information. If ‘y’ is entered, the user is prompted to give node separation. The separation is used by `read_file_for_vectors` as it reads only every n^{th} node from the file. `nodeInfo` is returned to the main program used again by `write_file_for_vectors`. It opens a text file, `outFnam`, for writing and writes

the information given in the previous section. After writing the file a note is given about conclusion of the program and it exits.

If the user has answered ‘n’ in the query about the vectors, input file, `inFnam`, is read and `nodeInfo`, `nodeStats`, `IDArray` are returned. `print_info` sorts the nodes according to tissues they are in contact with and prints the results. Then `ask_desires` queries about the tissue types the user wants to include in the output file and the elements used to form them. The default element types of Elmer are the linear and quadric elements of the Lagrange family [Elmer 2000]. Of these Cone utilizes 2D linear triangle (3 nodes, element type code 303 in Elmer, 1 in Cone), 3D linear tetrahedron (4 nodes, element type codes 504 and 2) and 3D linear cube (8 nodes, element type codes 808 and 3) (Figure 3-3). `print_file_stats` prints the information about the resulting output file. This includes approximations about the total amount of lines and characters to be written in the file. The approximations are made of empirical research data about the output files and making a generalisation about the amounts by fitting a curve in Matlab to suit the findings. From the fits, formulas for general cases of line and character amounts are obtained and implemented in Cone.

The main program goes through all user selected tissue codes and the elements requested for them. If the selected element is 1 (triangle) `find_tetra_and_triangle_nodes` and `build_triangle_elements` process the nodes. Finally `triangleCache` is browsed and all empty elements (`None`) are discarded by Boolean logic and the valid triangles are appended to `results`. Note that `results` is a dictionary with keys being the tissue codes and the element code being in the index 0 of the corresponding value (dictionaries are built as key-value pairs). The procedure is alike for tetrahedrons when the selected element is 2 (tetrahedron). If the selected element for a tissue is 3 (cube), `subIDArray` is formed by `form_array` to be used in speeding up search operations in `build_cube_elements`. Again if the mentioned function returned valid elements they are appended into `results`. `results` is given as an argument to `write_file` along with `outFnam` and `nodeStats`. `write_file`

writes the text file, which is saved to the local disc following a note of the completion of the whole procedure and the program exits.

4. Results

During development Ultraedit-32 text/HEX editor, 4NT command line tool and Elmer Post of Elmer software package were used to create and develop Cone. Cone is a converter between the solver, NBFS, and the visualization program, Elmer Post. The operation of Cone was designed to be as straightforward as possible. The operations are done in a very logical order as one might expect such a program to do. Cone first reads the input text file from NBFS given at the command line. The program then requests information from the user about the output file, and the user inputs the tissue codes wanted for visualization. Cone starts to build elements and then writes all coordinates, built elements and nodal information as an output text file.

4.1 An example process

The program is started from the command line; which is of the following format:

```
python cone.py inputFileName outputFileName
```

where *python* is the command to start the Python interpreter, `cone.py` is the converter script file name, *inputFileName* is the name (or path) to the input file and *ouputFileName* is the desired name for the resulting output file. The operation continues with the following print.

```
Do you want to visualize only vectors? (y/n)
```

where the user answers ‘n’ or ‘y’ (Figure 3-6) depending on the visualization needs. The program begins to read the given input file printing an asterisk (*) for every 500 nodes (100 if only vectors are chosen) read. When the reading operation is complete the program reports the total number of nodes read.

```
Reading input file: * * * * * * * * *
4500 nodes read.
```

Next, nodes are sorted by the types of tissue they are in contact with. Because a node is in contact with eight computational FDM elements (the nodes are in the apices of the cubical elements) at maximum, one node can be associated to multiple tissues.

```
Sorting nodes by tissue codes: * * * * * * * * * * * * * * * * *
```

After sorting, the nodes are reported by tissue codes.

```
Available tissue codes and the number of nodes in contact with
that tissue:
Tissue type 65: 87 nodes
Tissue type 67: 256 nodes
Tissue type 69: 1527 nodes
Tissue type 71: 1472 nodes
Tissue type 75: 2964 nodes
Tissue type 77: 73 nodes
Tissue type 79: 38 nodes
```

The report is followed by a query about the desired tissues and codes for them, where > is the prompt sign.

```
Which tissue codes do you wish to include in the output file?
 [Input tissue codes separated by spaces.]
 Example: >67 89 101
>69 71 75
 Choose elements for different tissues/boundary.
 [1 for triangle. 2 for tetrahedron. 3 for cube.]
 Element for 69
>1
 Element for 71
>2
 Element for 75
>3
```

After that an approximation of the output file is given and the element forming process commences printing the number of processed nodes.

```
Output file contains approximately 10500 lines and 454840
characters.
Processing nodes to elements:
100
200
300
```

When the process is finished the output file will be written and named as defined in the command line. End results are reported and the program exits.

```
674 nodes and 4354 elements written.
The process took 0 minutes and 2.6 seconds.
Exiting
```

4.2 Efficiency

Cone was tested multiple times during development with varying file sizes and contents. The test data was sets of `LW_Fnc.g21` and `level6A_V3.Wilson`. `LW_Fnc.g21` is an inhomogeneous thorax model of a human for modelling the behaviour of electric fields produced by the heart in the end of systole (*i.e.* the contraction of the heart and arteries).

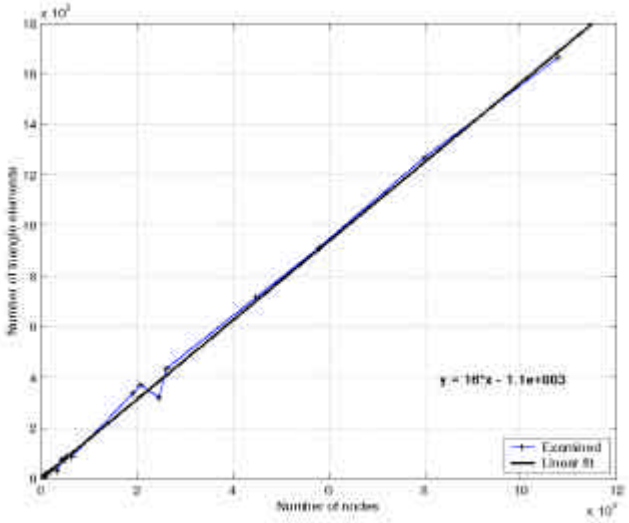

Figure 4-1 Output file comparison for triangles. The number of nodes in an output file versus the number of elements created of them. The thin line represents the examined files and the thick line is a linear fit made with Matlab. The fit function is visible above the chart key.

`level6A_V3.Wilson` is a calculated solution form the model for the *V3* precordial lead refered to the Wilson central terminal.

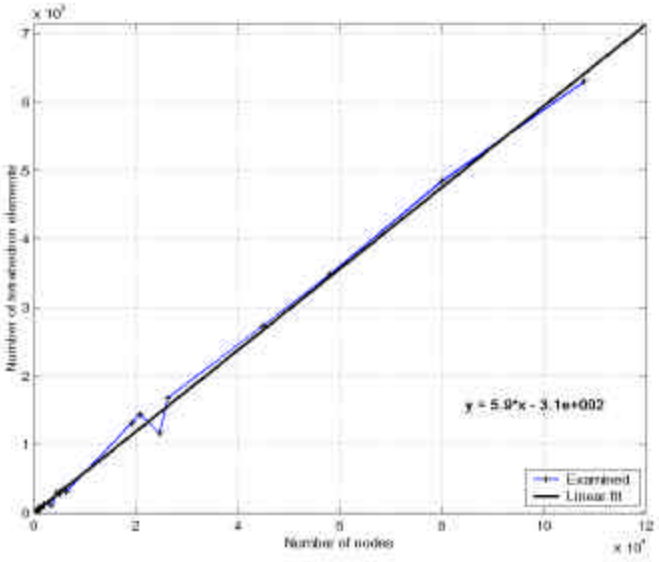

Figure 4-2 Output file comparison for tetrahedrons. See Figure 4-1.

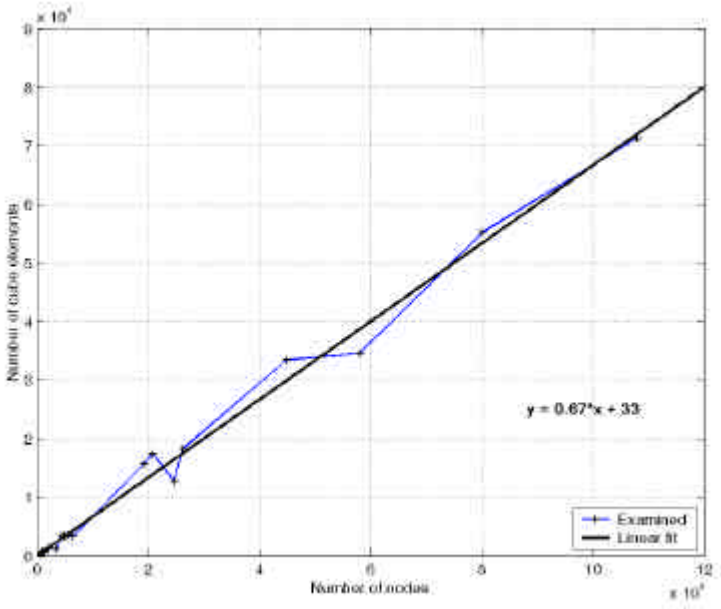

Figure 4-3 Output file comparison for cubes. See Figure 4-1.

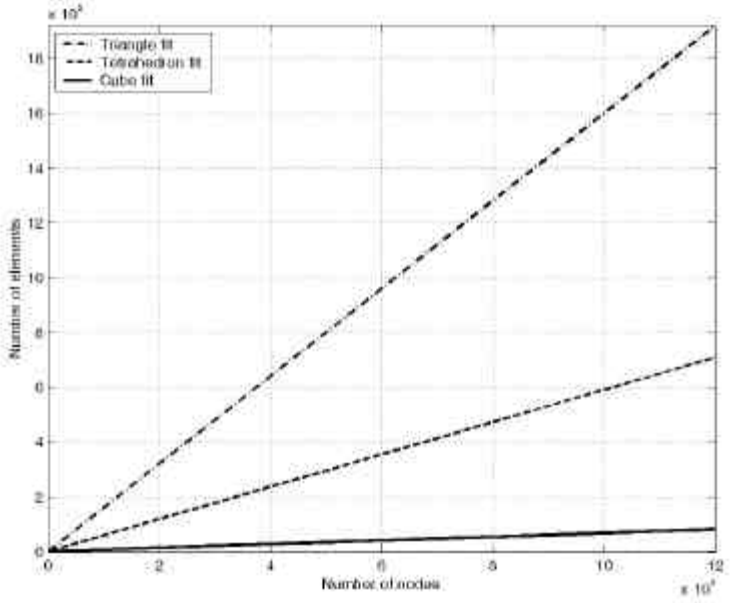

Figure 4-4 Comparison of curve fits for the three elements. Triangle produces the most elements, then tetrahedron and then cube. This is because the number on nodes in an element is 3, 4, and 8 respectively.

The number of elements produced by Cone of the nodes read from the input file increase almost linearly (Figure 4-1, Figure 4-2, Figure 4-3). Small deviation is due to different tissues in the test files. Some uniform large tissues, such as lungs, produce relatively more elements than fragmented tissues like the blood vessels or bones. Triangle is, because it has only three apices, the most versatile element and capable of producing fine structures. Because of that it also produces the most elements. The next element, tetrahedron, produces far less elements than the triangle. This is because one tetrahedron itself can be represented with four triangles. Cube is the most bulky element, as it has eight apices. Cube is poor for presenting delicate structures, but performs well with large solid volumes. It also produces the least elements thus leading to small output files. One advantage of cube is that it represents the computational elements used in solving the nodal information. If the resolution has not been reduced before the model is made in NBFS the cubes also depict the original voxels after the segmentation.

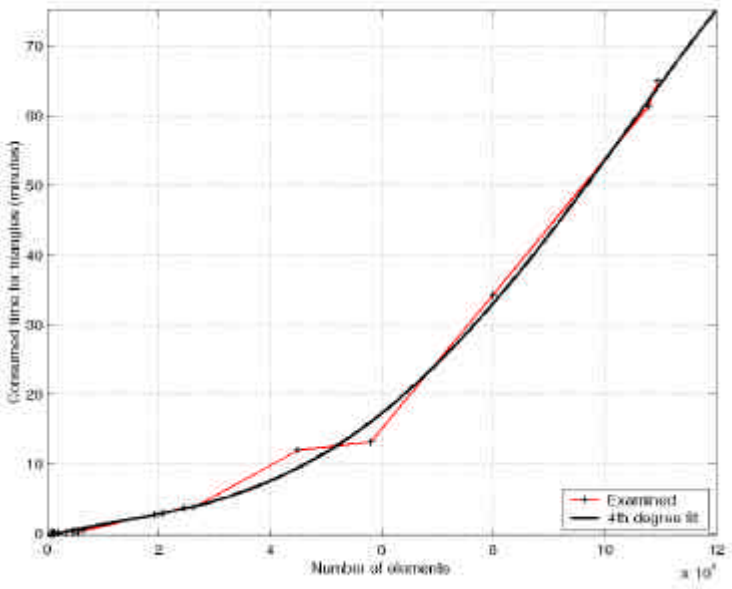

Figure 4-5 Time consumption examined using triangles during the making of an output file. All the files were made with the 350Mhz workstation. Time expenditure increases exponentially, as the number of nodes grows. A 4th degree fit was made with Matlab. The thin line represents the original data and the thick line depicts the curve fit.

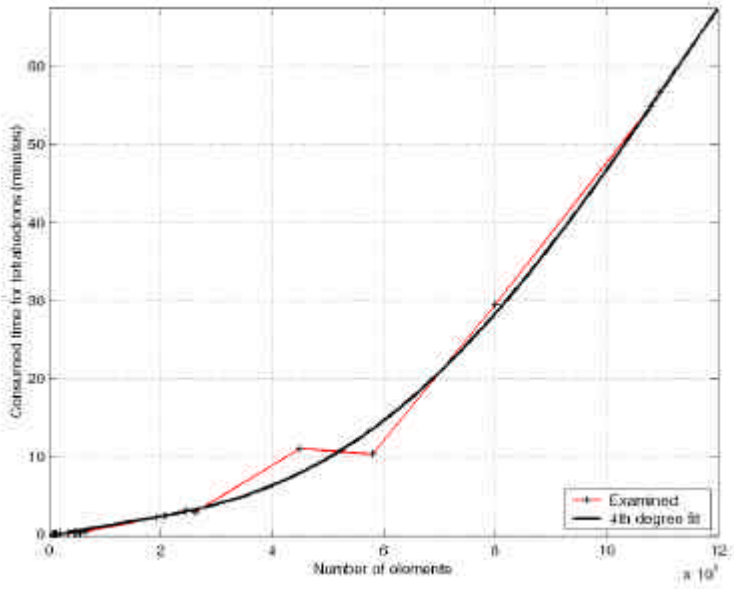

Figure 4-6 Time consumption using tetrahedrons. See Figure 4-.

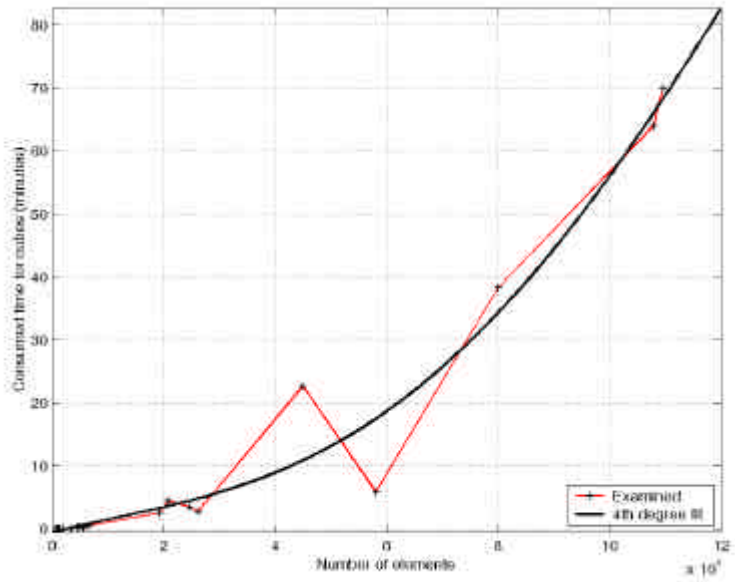

Figure 4-7 Time consumption using cubes. The test files were not very consistent resulting in poor data for curve fitting. Never the less the exponential trend can still be seen. See Figure 4-.

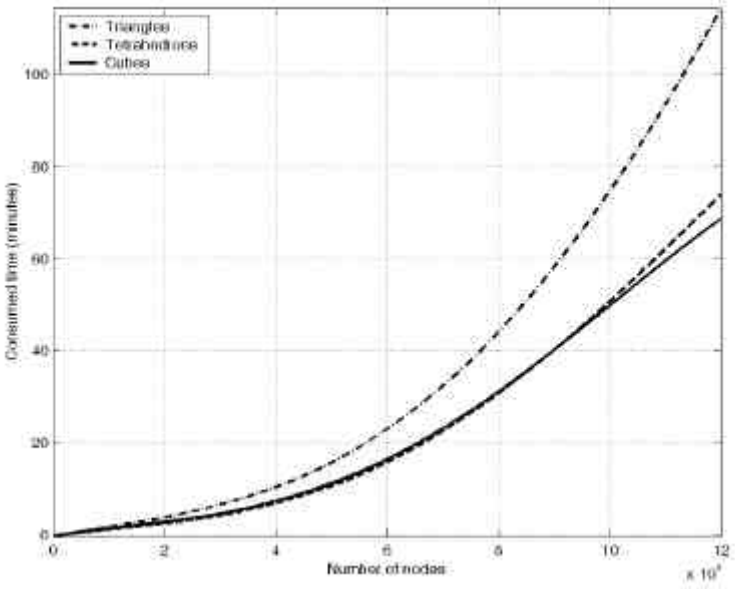

Figure 4-8 The 4th degree fits of time consumption of all the three elements. Many triangle elements are slower to form that fewer tetrahedrical or cubical. The efficiency between tetrahedrons and cubes depends on the amount of nodes in the requested tissues.

All the calculations were done with the 350Mhz Pentium II workstation. The time consumption during the running of the program depends on the desired tissues and the number of nodes in them (Figure 4-5, Figure 4-6, Figure 4-7). This is because Cone processes every node to see if

they have enough neighbours to form an element. The reading of the input file happens in seconds, as does the node sorting. After the program has information which tissues and nodes to take into account, the element processing starts. It was noted that many small sets process faster than a few large ones, because the bottle neck in the process is the validation of adjacent nodes' existence (`find_tetra_and_triangle_nodes`, `build_cube_elements`). Many small lists are faster searched than a few large lists.

The element building process was timed and exponential time consumption was observed. Deviations from the 4^{th} order curve fit can be explained by the inconsistency of the test data. As said earlier many small data sets process faster than a few large ones even if they altogether have the same number of nodes. Triangles were the slowest to process because many searches had to be made to produce many elements. Triangles and cubes were almost equal considering time consumption. Because the curve fits are merely trend setting it was noted in practise that tetrahedrons were a bit faster to make than cubes. Evidently they have the best ratio of nodes in an element to elements in the output file.

4.3 Visualized results

The text files written by Cone were opened in Elmer Post for visualization. In the file open dialog box the header for the input file is shown. The user can decide to generate only surface element side for surface visualization or he/she can choose to create also volume element sides, which enables structures inside volumes to be examined. Also the number of time steps can be chosen, but this has no importance in this case, as the modelled situations are quasistatic. After the input file has been read the user can select the means of visualization in the Elmer Post Processing window.

Surface reconstruction was tested with all three elements and it was noted that triangles and tetrahedrons give nice round surfaces with large

tissues such as the combined heart muscle and heart fat. Cubes on the other hand give an angular lego-block like surface, but they represent the computational elements faithfully (Figure 4-9). Please note that all the images presented here are available in the appendix C in color as larger versions. The surface can be colored according to desired field variables (Figure 4-10) or it can be presented as a mesh consisting of the elements' edges (Figure 4-11). The mesh can also be colored according to any desired variable. In these mappings the values for the entire volume are interpolated from the discrete node values. The colormap used to describe the change of field variables can be changed for different applications (Figure 4-12). The user can employ ready-made colormaps or define own maps to meet his/her needs. Isocontours of chosen field variables can be plotted on the elements to visualize the rate of change across the volume (Figure 4-12). Isosurfaces of a variable can be drawn and colored according any desired variable. For example, visualizing the rate of potential change by calculating potential isosurfaces and coloring them according to the current vector absolute values depicts the change of potentials and the current flowing through the so formed surfaces.

In vector visualization Elmer Post forms vector arrows according to the three components given in the input file. The vector length and coloring can portray different field variables. The vectors can be scaled and colored according to absolute values calculated from the vector components following the Lie factor rule (Figure 4-13). Information content for the visualization can be enhanced by changing the length and coloring to present other field variables.

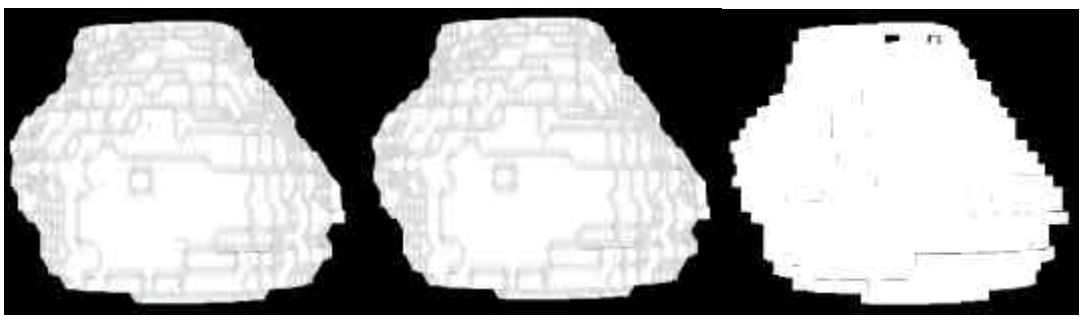

Figure 4-9 The heart presented with triangles, tetrahedrons and cubes. The heart is positioned as it would be seen in the thorax in anterior view i.e. the apex of the heart is pointing to the lower right. With a large volume triangles and tetrahedrons form fine surfaces, but the cubes give a slightly angular surface.

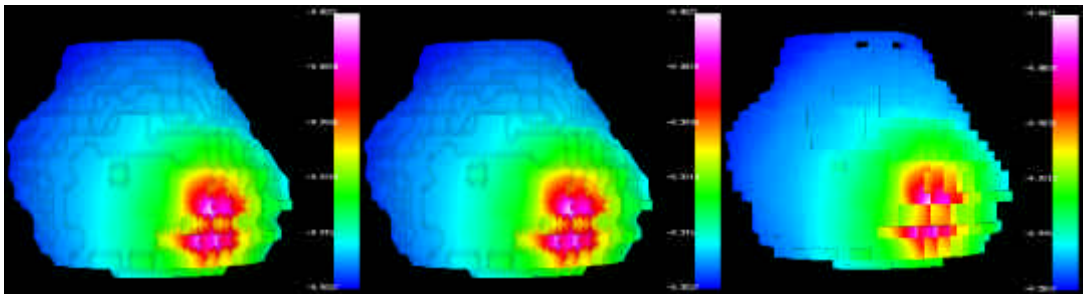

Figure 4-10 The triangular, tetrahedral and cubical heart surface colored according to scalar potential. Any field variable can be mapped onto the elements.

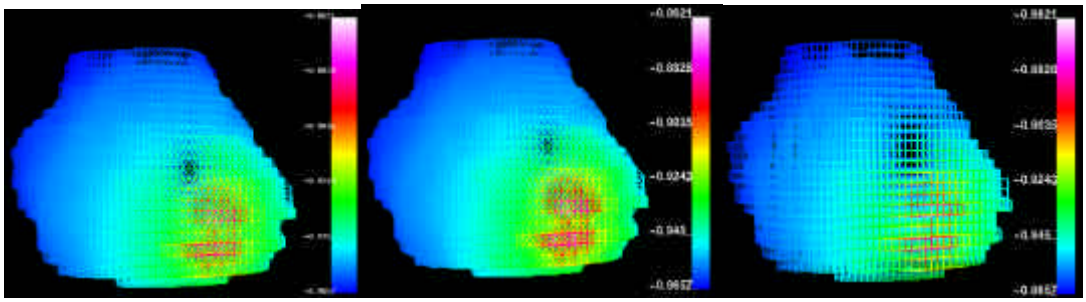

Figure 4-11 The volume presented as a triangular, tetrahedral and cubical mesh. The element edges can be colored according to any field variable.

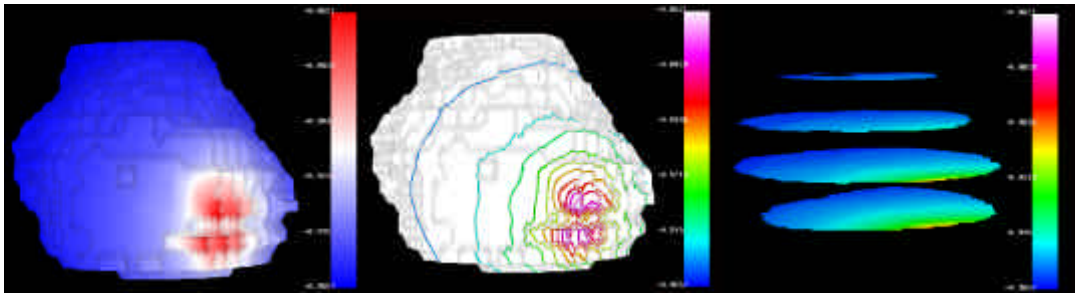

Figure 4-12 Left: The color map can be changed to suit the needs. A blue-white-red color map is used to present scalar potential on the heart's surface. Middle: Isocontours are plotted on the elements. The division between the contours is linear. Right: The heart volume has been cut with 4 isosurfaces of z-coordinates (axial). The formed cut planes are colored according to scalar potential.

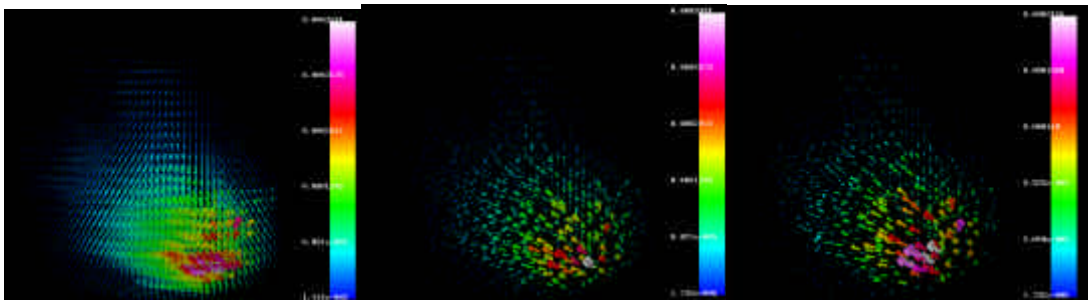

Figure 4-13 Visualization of current vectors in the heart volume. The vector length and coloring correspond to the absolute value of the current vector. Left: Vectors are drawn to all the nodes in the heart. Middle and right: Vectors are restricted to every 5^{th} and every 10^{th} node respectively.

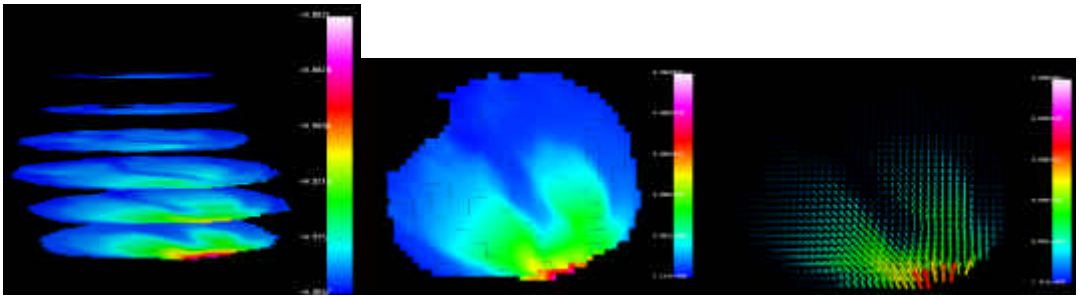

Figure 4-14 Left: The heart volume has been cut according to 6 isosurfaces of z-coordinates (axial). The planes are colored with the absolute value of the current vector. Middle: The rendered volume can be limited with object clip planes in the direction of the object's coordinate axes. This differs methodologically form the isosurface presentation. The cut volume is colored according to the absolute values of the current vectors. Right: The same clip plane as in the middle is used to present the current vectors. The length and the coloring correspond to the absolute value of the current vectors.

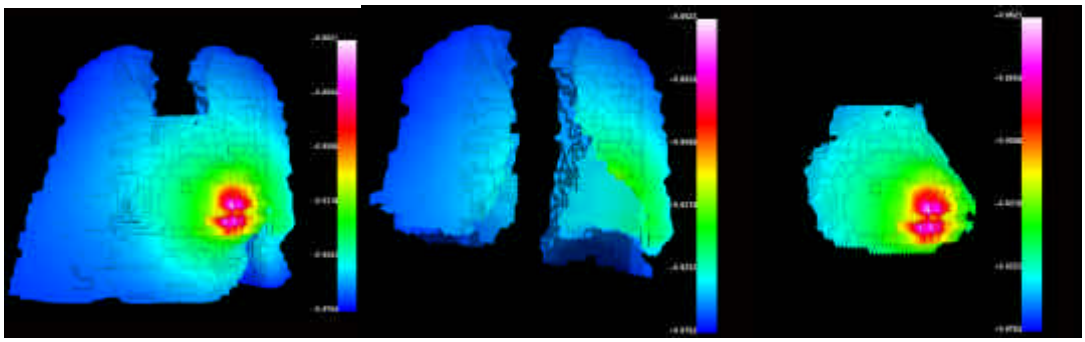

Figure 4-15 The heart and the lungs colored according to the scalar potential. When the elements are labeled by the tissue codes the user may choose to make some tissues transparent in order to examine the tissues behind or inside them. Here the heart is seen separated from the lungs.

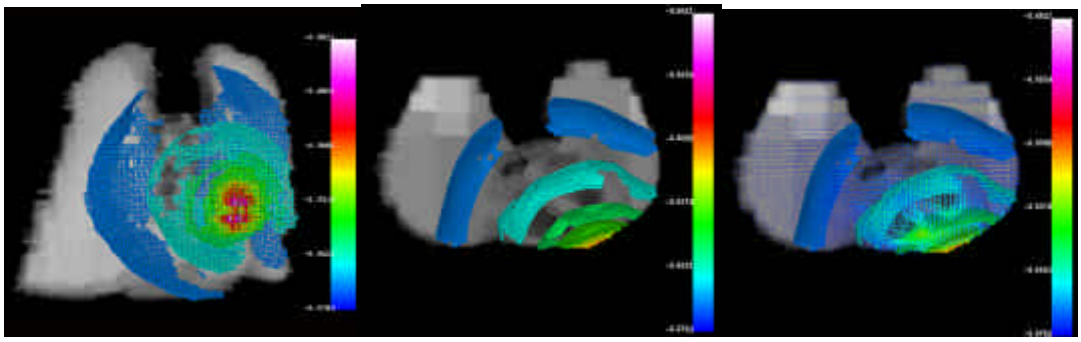

Figure 4-16 Fancier visualization. Left: The isopotential surfaces in the lungs and the heart are presented with meshes colored according to the scalar potential. The surface of the organs is illustrated as partly opaque for clarity. Middle: The same as the image on the left, but from axial direction. A cut plane made with object clip planes is shown partly opaque. Right: The same as in the middle, but current vectors are added. This is an example of bad visualization by putting too much data in one image resulting in confusion.

By definition Cone writes vector data for all nodes resulting in a confusing image if the visualized vector field is large. The amount of nodes read from the input file can be constrained, thereby also reducing the

number of vectors *i.e.* the user can define the separation of the read nodes before Cone starts to read the file. This significantly improves the quality of the vector visualization as the vectors do not overlap and obscure the result (Figure 4-13). Because it is difficult to detect the depth in vector mode, the object boundaries can be limited into depicting just a defined slice of the volume (Figure 4-14). The clipping applies to nodes, thus all information content in them is also omitted.

The segmentation, which separates the volume into different tissues, enables the user to choose the tissues he/she wants to be rendered, thereby giving a chance to examine different areas of interest without some other objects blocking the view (Figure 4-15). Visualization gives tools for viewing multiple aspects of the object simultaneously. Many filed variables can be shown in one image in addition to the geometry of the object (Figure 4-16). When making complex visualization it must be born in mind that the image must convey the important features to the observer efficiently, for it is very easy to stuff one image full of data and make a nice and colourful picture with no use whatsoever due to the overflowing information content.

The visualized surfaces, volumes and the different means of presenting the field variables shed light to the computational data derived from NBFS. The results were satisfactory considering that there has been no way of presenting this data before. The current and potential visualizations enable researchers to make deductions of the calculated data and how they correspond to real subjects. The algorithm for producing elements worked well considering the computational efficiency of the workstation giving results for visualization in seconds and for larger models in minutes. Implementing Cone into more efficient systems further accelerates the processing. Cone is easy to operate for new users and tests with the program are easy to run. When the data is better understood alternate ways can be used to process the data with Cone. Advanced users may modify the function based program code to make it produce also other kinds of elements that are available for Elmer Post to suit their needs.

5. Conclusions

The visualization process was considered during this project. It contains the gathering of data through measurements or producing the data through computational methods. The data is thereafter transformed for visualization by data processing, filtering or other computational techniques. Next, the data is mapped to a form appropriate for presentation to the user, and finally the data is rendered or displayed completing the process. The process from MRI acquisition to visualized bioelectric field variables with Elmer Post was completed by developing a converter, Cone, using Python programming language to modify and generate the output files from NBFS into input files for Elmer Post.

5.1 The development process of Cone

The process of developing Cone started by gathering knowledge of software development with Python and visualization from various sources in the Internet and literature. Using that knowledge Cone was coded to work as a Python script. Cone reads the input file given at the command line into the computer's memory. It divides the nodal information in the file according to the tissue types with which they are in contact, presents the results, and prompts the user for desired tissues in the output file. Next, element types for the presentation of those tissue types are given. Cone can generate three

types of elements of the Lagrange's element family used in Elmer. These are linear triangle, linear tetrahedron and linear cube. As the elements are created Cone writes all nodal information and the elements to the output file named in the command line. The files written by Cone were opened in Elmer Post for visualization. Different means of presenting the data were applied with different element types.

The amount of elements created for surface and volume visualization was liner when compared with the amount of nodes read. Fragmented tissues produce fewer elements than large homogenous tissues. The time consumption was examined to be exponential. Small files processed in a matter of seconds to minutes, but large thorax files consisting of multiple tissues were heavy to create with the workstation used in the development.

5.2 Effective visualization

The visualized results in Elmer Post showed smooth surface reconstruction using triangles or tetrahedrons. Cubical elements give angular surfaces, but they present the original voxels used in the model creation faithfully. Small delicate features are better visualized with triangles, as they need only three available adjacent nodes to form an element. The surfaces and volumes can be colored according to any field variable presenting scalar information. The vector visualization gives information about the current and vector fields inside the volume. Volumes can be cut and made partly or totally transparent to reveal structures inside the volume.

The goals set in the introduction were reached, as the visualization process is now complete. The results of the bioelectric fields simulated with NBFS can be read into Cone, which not only produces direct output, but gives the user possibility to view the contents of the input file and choose different elements to present the node array. During the process Cone prints information about the progress giving possibility to abort at any situation. The written text file can be directly opened in Elmer Post and visualization of the bioelectric fields can commence.

5.3 Future development

The developed program and its operation works efficiently with small input files containing nodes in the order of 10^5 and less, as it was designed and tested mostly with light input files due to the computing efficiency of the workstation. This is adequate in the study of individual organs or regions of the body. When the computational load increases, the time consumption grows exponentially.

5.3.1 New approaches

During the subsequent development a new element-forming algorithm and more efficient memory management must be developed to accelerate the process, or more computational power must be harnessed. Cone is going to be implemented into a fixed pipe between the NBFS' node extraction programs and the Elmer Post to function as a filter getting input from NBFS and giving output into Elmer Post. For more efficient operation of the program it may be implemented in another lower level programming language such as Java or C++. As the utilization environment changes to a more powerful frame the problem of limited computational power is also solved.

As Cone gets rewritten new approaches will be experimented. The program will run more lightly as it will read and process as it runs through the file unlike now as it reads the input file into the memory demanding an abundance of RAM to hold all of the information available during the processing. Also a new algorithm for making the elements must be developed. This can embrace more intelligent adjacent node seeking and surface recognition to form smoother and complete surfaces. Even implementation of Delaunay triangulation is a possibility.

Summa summarum: The set goals were reached but the visualization process is by no means perfect. Future development must be done, but this has been a promising start.

References

[Damanian 1972] Damanian R., 1972. Apparatus and method for detecting cancer in tissue. U.S. Pat. 3,789,832

[Dowsett et al. 1988] Dowsett DJ, Kenny PA, Johnson RE, 1998. The physics of diagnostic imaging, London, Chapman & Hall, 609pp

[Elmer 2000] Elmer User's Guide CSC, 1st ed., 2000. Helsinki, 201 p.

[Elmer 2001] Elmer homepage, 10.9.2001. *http://www.csc.fi/elmer/*

[Heinonen et al. 1997] Heinonen T, Eskola H, Dastidar P, Laarne P, Malmivuo J, 1997. Segmentation of T1 MR Scans for Reconstruction of Resistive Head Models. *Computer Methods and Programs in Biomedicine* 54, Ireland. Elsevier Science Ltd. pp. 173-181.

[Heinonen et al. 1998] Heinonen T, Dastidar P, Kauppinen P, Malmivuo J, Eskola H, 1998. Semi-automatic Tool for Segmentation and Volumetric Analysis of Medical Images*. Medical & Biological Engineering & Computing* 36, 3, pp. 291-296.

[Hyttinen 1994] Hyttinen J, 1994. Development of regional aimed ECG leads especially for myocardial ischemia diagnosis, Ph.D. dissertation. Tampere University of Technology, Finland

[Kauppinen 1999] Kauppinen P, 1999. Application of Lead Field Theory in the Analysis and Development of Impedance Cardiography, Ph.D. dissertation. Tampere University of Technology, Finland

[Kauppinen et al. 1999] Kauppinen P, Hyttinen J, Laarne P, Malmivuo J, 1999. Software implementation for detailed volume conductor modelling in electrophysiology using finite difference method, *Computer methods and programs in biomedicine* 58 (1999), pp. 191-203

[Laarne 2000] Laarne P, 2000. Implementation of a Realistic Conductivity Model for the Head, Ph.D. disertation, Tampere University of Technology, Finland

[Lyly 2000] Lyly M, Ruokolainen J, Järvinen E, 2000. ELMER - A finite element solver for multiphysics*. CSC-report on scientific computing 1999-2000.* 6 p.

[Heinonen et al. 1990] Heinonen T, Dastidar P, Frey H, Eskola H, 1990. Applicaiton of MR Image Segmentation. *IJBEM* 1999 1(1), pp. 35-46

[Malmivuo et al. 1995] Malmivuo J, Plonsey R, Bioelectromanetism: Principles and applications of bioelectric and biomagnetic fields. New York, Oxford University Press, 480 p.

[Python 2002] The Python Language Website. 24.5.2002, www.python.org

[Ruokolainen 1997] Ruokolainen J, 1997. Elmer Post Help. /ELMER2.0/post/help/index.html

[Ruokolainen et al. 2000a] Ruokolainen J, Savolainen V, 2000. ELMER Post –visualisointiohjelmisto. *@CSC* 1 (2000), pp. 27-28

[Ruokolainen et al. 2000b] Ruokolainen J, Savolainen V, 2000. ELMER Solver – yleinen FEM-ratkaisija. *@CSC* 2 (2000), pp. 14-16

[Schroeder et al. 1998] Shroeder W, Martin K, Lorensen W, 1998. The visualization toolkit, New Jersey, Prentice Hall Inc. 645 p.

[Takano 2000] Takano N, 15.9.2000. New FDM Software: Sample Procedure. *http://alpha.cc.tut.fi/~noriyuki/model/doc/bef/sample01/sample01.html*

[Takano 2001] Takano N, 2001. Noname Bioelectric Field Software – document version 1.20 for software edition 1.20. 45 p.

[Walker et al. 1987] Walker JS, Kilpatrick D, 1987. Forward and inverse electrocardiographic calculations using resistor network models of the human torso. *Circulation Res* 61. pp. 504-513

[Webster 1988] Webster JG, 1988. Encyclopedia of medical devices and instrumentation, Volume 3, United States of America, John Wiley & Sons, 2174 p

Appendices

Appendix A

This is an example of the text file that serves as an input for Cone. In this file there are all the nodal information for the heart and its surroundings. Only the beginning is given as the whole file consists of 20114 lines of text. The order of information is given in the methods chapter. Note that the lines have been wrapped due to space limitations.

```
93104 93105 93188 96772 93103 93020 89402 49 19 34 2.4150000e+02
1.7100000e+02 1.7000000e+02 87 117 117 117 69 117 67 117 -
9.6439927e-01 -8.0942744e-05 2.0549878e-04 1.6296489e-04 5.1120285e-
06 -1.3972650e-05 -6.7478101e-06 0.0000000e+00
93105 93106 93189 96773 93104 93021 89403 50 19 34 2.4450000e+02
1.7100000e+02 1.7000000e+02 67 87 69 117 69 69 69 67 -9.6454386e-01
-6.5751584e-05 2.2212036e-04 1.6327669e-04 2.1008505e-06 -
5.9719629e-06 -2.6137354e-06 0.0000000e+00
93172 93173 93256 96840 93171 93088 89470 33 20 34 1.9350000e+02
1.7400000e+02 1.7000000e+02 87 69 117 117 69 69 117 69 -9.6500722e-
01 1.1109465e-04 1.6946598e-04 9.0373139e-05 -4.7598587e-06 -
3.5603775e-06 -2.1054062e-06 0.0000000e+00
93173 93174 93257 96841 93172 93089 89471 34 20 34 1.9650000e+02
1.7400000e+02 1.7000000e+02 117 87 117 117 69 69 117 117 -
9.6483110e-01 1.1097258e-04 1.4134471e-04 1.0030767e-04 -7.1734318e-
06 -6.9973490e-06 -3.7781557e-06 0.0000000e+00
93188 93189 93272 96856 93187 93104 89486 49 20 34 2.4150000e+02
1.7400000e+02 1.7000000e+02 87 117 87 117 69 117 69 117 -9.6406951e-
01 -6.1728954e-05 2.1320840e-04 1.6487582e-04 1.5324614e-06 -
1.1459951e-05 -5.3172451e-06 0.0000000e+00
93189 93190 93273 96857 93188 93105 89487 50 20 34 2.4450000e+02
1.7400000e+02 1.7000000e+02 87 87 67 87 67 69 69 69 -9.6422001e-01 -
7.8170848e-05 2.4110432e-04 1.4788759e-04 1.0063413e-06 -3.6165648e-
06 -1.3309883e-06 0.0000000e+00
93190 93191 93274 96858 93189 93106 89488 51 20 34 2.4750000e+02
1.7400000e+02 1.7000000e+02 69 87 69 67 69 67 69 69 -9.6430402e-01 -
9.4008838e-05 1.8691036e-04 1.5009213e-04 1.1251262e-06 -2.8036554e-
06 -1.3508291e-06 0.0000000e+00
```

Appendix B

This is a sample of output text file written by Cone, which serves as an input file for Elmer Post. The file is written as a result after the processing of the file in appendix A. Most of the coordinate, element and nodal data is omitted due to space limitations, as the file contains 60951 lines of text. The header and nodal information lines are wrapped.

```
20114 20719 8 1 scalar: potential vector: gradient vector: current
scalar: impressed_current

241.500000 171.000000 170.000000

244.500000 171.000000 170.000000

193.500000 174.000000 170.000000

196.500000 174.000000 170.000000

241.500000 174.000000 170.000000

#group all

87cube 808 2 3 10 9 164 165 180 179

87cube 808 4 5 12 11 169 170 187 186

87cube 808 5 6 13 12 170 171 188 187

87cube 808 7 8 17 16 177 178 195 194

87cube 808 11 12 22 21 186 187 202 201

#endgroup all

#time 1 1 0

-0.964399 -0.000081 0.000205 0.000163 0.000005 -0.000014 -0.000007
0.000000

-0.964544 -0.000066 0.000222 0.000163 0.000002 -0.000006 -0.000003
0.000000

-0.965007 0.000111 0.000169 0.000090 -0.000005 -0.000004 -0.000002
0.000000

-0.964831 0.000111 0.000141 0.000100 -0.000007 -0.000007 -0.000004
0.000000

-0.964070 -0.000062 0.000213 0.000165 0.000002 -0.000011 -0.000005
0.000000
```

Appendix C

Here can be found all the images presented in the results chapter. All the images were produced using the Elmer Post visualization program. These images and others not presented here can be found in appendix D.

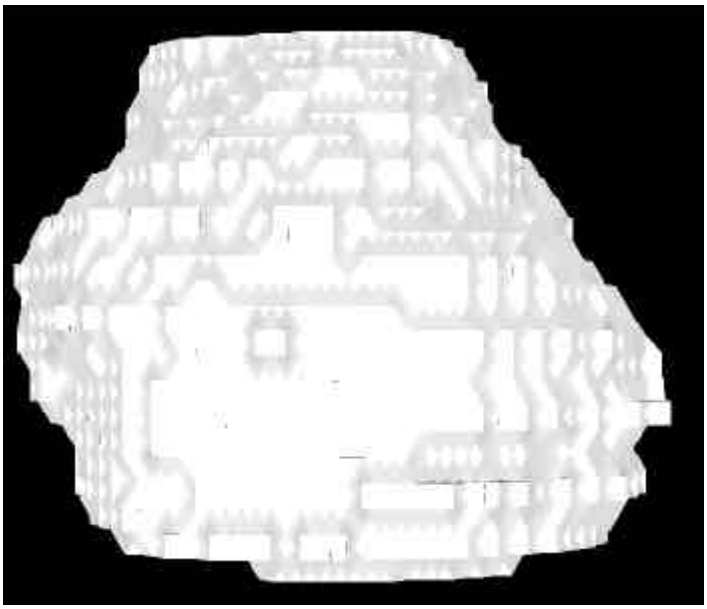

Figure 4-9 The heart presented with triangles, tetrahedrons and cubes. The heart is positioned as it would be seen in the thorax in anterior view i.e. the apex of the heart is pointing to the lower right. With a large volume triangles and tetrahedrons form fine surfaces, but the cubes give a slightly angular surface.

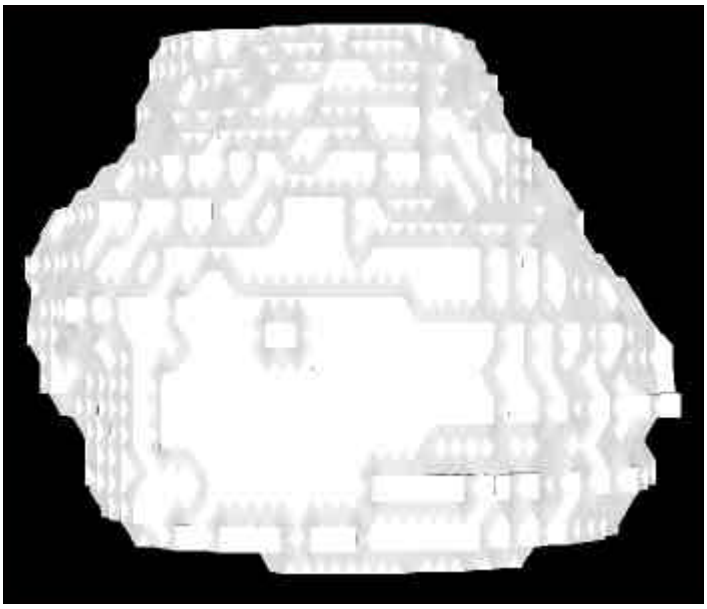

Figure 4-9: Middle.

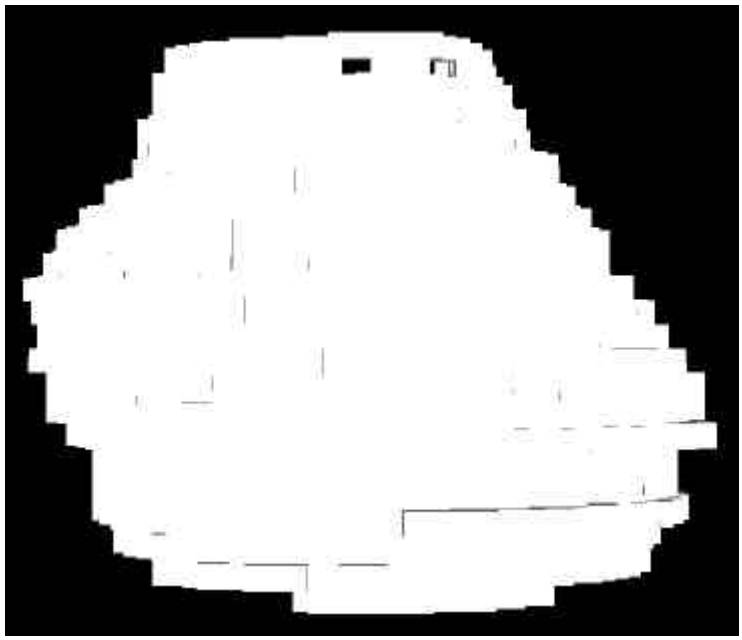

Figure 4-9: Right.

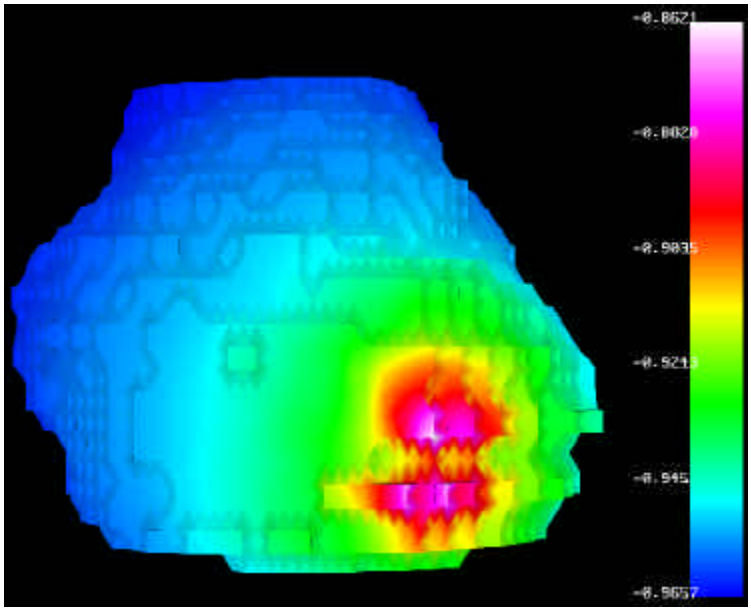

Figure 4-10 The triangular, tetrahedral and cubical heart surface colored according to scalar potential. Any field variable can be mapped onto the elements.

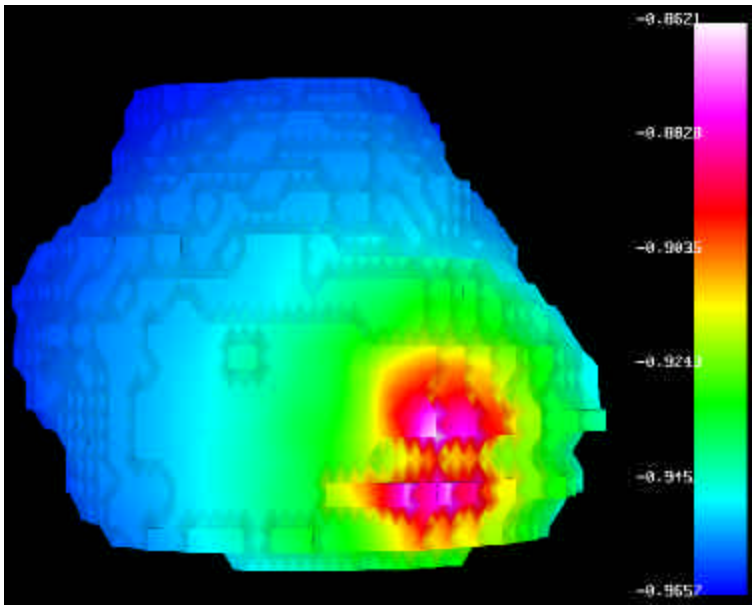

Figure 4-10: Middle.

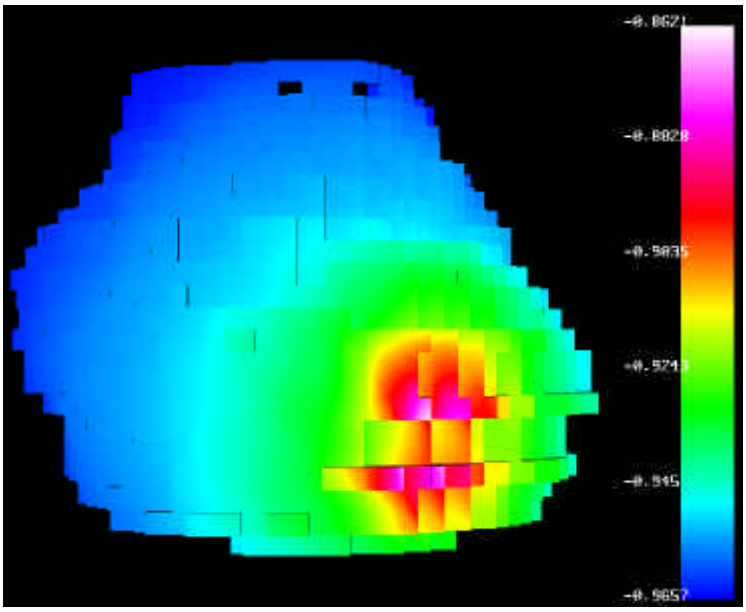

Figure 4-10: Right.

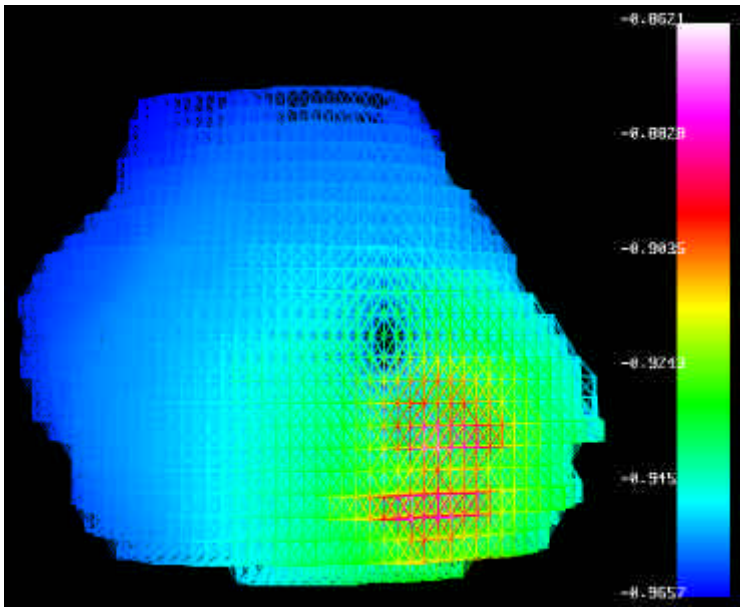

Figure 4-11 The volume presented as a triangular, tetrahedral and cubical mesh. The element edges can be colored according to any field variable.

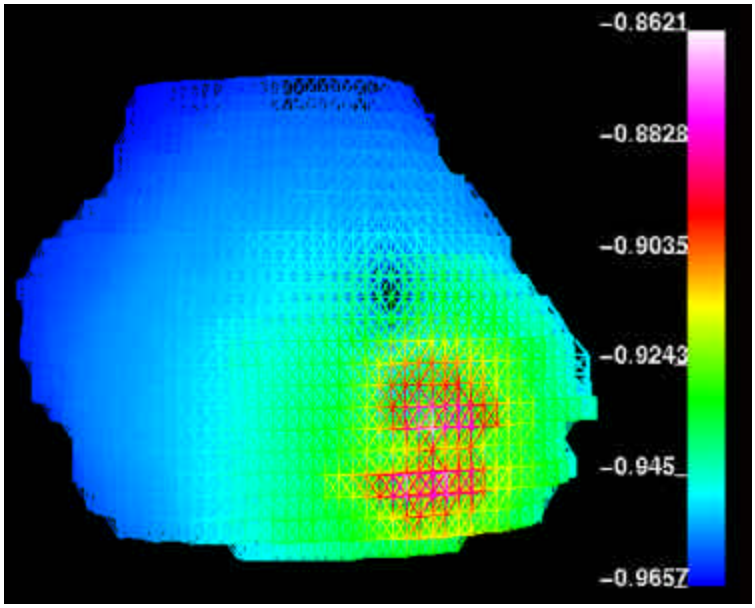

Figure 4-11: Middle.

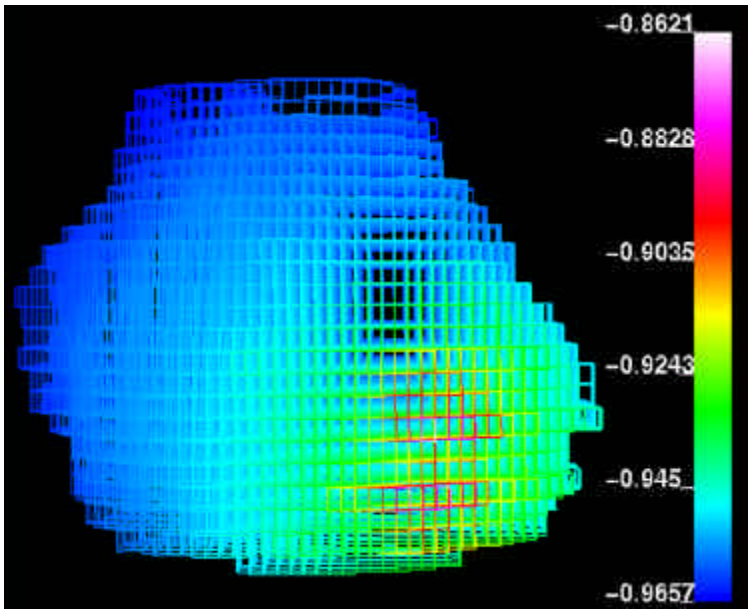

Figure 4-11: Right.

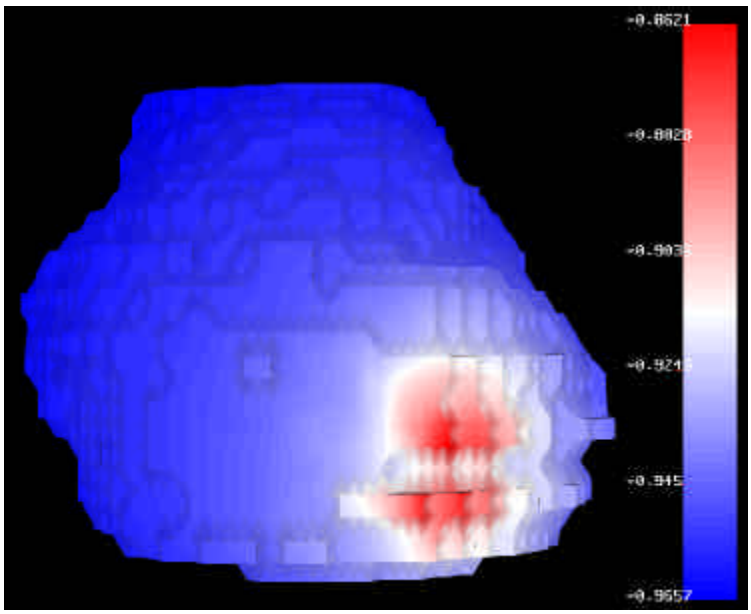

Figure 4-12 Left: The color map can be changed to suit the needs. A blue-white-red color map is used to present scalar potential on the heart's surface. Middle: Isocontours are plotted on the elements. The division between the contours is linear. Right: The heart volume has been cut with 4 isosurfaces of z-coordinates (axial). The formed cut planes are colored according to scalar potential.

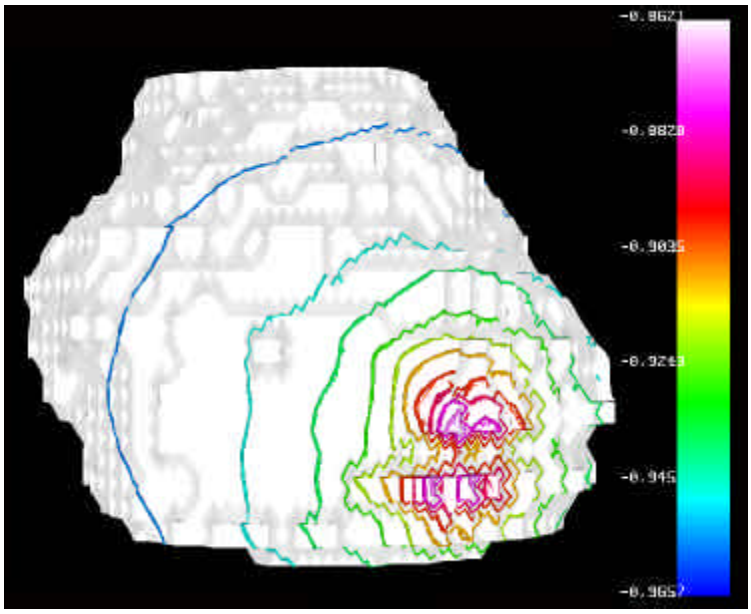

Figure 4-12: Middle.

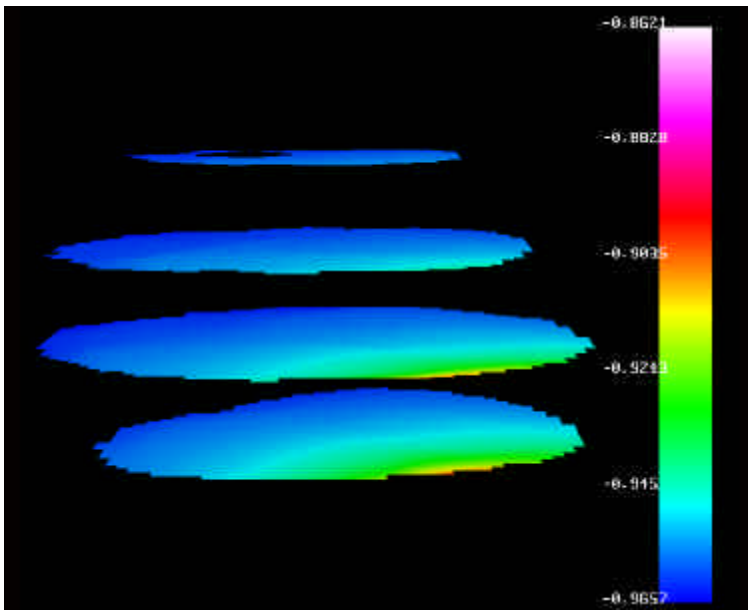

Figure 4-12: Right.

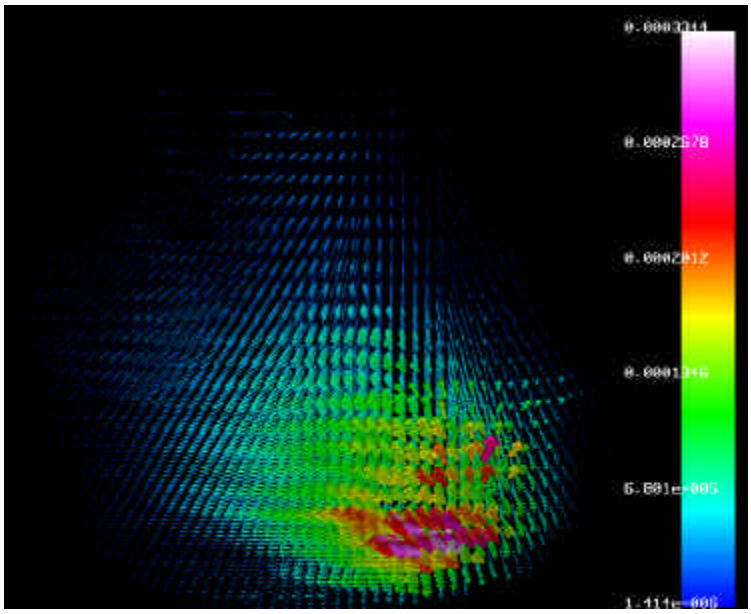

Figure 4-13 Visualization of current vectors in the heart volume. The vector length and coloring correspond to the absolute value of the current vector. Left: Vectors are drawn to all the nodes in the heart. Middle and right: Vectors are restricted to every 5th and every 10th node respectively.

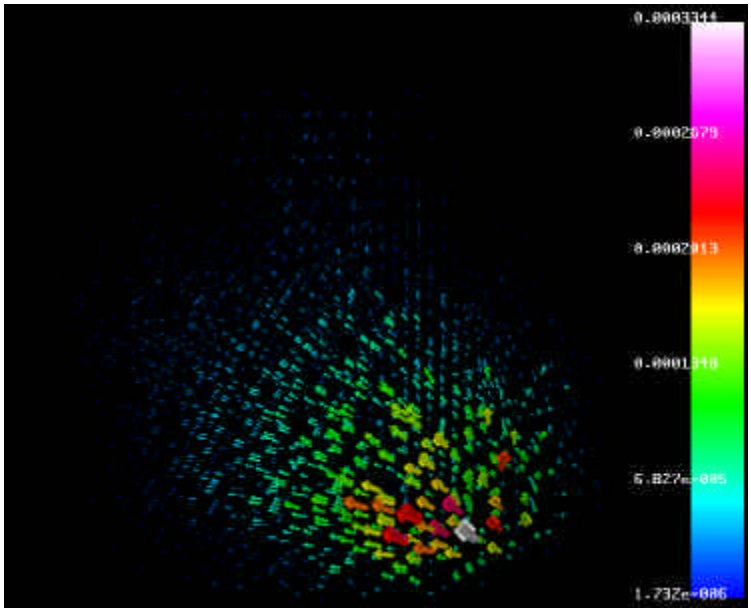

Figure 4-13: Middle.

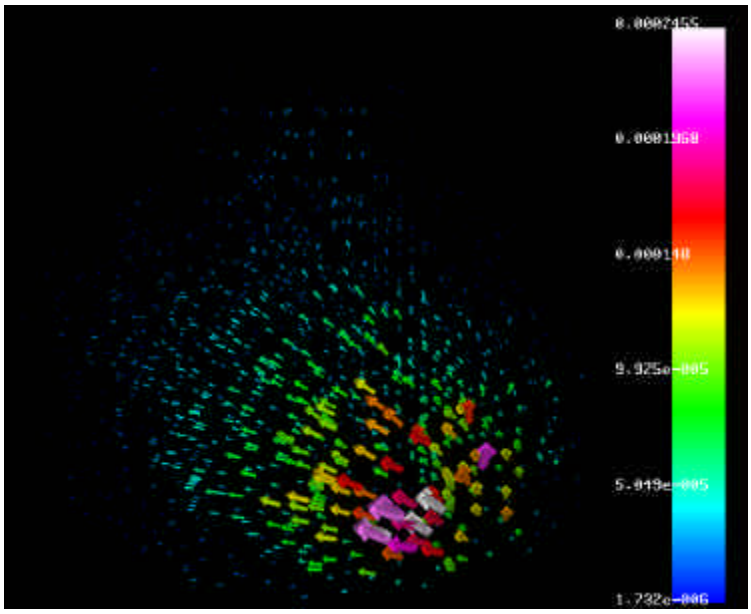

Figure 4-13: Right.

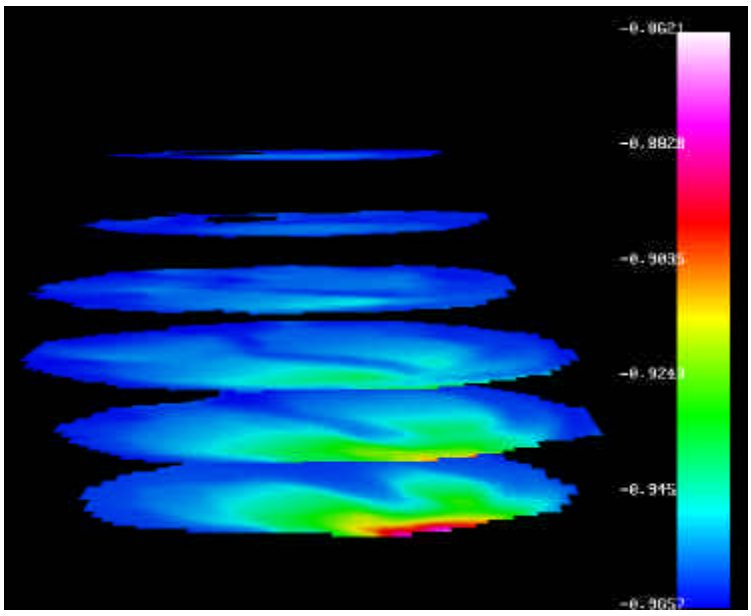

Figure 4-14 Left: The heart volume has been cut according to 6 isosurfaces of z-coordinates (axial). The planes are colored with the absolute value of the current vector. Middle: The rendered volume can be limited with object clip planes in the direction of the object's coordinate axes. This differs methodologically form the isosurface presentation. The cut volume is colored according to the absolute values of the current vectors. Right: The same clip plane as in the middle is used to present the current vectors. The length and the coloring correspond to the absolute value of the current vectors.

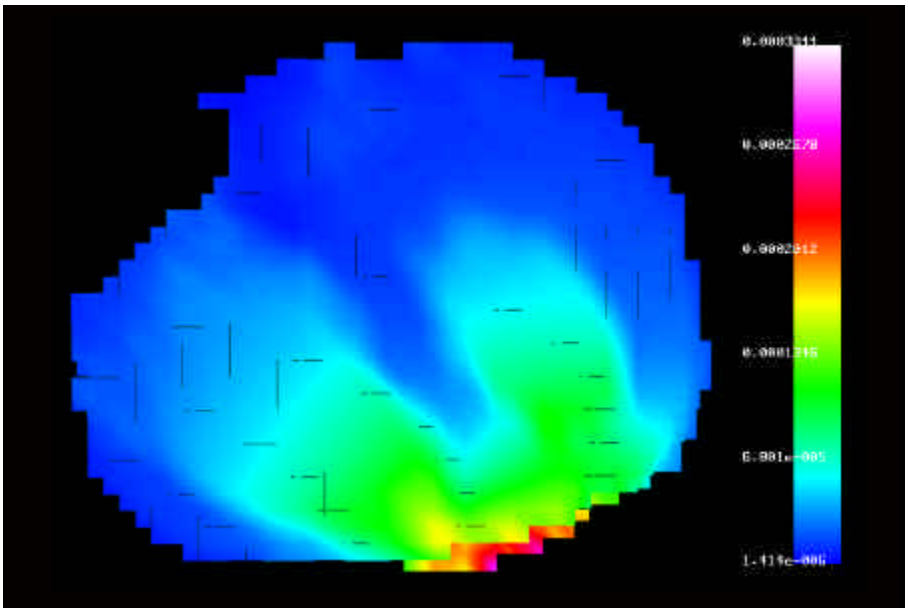

Figure 4-14: Middle.

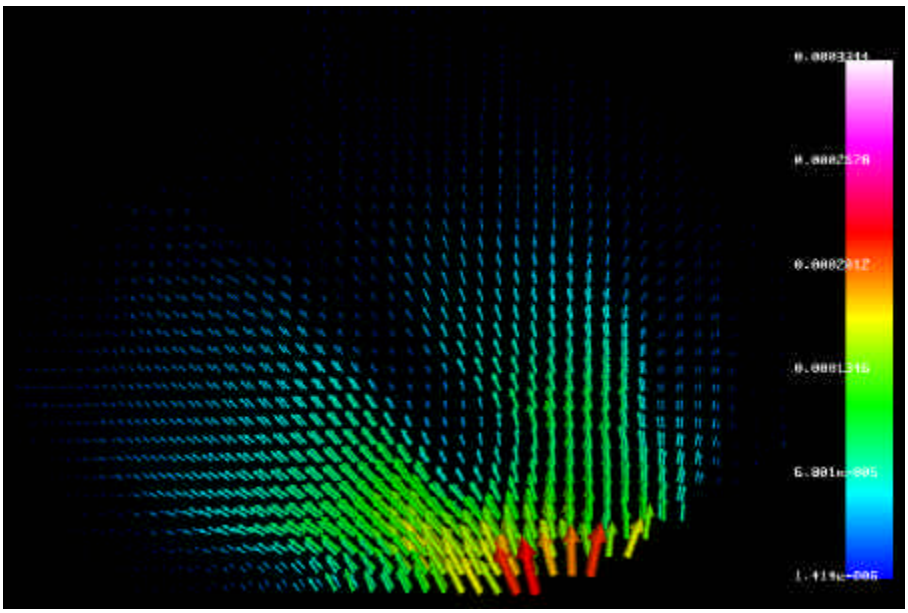

Figure 4-14: Right.

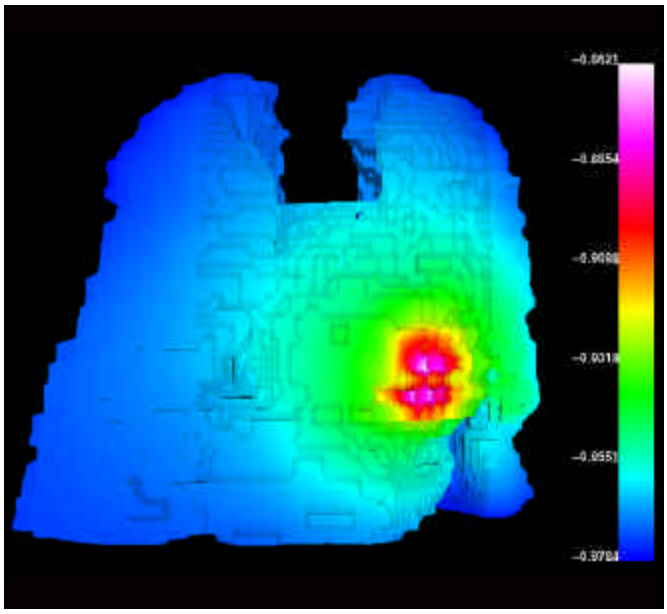

Figure 4-15 The heart and the lungs colored according to the scalar potential. When the elements are labeled by the tissue codes the user may choose to make some tissues transparent in order to examine the tissues behind or inside them. Here the heart is seen separated from the lungs.

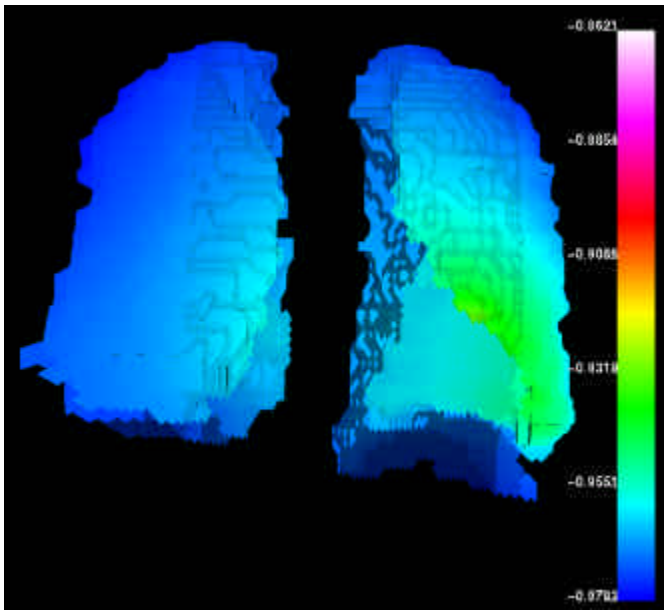

Figure 4-15: Middle.

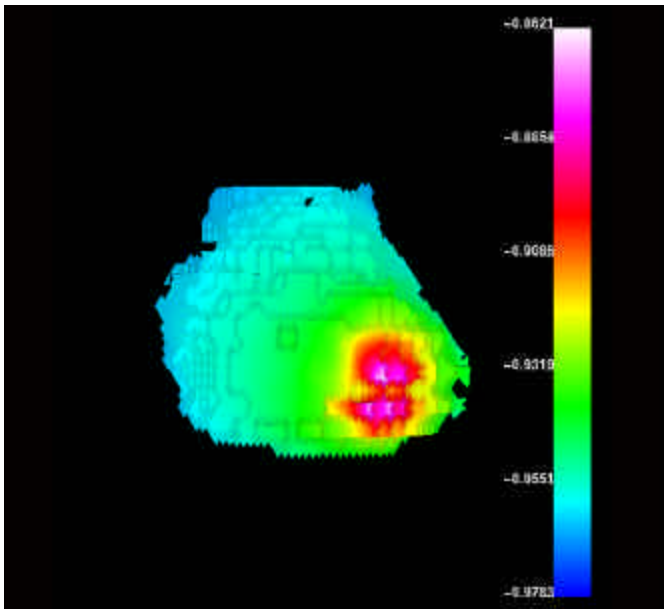

Figure 4-15: Right.

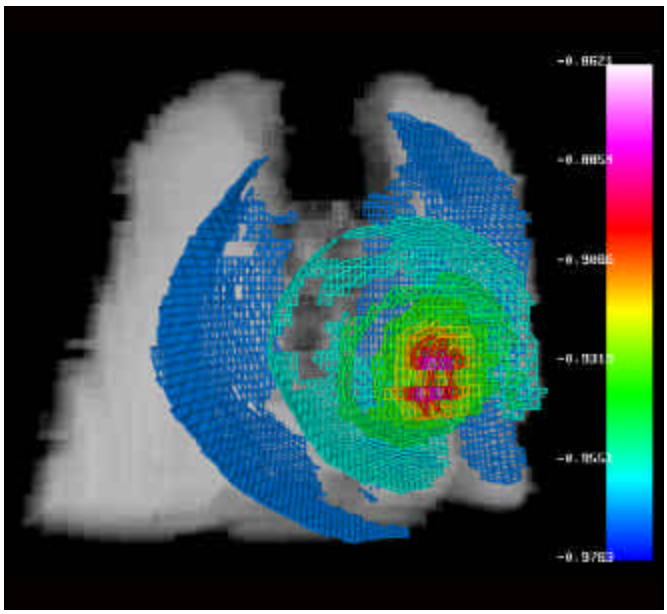

Figure 4-16 Fancier visualization. Left: The isopotential surfaces in the lungs and the heart are presented with meshes colored according to the scalar potential. The surface of the organs is illustrated as partly opaque for clarity. Middle: The same as the image on the left, but from axial direction. A cut plane made with object clip planes is shown partly opaque. Right: The same as in the middle, but current vectors are added. This is an example of bad visualization by putting too much data in one image resulting in confusion.

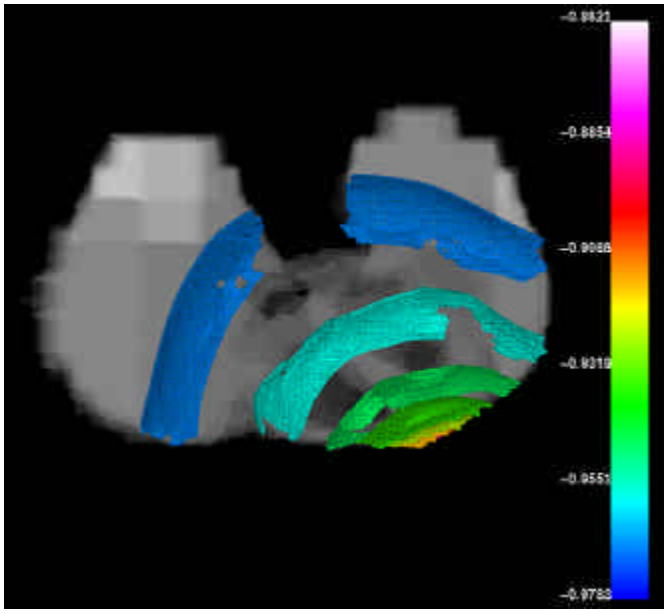

Figure 4-16: Middle.

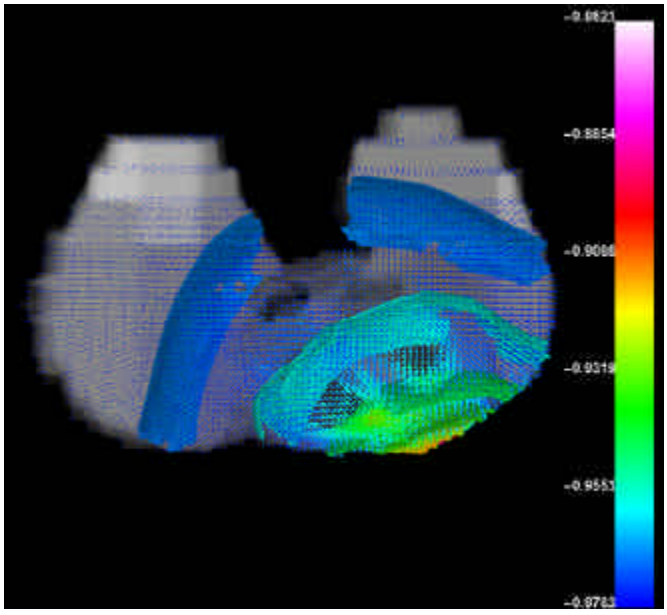

Figure 4-16: Right.

Appendix D

In the back cover of this thesis there is a pocket containing a CD-ROM, which is appendix D.

Contents of the CD:

- Master of Science Thesis as an Adobe Acrobat pdf-file
- Installation file of Adobe Acrobat Reader 5.0 for Microsoft NT/XP
- All the images presented in the thesis
- Additional images not appearing in the thesis about visualization of the thorax, and the behavior of bioelectric fields therein
- Cone file conversion script (`cone.py`)
- Installation files for Python interpreter 2.0.1
- Operating manual for Cone
- Some input files for Cone extracted using FDM-node pick-up and FDM-node reporter of the Noname Bioelectric Field Software
- Calculated output files of those input files

Some of the images of the thorax were created using SGI Onyx2 computer located in the Virtual Reality Center (VRC) of TUT. This was because of the limited performance of the workstation in visualizing models consisting of 10^5 nodes and more.