Strategies for Optimal Design of Biomagnetic Sensor Systems

D I P L O M A R B E I T

zur Erlangung des akademischen Grades

Diplom-Informatiker

FRIEDRICH-SCHILLER-UNIVERSITÄT JENA

Fakultät für Mathematik und Informatik

eingereicht von Stephan Lau

geb. am 28.03.1980 in Jena

Betreuer:

Prof. Dr.-Ing. habil. J. Haueisen
Fakultät für Informatik und Automatisierung
Technische Universität Ilmenau

Prof. Dr. E. G. Schukat-Talamazzini
Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena

Jena, June 29, 2007

Abstract

Magnetic field imaging (MFI) is a technique to record contact free the magnetic field distribution and estimate the underlying source distribution in the heart. Currently, the cardiomagnetic fields are recorded with superconducting quantum interference devices (SQUIDs), which are restricted to the inside of a cryostat filled with liquid helium or nitrogen. New room temperature optical magnetometers allow less restrictive sensor positioning, which raises the question of how to optimally place the sensors for robust field reconstruction.

The objective in this study is to develop a generic object-oriented framework for optimizing sensor arrangements (sensor positions and orientations) which supports the necessary constraints of a limited search volume (only outside the body) and the technical minimum distance of sensors (e.g. 1 cm). In order to test the framework, a new quasi-continuous particle swarm optimizer (PSO) component is developed as well as an exemplary goal function component using the condition number (CN) of the leadfield matrix. Generic constraint handling algorithms are designed and implemented, that decompose complex constraints into basic ones. The constraint components interface to an operational exemplary optimization strategy which is validated on the magnetocardiographic sensor arrangement problem. The simulation setup includes a three compartment boundary element model of a torso with a fitted multi-dipole heart model.

The results show that the CN, representing the reconstruction robustness of the inverse problem, can be reduced with our optimization by one order of magnitude within a sensor plane (the cryostat bottom) in front of the torso compared to a regular sensor grid. Reduction of another order of magnitude is achieved by optimizing sensor positions on the entire torso surface. Results also indicate that the number of sensors may be reduced to 20-30 without loss of robustness in terms of CN.

The original contributions are the generic reusable framework and exemplary components, the quasi-continuous PSO algorithm with constraint support and the composite constraint handling algorithms.

Keywords: Magnetic sensor setup, constraint optimization, swarm intelligence, magnetometer, magnetocardiography

2000 Math. Subject Classification: 90C31, 90C56, 15A12, 92C50, 92C55

Contents

Symbols, Abbreviations and Terms

ρ	In the spherical coordinate system, the distance from the origin to a given point
ϕ	In the spherical coordinate system, the angle between the positive z-axis and the line formed between the origin and a given point (zenith)
θ	In the spherical coordinate system, the angle between the positive x-axis and the line from the origin to a given point projected onto the xy-plane (azimuth)
$\vec{D}$	electric flux density
$\vec{E}$	electric field strength
$\vec{B}$	magnetic flux density
$\vec{H}$	magnetic field strength
B_j	measured magnetic flux density at jth sensor
$\vec{m}$	magnetic dipole
σ	conductivity
σ_S	conductivity of the volume containig the sources
σ_i	conductivity when the field is intracellular
σ_e	conductivity when the field is extracellular
σ_j^-	conductivity on the inner side of the surface S_j
σ_j^+	conductivity on the outer side of the surface S_j
σ_k^-	conductivity on the inner side of the surface S_k
σ_j^+	conductivity on the outer side of the surface S_k
μ_0	permeability of free space
I_0	total current of the point source at the origin
$\vec{J}$	electric current density
$\vec{J}_v$	volume current density
$\vec{J}_i$	impressed current density
$\vec{J}_i^V$	impressed current density (vortex field)
$\vec{J}_i^F$	impressed current density (flow field)
$\vec{K}_s^V$	secondary current density (vortex field)
$\vec{K}_s^F$	secondary current density (flow field)
Φ	scalar potential
Φ^+	potential due to positive point source
Φ^-	potential due to negative point source
Φ_0	potential from the source located at the origin
Φ_d	total dipole field
Φ_i	inner membrane potential
Φ_e	outer membrane potential
∇'	nabla operator at the source point
$\vec{e}_R$	unit vector in the radial direction
$\vec{e}_d$	unit vector in the dipole direction
$\vec{n}$	surface vector
$\vec{n}(\vec{r}')$	surface normal on S_j
$\vec{r}'$	source point
$\vec{r}$	field point
R	radial distance
$S_j,\ S_k$	surfaces
V	volume
dv'	volume element in source region
$Z,\ T$	coefficient matrices

V_j	potential difference at bipolar electric lead j
$\vec{L}_j^E$	electric lead vector field of bipolar lead j
$\vec{L}_j$	magnetic lead vector field of magnetometer j
L	magnetic lead field matrix
$\vec{d}$	dipole current density amplitudes of all dipoles
$\vec{s}$	magnetic flux density amplitudes of all magnetometers

λ_{max}	largest eigenvalue of a matrix
λ_{min}	smallest eigenvalue of a matrix
σ_{max}	largest singular value of a matrix
σ_{min}	smallest singular value of a matrix
κ_2	condition number based on L_2 norm
BEM	Boundary element method
CN	Condition number
CORBA	Common object request broker architecture
CREW	Concurrent read exclusive write shared memory model
CT	Computer tomorgram
ECG	Electrocardioogram
EEG	Electroencephalogram
EREW	Exclusive read exclusive write shared memory model
FEM	Finite element method
fMCG	Fetal magnetocardiogram
fMEG	Fetal magnetoencephalogram
HTS	High-temperature superconductivity
LTS	Low-temperature superconductivity
MCG	Magnetocardiogram
MEG	Magnetoencephalogram
MFI	Magnetic field imaging
MRI	Magnetic resonance imaging
PCA	Principle component analysis
PSO	Particle swarm optimization
SQUID	Superconducting quantum interference device
STL	Standard template library, a generic C++ library of container classes, algorithms and iterators
TS	Tabu search

#	'Number of', e.g. '# sensors' means 'number of sensors'
multiset	A set of non-unique elements, as opposed to a set
search space	The abstract, possibly high-dimensional and non-physical parameter space to be searched, as opposed to search volume
search volume	The three-dimensional, physical volume to be searched, as opposed to search space
set	A set of unique elements, as opposed to a multiset

1 Introduction

1.1 Biomagnetism

Biomagnetism is concerned with the measurement, analysis and interpretation of magnetic fields produced by electrically active living tissue [37]. The two main types of such tissue are nerve cells and muscles. Most of the biomagnetic research has been conducted on the activity of the brain (neuromagnetism). The heart, a muscle, and its excitation is the second area of broad research. Further topics of interest are the functional state of the intestinal system or sceletal muscles.

The biomagnetic fields produced by living tissue are very weak compared to fields of the environment (Figure 1). For example, the peak magnetic flux density on the torso surface close to the heart measures around 50 pT [94] and at the scalp of the head the field of the brain is in the range of fT up to 1 pT [37]. This is about 1 million times smaller than the earths magnetic field [98]. Typical disturbances for biomagnetic measurements are an elevator in the same building or the close by tram or subway line. Such measurements therefore require very sensitive technology and magnetic shielding.

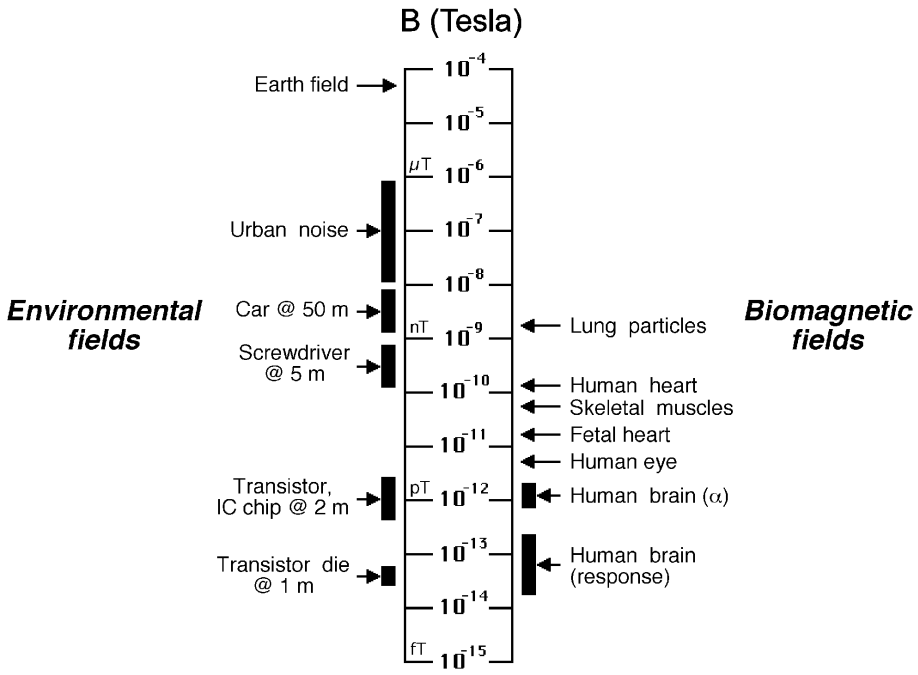

Figure 1: Comparison of biomagnetic fields and environmental noise [98]

Cardiac Field Sources The heart pumps blood through the body by rhythmic contractions of the heart muscle. These contractions are triggered by the self-excitatory sinus node (Figure 2), which is regulated by sympatho-vagal nerves. The stimulus of the sinus node is first transmitted through the fast excitatory system of the heart, which consists of the internodal pathways, the atrio-ventricular node, the bundle of His and the Purkinje fibers. At rest the muscle cells are polarized with a negative intracellular potential compared to the extracellular space [35].

If the stimulus from the excitatory system arrives, the muscle cells depolarize causing an action potential. Ions move across the cell membranes, the potential difference is inverted and a current flows. After 200-300 ms the cells repolarize gain, while being refractory. The heart muscle acts as a function syncytium due to the gap junctions. Therefore, the action potential is transmitted from cell to cell forming a wave front starting from the cardiac septum and the apex and moving to the base. Because the currents at the wave front are aligned, they sum up to a stronger current. These primary currents and the secondary volume currents caused by the primary fields produce a measurable electric and magnetic field (Figure 3) which give valuable insight into the functional condition of the heart [57].

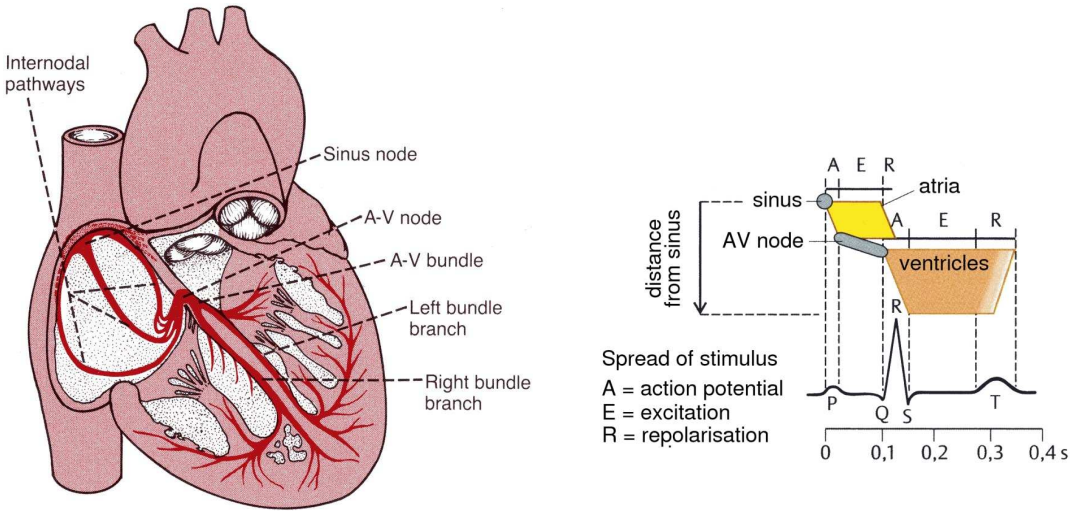

Figure 2: Sinus node and Purkinje system of the heart (left) [35, p. 112] and relation between excitation and ECG in sinus rhythm [16, p. 201]

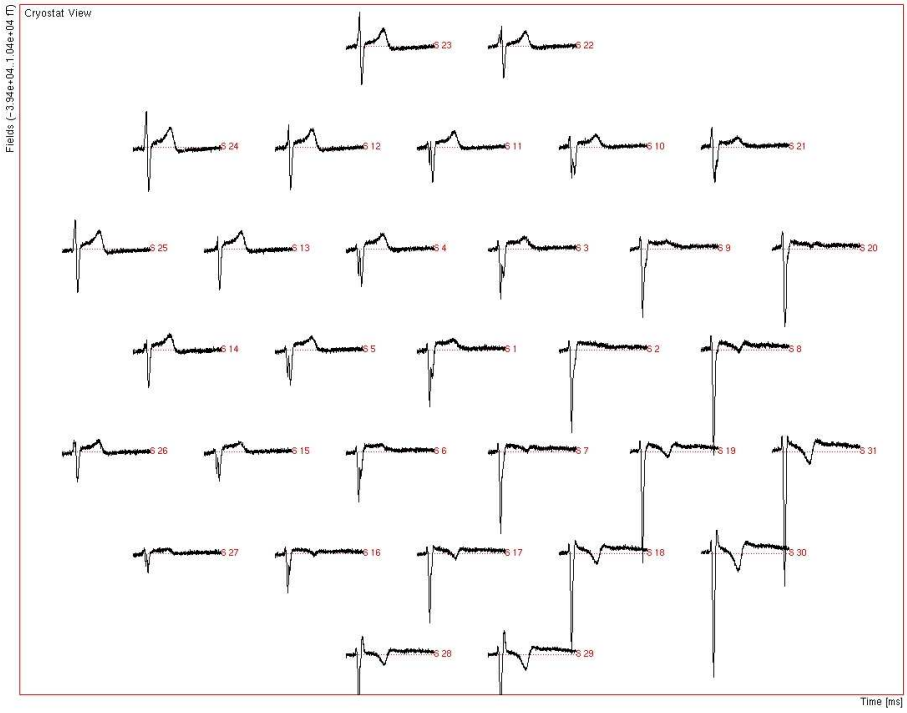

Figure 3: 31-channel MCG of 1000 ms. The sensors are arranged on the body surface above the heart. The feet are in the direction to the left.

Field Measurement and Analysis The electrical current flows produce an electric field which propagates through the human body. The conductivity of the body compartments moderates this propagation. Non-invasively, the resulting potentials on the body surface can be measured with electrodes, producing an Electrocardiogram (ECG) or an Electroencephalogram (EEG). This technique has been established in clinical practice and standards of electrode positioning have been developed. For example, an anteroseptal myocardial infarction can be differentiated from an anterolateral infarction by the particular ECG lead positions which show prominent infarction patterns, in this case V_2 and V_3 and not V_5 and V_6 [21] (Figure 4).

The electrical currents, consisting of moving charges, produce a magnetic field which today is measured contact-free with superconducting quantum interference devices (SQUIDs) [37], producing an Magneto-

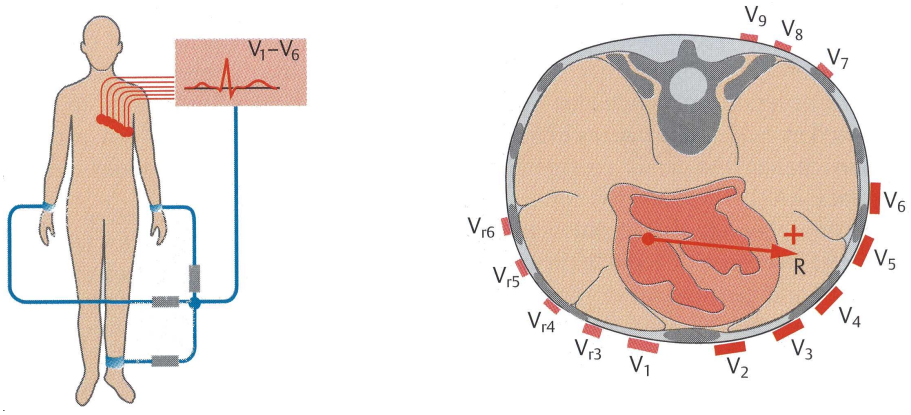

Figure 4: Wilson chest leads [16, p. 199]

cardiogram (MCG) or Magnetoencephalogram (MEG). In order to maintain the superconducting property these sensor devices need to be cooled by liquid helium. New technologies, such as optically-pumped magnetometers [6], however will be more flexible to build and much easier to maintain. The magnetic sensor locations are not standardized, rather each biomagnetic system has custom types of sensors usually arranged regularly inside a dewar. An advantage of measuring the magnetic field instead of the electric one is that the influence of the propagation medium, the human body, is weaker. The question of whether the MCG, for example, contains more information than the ECG has caused much controversy. It was shown that the amplitudes and morphological patterns in the MCG are different [86] and the magnetic lead fields can not be produced by any combination of electric lead fields [75].

The goal of the analysis is not so much to categorize the patterns in the individual channels, but to locate the electric current source inside the body. The multiple channels around the body can be understood as spatial sampling of the three-dimensional field produced by the source. This requires two models: (1) the parameterized source model and (2) the volume conductor model, which describes how the electric current propagates through the body. The forward problem is to compute the magnetic flux density of a given source at the sensor locations. The inverse, typically ill-posed, problem is to find parameters of the source model, that explain the sensor readings.

1.2 Magnetic Sensor Technologies

The magnetic field strength or flux density can be measured indirectly through the effects it has on certain materials under certain conditions. Thus, there is a range of detection methods, which differ in their magnetic flux density range and frequency range which can be detected reliably. For biomagnetic purposes the minimum detectable magnetic flux density must be less than $1\ nT$. A concise overview of techniques capable of measuring in this range is included in [37], which is summarized in this section. Figure 5 summarizes the respective sensitivities.

Coil-based Magnetometer A coil with one or more windings is positioned in the magnetic field. The time-variant magnetic field causes a proportional electromotive force in the coil according to the law of electromagnetic induction. The coil picks up the field component parallel to the normal vector on the coil area, so it is a vectorial sensor.

Anisotropic Magnetoresistive Effect Some anisotropic ferromagnetic thin films have an electric resistance that changes with the angle between the current density vector and the spontaneous magnetization. These sensors are vectorial as well.

Fluxgates Fluxgates consist of magnetically soft cores around which two coils are arranged in opposite orientation. A third field coil drives the cores into saturation periodically. If no external field exists, both pick up coils cancel out each other. If a field exists, the vectorial component parallel to the cores produces a deviation between both pick up coils.

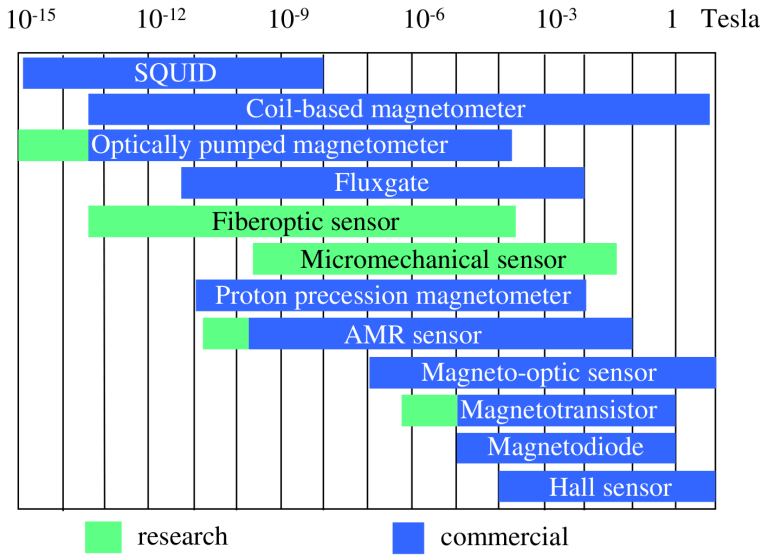

Figure 5: Current sensitivities of magnetic sensor technologies [71]

Proton Precession Magnetometer Just like magnetic resonance imaging (MRI), this magnetometer type is based on the observation that the precession frequency (Larmor-frequency) of protons (hydrogen) changes proportionally to the strength of an external magnetic field. Because the spin of protons is evenly distributed, their very small magnetic moments cancel out each other. Therefore, the spin axes are aligned with a strong homogeneous magnetic field. Once this field is switched off, the spin axes start scatter. The changing magnetic field causes an electromotive force in a pick up coil with the Larmor-frequency. Proton precession magnetometer only measure the scalar magnetic flux density.

Optically Pumped Magnetometer Optically pumped magnetometer utilize the Zeeman effect. Helium or an gaseous alkali metal (opaque), such as cesium or potassium, are polarized by circularly polarized light of a wave length equal to the difference between two energy levels E_a and E_b. Electrons, whose spins are aligned with the rotational direction of the light, are shifted from the base level E_a to the instable level E_b. These electrons fall down to an energy level $E_c < E_a$ spontaneously while emitting a photon and changing their spin. Because $E_a - E_c < E_b - E_a$, the electrons can not absorb any new photons. Because E_c is more stable than E_b a balanced situation is reached where E_c is full and E_a is depleted. The pumped metal can not absorb any more photons, becomes transparent and the light is detected on the other side of the glas container by a light sensor. In the second phase, the gaseous metal is depolarized by RF-radiation with a frequency of $E_a - E_c$ (Larmor frequency). The electrons are shifted to E_a and the gas becomes opaque again. An external magnetic field spreads E_c into several sublevels each having a distinct Larmor frequency (Zeeman effect). This frequency shift is proportional to the scalar magnetic flux density.

Micromechanical Sensors Micromechanical sensors consist of a pivoted permanent magnet on top of a base. This micromechanical torsion element is turned by an external magnetic field by a certain angle with a certain force. Two electrodes at the bottom of the torsion element and on the base represent a capacitor. A tunneling electrode at the base allows quantum tunneling, which has properties that are proportional to the magnetic flux density. Vectorial field components can be measured.

Faraday-Effect-based Sensors The Faraday-effect refers to the observation that plane of polarization of light is rotated inside optic fibre that is exposed to a magnetic field. The rotation angle is proportional to the strength of the component of the magnetic field that is aligned with the light beam. Of course, the magnetic flux density range is limited by the saturation threshold of the material.

Magnetostrictive Fiber-optic Sensors Some materials, such as metallic glas, change their physical dimensions under exposure of a magnetic field. This can be utilized by attaching an optic fiber to it, for

example by coating the fiber with metallic glas or winding the fiber around a metallic glas core. The change in length of the fiber causes a phase-shift of the light passing through it which is proportional to the magnetic flux density.

SQUID Sensors Superconducting quantum interference devices (SQUIDs) are the current standard sensors in biomagnetic instrumentation. Superconductivity is the property of certain materials to loose their electrical resistance below a certain temperature threshold. A SQUID sensor consists of a superconductive ring in which one (rf-SQUID) or two (dc-SQUID) weakly superconductive connections (Josephson junctions) are inserted (Figure 6). The electrons pair up to so called Cooper pairs, which all have the same energy. Both electrons have opposite impulse and spin and thus do not interact with the grid. The electron pairs condense into a single quantum state, which can be described through a macroscopic wave function. At the Josephson junctions, interference phenomena can be observed. Cooper pairs tunnel through the barriers, causing a phase shift in the quantum-mechanical wave function. An external magnetic field is compensated inside the superconductor (Meissner-Ochsenfeld- Effect) through a compensational current. This current characterizes magnetic field strength. Additionally, the magnetic flux is quantized and the number of quanta characterizes the magnetic field strength as well. In practice, the SQUID is included in a closed loop which keeps the flux inside the SQUID (flux locked mode) constant or the compensation current in the flux transformator (current locked mode)

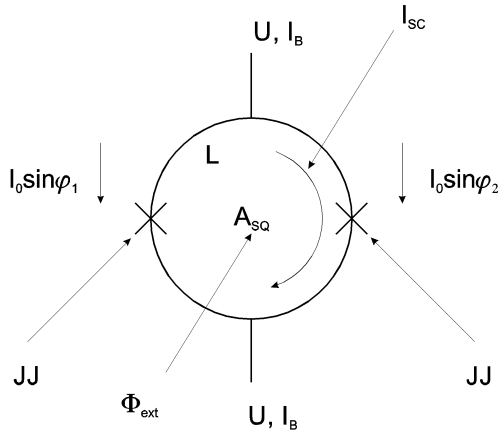

Figure 6: Scheme of a dc-SQUID consisting of two Josephson junctions(JJ) in a superconducting loop of inductance L and area A_{SQ} [2, p. 105]

In order to efficiently inject the magnetic flux into the SQUID, antennas in the shape of coils are used. A magnetometer can be implemented as shown in Figure 7 on the left. A gradiometer consists of two loops with opposite orientation (Figure 7 right). An external magnetic field affects both coils equally and is thus compensated. However, if the pick-up coil is close to the measured object, the influence on it will be stronger than on the distant reference coil, which results in a flux. The main advantage of gradiometers is the noise reduction.

The current industrial standard are integrated chips, such as the one in Figure 8. In this example from Elekta Neuromag© two planar gradiometers measuring $\triangle B_z/\triangle x$ and $\triangle B_z/\triangle y$ are combined with a magnetometer measuring B_z. While the gradiometers have a focal sensitivity and a noise density of $2.7\ fT/cm/\sqrt{Hz}$, the magnetometer has a widespread one and a noise of $3\ fT/\sqrt{Hz}$ (white noise) [54, 2]. The magnetometer dimensions are 2.1cm x 2.1cm.

A disadvantage of the SQUID technique is that the sensors need to be cooled below the superconduction threshold. Typically, this requires them to be surrounded by liquid helium (low temperature SQUIDs) or liquid nitrogen (high temperature SQUIDs). This is expensive and requires periodic refilling.

Current Research Optically pumped magnetometers are under active development. Recent advances have been made in the sensitivity, which now reaches that of conventional SQUIDs [10] and is expected to reach $10^{-17}T/\sqrt{Hz}$ [10]. The initially scalar optic magnetometers can be operated to obtain vectorial field information. One vectorial component is acquired by adding a bias field to the sensor in that direction. This

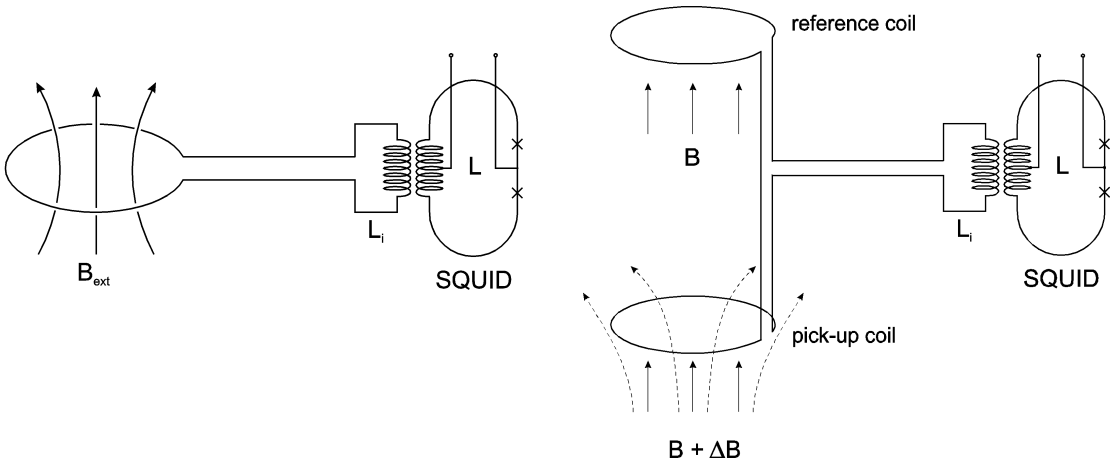

Figure 7: Scheme of a flux transformer coupled to a SQUID (left) [2, p. 107] and a first-order axial gradiometer (right); L_i = inductance of the input coil, L = inductance of the SQUID [2, p. 114]

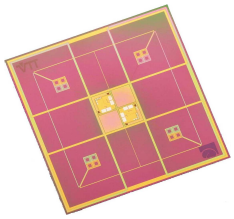

Figure 8: Single triple-sensor thin-film element manufactured on a silicon chip using a photolithographic technique [2, p. 133]

is done consecutively in all three dimensions. The band width of the magnetometer can be adjusted with the laser beam intensity. An important criterion is the spatial resolution of the magnetometers which is limited to their size. Newest technologies allow a sensor length in the millimetre scale [10].

The cost of a biomagnetic measurement system can be reduced significantly with optical magnetometers, because they do not require cooling. For magnetocardiography, cesium vapor cells can be operated at 30° C [5, 6, 51]. For magnetoencephalography potassium cells have been successfully operated at 180° C [102]. This requires heating, which is however significantly cheaper and easier to maintain than helium cooling.

A recent review on advances in SQUID technology is [23]. The evolution of coil-based magnetometers is discussed in [96]. The current state of solid-state magnetometers (magneto-resistance devices) is described in [82].

1.3 Magnetic Sensor Arrangements

For biomagnetic measurements a number of different sensor arrangements exist. Theoretically, the sensor arrangement is a spatial sampling of the continuous magnetic field around the measured organ. Therefore, without knowing the nature of the field to be measured, the general design goal is to place many sensors as close as possible to the measured organ. For particular biomagnetic problems, such as magnetocardiography, it is possible to determine the confidence in a particular sampling width [50].

In practice, a number of constraints need to be considered. The sensors have a certain size, so they can not be put arbitrarily close. In SQUID Systems, the sensors have a fixed configuration inside a crystat, also called dewar. Therefore, the sensor arrangement can only be customized to suit a particular body part, such as the head, during the construction phase. The dewar needs to be customized to this setup. New systems, which do not require cooling, will allow a much more flexible positioning of the sensors.

Another design issue is whether to support scalar or vectorial representations of the field. For vectorial recordings it is then desirable to still be able to obtain B_z. Many sensor techniques support vectorial

acquisition. An example of a vectorial system is the AtB Argos 200 system (Figure 9). Three orthogonal components are measured in this case. In this section examples of different types of sensor arrangements for different applications are introduced to illustrate concrete design implementations.

Magnetocardiography (MCG) For cardiologic investigations, the sensors need to be positions on the body surface close to the heart. Typically, a flat, slightly bent or slightly concave layer of sensors is arranged at the bottom of a cryostat with the same shape. An example is the AtB Argos 200 system (Advanced Technologies Biomagnetics (AtB), Chieti, Italy) in Figure 9. The cryostat has a flexible mounting to provide a limited positioning flexibility towards the heart. The cryostat can only be tilted by a limited degree of no more than about 35-40° because of the liquid helium inside.

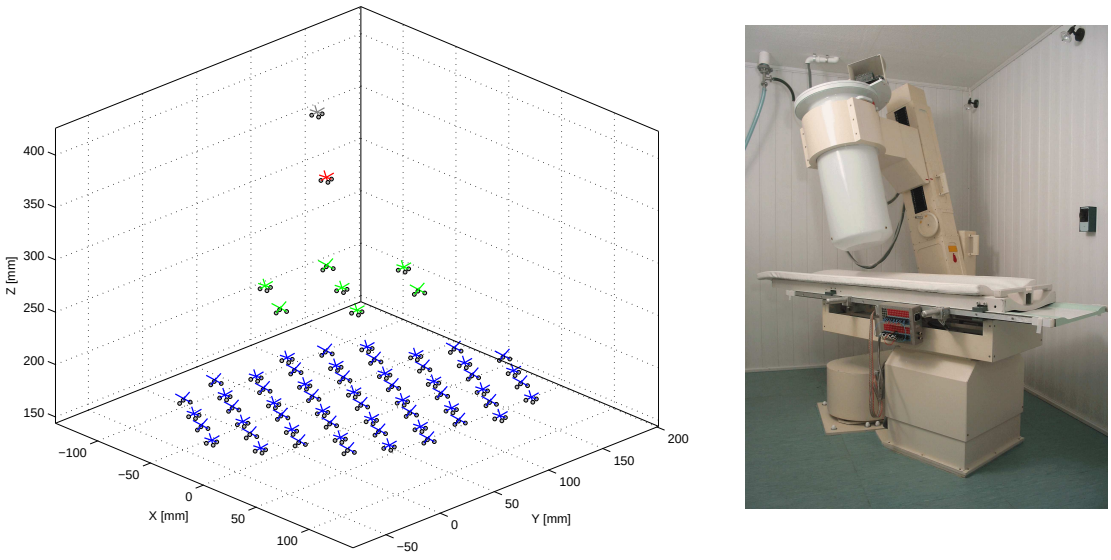

Figure 9: Multi-layer sensor arrangement of the AtB Argos200 system for MCG (points indicate sensor centers, lines indicate the normal vector on the sensor plain, layers are color-coded) and gantry for cryostat positioning relative to human body

The Argos 200 system in Figure 9 has 56 magnetometer triplets in the measurement layer. The magnetometers are as close to each other as possible, while truely measuring orthogonal field components [70]. The sensors have a square area of 8x8 mm each and are arranged on three sides of a cube which are adjacent to a corner which is pointing towards the heart in z-direction. The noise level is $\leq 7fT/\sqrt{Hz}$ at 10 Hz. Reference magnetometers used for software noise reduction are arranged in three distant layers of 9.8 cm, 19.6 cm and 25.5 cm from the measurement plane. The cryostat is composed of fiberglas and has a diameter of 23 cm. The distance of the closest layer to the surface of the cryostat is 1.8 cm.

Fetal magnetocardiography (fMCG) is also an active field of research [33]. Fetal MCGs can also be recorded with MCG systems by positioning the sensor array on the abdominal surface. The orientation of the fetus can be determined with prior sonography. A standard for fMCG procedure has been developed [33]. The magnetic field of the fetus and the mother are superposed and are separated during data analysis.

Magnetoencephalography (MEG) A number of commercial MEG systems exist. An example is the Magnes 3600 system (4-D Neuroimaging, San Diego, USA) in Figure 10. Typically, the sensors are arranged around the head, extending below the temporal lobe and the occipital lobe. Reference sensors for noise reduction are positioned at a certain distance to the measurement sensors. The Magnes 3600 cryostat can be equipped with 248 MEG sensors, which can be all magnetometers or combinations with gradiometers. The average inter-sensor distance is 2.2 cm. After noise cancelation white noise is below $5fT/\sqrt{Hz}$ [99].

A second example is the Elekta Neuromag© helmet-shaped MEG (Elekta Neuromag Oy, Helsinki, Finland) with 102 sensor components (Figure 8), each containing a magnetometer and two gradiometers. The sensor arrangement is shown in Figure 11. The sensor positions are derived from anatomical data.

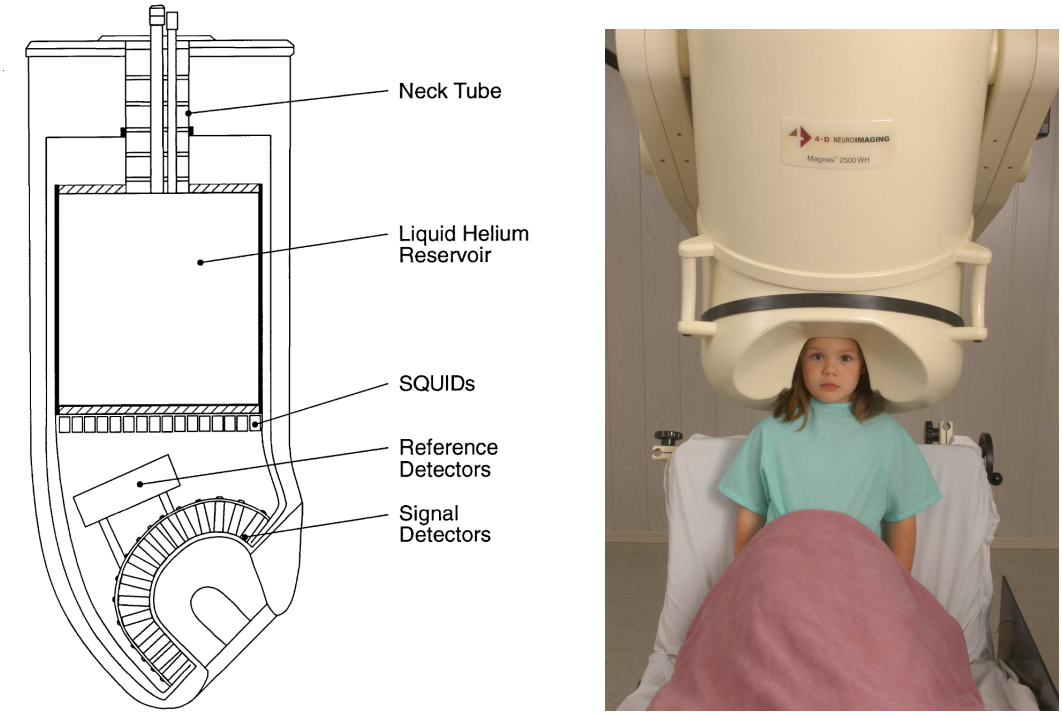

Figure 10: Magnes 3600 WH dewar as cutaway drawing (left) showing the location of the reference channels relative to the sensor coils and its positioning for a seated study (right) [2, pp. 120-122]

Parallel EEG recordings with 64 or 128 electrodes are supported. Sampling rates up to 10kHz are possible. The cryostat and mounting are depicted in Figure 11.

Fetal Magnetoencephalography (fMEG) can be conducted with non-helmet devices, such as the Argos 200 system or the 62-channel twin-cryostat system by Philips (Philips, Hamburg, Germany) [88]. In the Philips system each of the two cryostats holds a concave sensor array, which can be positioned independently on the abdominal surface of the mother or the head surface of an adult or child. A customized fMEG system is the CTF MEG^{TM} (VSM MedTech Ltd., Vancouver, Canada) in Figure 12. The sensors are arranged to shape around the abdominal surface of the mother, covering the greatest possible area [84]. The sensors are axial first-order gradiometers with a base line of 8 cm.

General Purpose Setups A general purpose sensor setup has been developed at the Physikalisch- Technische Bundesanstalt (PTB) in Berlin for use in the second Berlin Magnetically Shielded Room (BMSR-2) [8]. The design goals were to be able to measure three components of the magnetic field at as many positions in the cryostat volume as possible [87] and to be able to decompose it into identical sub modules. The resulting 4-layer setup with 304 SQUIDs in Figure 13 accomplishes that with a small number of sensors.

In order to measure all three components in one position, one sensor can be placed in the center of each side of a cube. Opposite sensors can be used to determine the vectorial component. In the PTB setup, several layers of cubes are stacked, which enables reuse of the sensors. Because the field decays rapidly with the distance, the second layer is close to the first one, while the others have a larger distance. Thus, we have cuboids. Further, the opposing sensors can be moved away from the centers of the sides, as long as the connection line passes through the cube center.

Using these design flexibilities, the setup was modularized into 19 vertical modules, each containing 16 SQUIDs. Each module is part of every layer. The modules are arranged parallel as a bundle. The lowest layer is a hexagonal grid of 57 SQUIDs and base length 2.9 cm. The second layer contains a grid of four SQUIDs per module and the third and fourth layer contain 3 SQUIDs per module.

The biomagnetic systems presented in this chapter are selected examples of different types of sensor arrangements. For a comprehensive overview of current systems see [2].

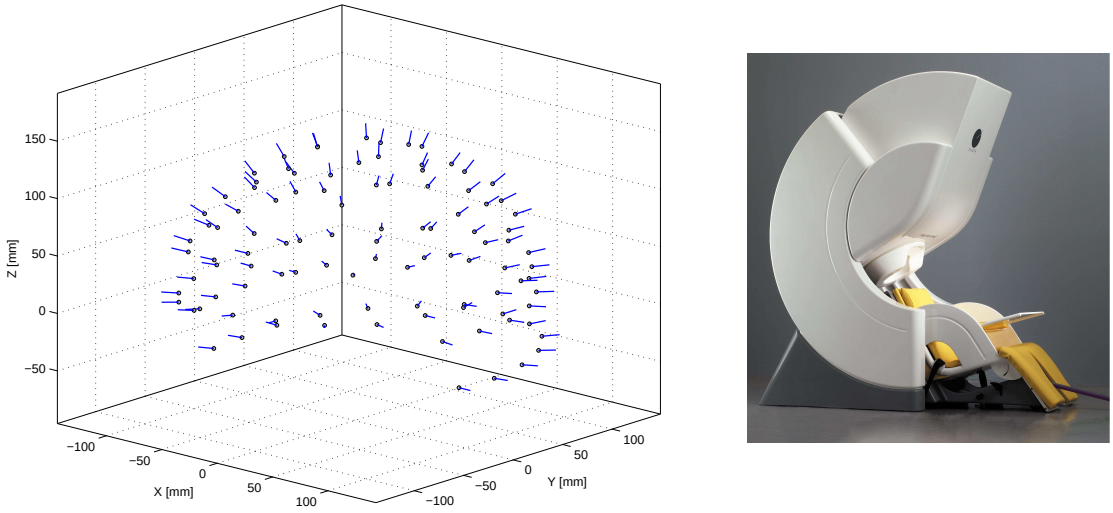

Figure 11: Helmet-shaped sensor arrangement of the Elekta Neuromag© MEG (points indicate sensor centers, lines indicate the normal vector on the sensor plain) and technical realization

Figure 12: CTF MEG^{TM} system configuration for fetal MEG array [2, p. 130]

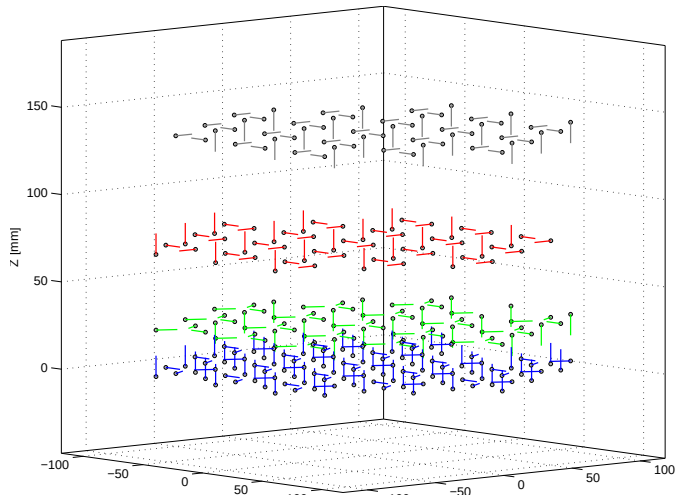

Figure 13: Modular general purpose sensor arrangement at the PTB (points indicate sensor centers, lines indicate the normal vector on the sensor plain, layers are color-coded)

1.4 Numerical Modeling

1.4.1 Overview

The mathematical base for modeling the propagation of magnetic field inside and outside a volume conductor has been described in the 1970s [29, 34, 30, 15]. In this section, the notion of a dipole is formally introduced, the assumptions underlying forward and inverse solution and their implications are described and the leadfield matrix is defined. The formal derivation follows most closely to Chapter 2 of [94], because it is concise and yet exact. Relevant details from original publications have been added in several instances.

1.4.2 Source Models

The simplest, and in physics well-described, electro-magnetic source model is an electric dipole in infinite space of homogeneous conductivity. To describe the nature of a dipole, we should start with a monopole.

Monopole Consider a very thin and very long insulated wire with one of its bare tips at the origin extending till infinity. Since the wire is very thin, its influence can be neglected. This construction approximates a point-like current source at the origin, a monopole. The electric field strength around the monopole is given by:

$$\vec{E} = \frac{I_0}{4\pi\sigma R^2}\vec{e}_R \; = \; -\nabla\Phi \tag{1.1}$$

where σ is the conductivity, I_0 is the magnitude of the monopole at the origin, $\vec{e}_R$ is a radial unit vector and R is the distance to the point source. As Plonsey and Heppner [77] pointed out, the low frequency (below 1kHz) time-varying electrophysiologic fields can be treated in the same way as static fields under certain conditions. The electric field strength can therefore be defined as the gradient of a scalar potential Φ. Considering that for large distances the field only decays in radial direction, the gradient can be expressed through the radial derivative:

$$-\frac{d\Phi}{dR}\vec{e}_R = \frac{I_0}{4\pi\sigma R^2}\vec{e}_R \tag{1.2}$$

We are interested in the potential Φ, so we integrate with respect to R and yield:

$$\Phi = \frac{I_0}{4\pi\sigma R} \tag{1.3}$$

Equation 1.3 describes the scalar potential generated by a monopole in a homogeneous infinite medium at any point with distance R from the monopole. The potential field generated by a set of monopoles is simply the superposition of the individual fields:

$$\Phi = \sum_i \frac{I_i}{4\pi\sigma r_i} \tag{1.4}$$

where I_i is the magnitude of source i and r_i is the distance.

Dipole A dipole is defined as two monopoles of same magnitude but with opposite sign, which are infinitely close to each other. The dipolar potential field can thus be described as a superposition as in Equation 1.4. Consider a dipole located at the origin of a cartesian coordinate system aligned with the z-axis. Let the positive charge of magnitude I_0 be $d/2$ in positive z-direction and the negative charge with magnitude $-I_0$ be $-d/2$ in negative z-direction. In order to produce an interpretable definition of the dipolar potential Φ_d we describe the potential of the positive charge Φ^+ and of the negative charge Φ^- through their first order taylor expansion at the origin:

$$\Phi^+ = \Phi_0 + \frac{d}{2}\frac{\partial\Phi_{(z')}}{\partial z'}\bigg|_{z'=0} \qquad \Phi^- = -\left(\Phi_0 - \frac{d}{2}\frac{\partial\Phi_{(z')}}{\partial z'}\bigg|_{z'=0}\right) \tag{1.5}$$

This means that the potential of the shifted source is approximated linearly by the sum of the potential of the same source at the origin and the deviation resulting by shifting it by $\pm d/2$ along the z-axis. This is sufficient since d is very small ($d \to 0$). The total dipolar field is then:

$$\Phi_d = \Phi^+ + \Phi^- = d\frac{\partial\Phi_{(z')}}{\partial z'}\bigg|_{z'=0} \tag{1.6}$$

We generalize from a dipole at the origin aligned with the z axis to a dipole located at any position and oriented in any direction by taking the scalar product of the potential gradient ∇' at the position (x', y', z') and a unit vector $\vec{e}_d$ in the dipole direction:

$$\Phi_d = \nabla' \Phi|_{x'=y'=z'=0} \cdot \vec{e}_d d \tag{1.7}$$

Now we can substitute with Equation 1.3 and bring the constants to the front:

$$\Phi_d = \frac{I_0 d}{4\pi\sigma} \nabla' \left(\frac{1}{R}\right) \cdot \vec{e}_d \quad = \quad \frac{1}{4\pi\sigma} \nabla' \left(\frac{1}{R}\right) \cdot \vec{m} \quad = \quad \frac{1}{4\pi\sigma} \frac{\vec{e}_R \cdot \vec{m}}{R^2} \tag{1.8}$$

Note that since $d \to 0$, we require $I_0 \to \infty$ for the potential to remain finite. Therefore, the product $I_0 d\vec{e}_d$ is combined to the vectorial dipole moment $\vec{m}$. As a third step, the gradient can be applied under the assumption $R \neq 0$. The vector $\vec{e}_R$ points from the source position (x', y', z') to the field position (x, y, z).

The monopole and the dipole are actually only the first and second order terms of a general multipole expansion described formally in electrodynamics. For an introduction to general multipole expansion see [43] and for a magnetocardiography view see [94]. In reality, a biological source is not a single dipole. Therefore several source models are used. We differentiate between focal and distributed source models. Focal models are the single dipole, the superposition of dipoles and the superposition of multipoles. Distributed models are regular distributions of dipoles, surface and volume sources and potential layers [37]. The most common source models are single and multiple dipoles and distributed dipoles. Indeed, the dipole is the basic source element of excitable tissue [94, 76].

1.4.3 Forward Problem

The electric and magnetic fields of the body are produced by transmembrane ionic flows. The primary fields consist of two types. In the intracellular and extracellular volume the electric field $\vec{E}$ is associated with a current flow $\vec{J}$ obeying Ohm's Law ($\vec{J} = \sigma\vec{E}$) [76]. This part of the tissue is passive. In the membranes however, diffusion and active transport mechanisms [35] generate active transmembrane currents, which locally depolarize the membrane. This current is called the impressed current $\vec{J}_i(\vec{r})$ as opposed to the passive volume current $\vec{J}_v(\vec{r})$. Between the depolarized area and the non-depolarized area of the membrane a potential difference is generated, which causes an intracellular current along the membrane. These microscopic currents in many parallel muscle fibres or neuron dentrites sum up to a measurable current, which is sometimes, ambiguously, also called the impressed current in the literature [38]. The total current density in the volume conductor is the sum of the volume current density and the impressed current density:

$$\vec{J}(\vec{r}) = \vec{J}_v(\vec{r}) + \vec{J}_i(\vec{r}) \quad = \quad \sigma(\vec{r})\vec{E}(\vec{r}) + \vec{J}_i(\vec{r}) \quad = \quad -\sigma(\vec{r})\nabla\Phi(\vec{r}) + \vec{J}_i(\vec{r}) \tag{1.9}$$

The impressed currents can be further divided into a flow field J_i^F and a vortex field J_i^V [30]:

$$\vec{J}_i(\vec{r}) = \vec{J}_i^F(\vec{r}) + \vec{J}_i^V(\vec{r}) \tag{1.10}$$

These volume currents, or back-flowing currents, can be thought of distorting the electric and magnetic field.

A biologic volume conductor, such as the human body, may be seen as having an inhomogeneous conductivity, which moderates the idealized electric and magnetic fields. These theoretical discontinuities in the conductivity at the cellular and macroscopic level [76] produce secondary sources. The simplifying assumption is made that the volume conductor is piecewise homogeneous. Then, the secondary sources appear only at the boundaries between media of different conductivities. Therefore, the secondary sources depend on the electrical field and the geometry. The electrical scalar potential $\Phi(\vec{r}')$ can then be expressed as:

$$\Phi(\vec{r}') = -\frac{1}{4\pi\sigma} \left\{ \int_v \frac{\nabla \cdot \vec{J}_i(\vec{r})}{|\,\vec{r}' - \vec{r}\,|} \mathrm{d}v' + \sum_j \int_{S_j} (\sigma' - \sigma'')\, \Phi(\vec{r}) \vec{n} \cdot \nabla \left(\frac{1}{|\,\vec{r}' - \vec{r}\,|}\right) \mathrm{d}S_j \right\} \tag{1.11}$$

where σ is the local conductivity and $\vec{n}$ is the unit normal vector of the j^{th} boundary directed from the region with conductivity σ' to the region with conductivity σ''. The first term in the braces represents the

divergence of the primary impressed current, while the second term expresses the secondary currents due to boundaries of inhomogeneities. The magnetic flux density can be expressed similarly, using the law of Biot-Savart:

$$\vec{B}(\vec{r}') = -\frac{\mu_0}{4\pi}\left\{\int_v \frac{\nabla \times \vec{J}_i(\vec{r})}{\mid \vec{r}' - \vec{r} \mid} \mathrm{d}v' - \sum_j \int_{S_j} (\sigma' - \sigma'')\, \Phi(\vec{r})\vec{n} \times \nabla \left(\frac{1}{\mid \vec{r}' - \vec{r} \mid}\right) \mathrm{d}S_j\right\} \tag{1.12}$$

where the first term in the braces models the curls of the primary impressed current and the second term models possible secondary currents.

The electric potential is thus depended on the flow fields, whereas the magnetic field depends on the vortex field directly and on the flow fields through secondary currents [46] (Figure 14). On the cellular level, the ratio of secondary to primary source magnitudes was estimated to be $10^7 - 10^{11}$ [76]. The directly impressed current may thus be ignored. The sources of electric and magnetic fields are mostly of the secondary type. These sources themselves depend only on the boundary conditions on the electric field [76]. The question of whether electric and magnetic field contain identical information does not have proven answer. However, Plonsey's [76] formal argumentation advocates for both to be exchangeable.

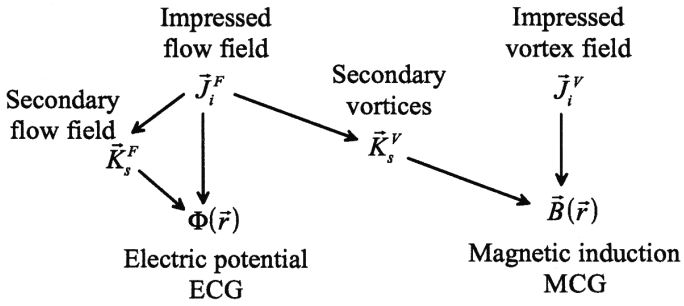

Figure 14: Dependency of electric potential $\Phi(\vec{r})$ and magnetic flux density $\vec{B}(\vec{r})$ on impressed current density $\vec{J}_i$ [46, 94]

1.4.4 Boundary Element Method

The above formulas of the electric potential and magnetic flux density may be used for an analytical solution of the forward problem. While this approach is fast, it is also numerically not very accurate [94]. With the boundary element method (BEM) inhomogeneities can be modelled much more accurately.

The principle idea is to assume piecewise homogeneous conductivity and to discretize the boundaries into elements of constant potential. Then we only require the potential at the boundaries, which can be computed through an integral equation. We additionally assume that the permeability μ_0 of the volume is constant. Since we can assume the fields to be quasi-static [77], the simplified quasi-static Maxwell equations explain the electromagnetic process:

$$\nabla \times \vec{E} = 0 \tag{1.13}$$

$$\nabla \times \vec{B} = \mu_0 \cdot (\vec{J}_i + \vec{J}_v) \tag{1.14}$$

$$\nabla \cdot \vec{D} = 0 \tag{1.15}$$

$$\nabla \cdot \vec{B} = 0 \tag{1.16}$$

As in the previous sections, $\vec{J}_i$ is the impressed current density and $\vec{J}_v$ is the volume current density. Applying $\vec{E} = -\nabla\Phi$ and $\vec{J} = \vec{J}_i + \vec{J}_v = \vec{J}_i + \sigma\vec{E}$ to Equation 1.15, we can write for constant conductivity:

$$\triangle\, \Phi = \frac{\nabla \cdot \vec{J}_i}{\sigma} \tag{1.17}$$

which is a Poisson equation with the following boundary conditions:

$$\Phi_1 = \Phi_2, \qquad \sigma_1 \frac{\partial \Phi_1}{\partial n} = \sigma_2 \frac{\partial \Phi_2}{\partial n} \tag{1.18}$$

At the outermost boundary of the body, the air is an isolator and thus its conductivity is zero. The second boundary condition then simplifies to:

$$\sigma_1 \frac{\partial \Phi_1}{\partial n} = 0 \tag{1.19}$$

The solution of Equation 1.17 in a piecewise homogeneous volume conductor with conductivity boundaries S_j is defined by:

$$\left(\sigma_k^- + \sigma_k^+\right) \Phi_k(\vec{r}) = 2\sigma_s \Phi_\infty(\vec{r}) - \frac{1}{2\pi} \sum_{j=1}^{N} (\sigma_j^- - \sigma_j^+) \int_{S_j} \Phi(\vec{r}') \cdot \frac{\vec{r} - \vec{r}'}{\mid \vec{r} - \vec{r}' \mid^3} \mathrm{d}\vec{S}_j \tag{1.20}$$

with σ_k^- and σ_k^+ denoting the conductivities of the inner and outer side of the surface S_k and σ_s denoting the conductivity of the volume that the sources are in [9]. Vector $\vec{r}'$ points to the location of the source while $\vec{r}$ points to the field point. $\vec{n}(\vec{r}')$ denotes the normal on surface S_j. The potential on any boundary is thus determined by adding the secondary potentials of every boundary in the model to the potential $\Phi_\infty(\vec{r})$ in an infinite homogeneous medium, which is itself given by:

$$\Phi_\infty(\vec{r}) = \frac{1}{4\pi\sigma} \int_V \vec{J}_i(\vec{r}) \cdot \frac{\vec{r} - \vec{r}'}{\mid \vec{r} - \vec{r}' \mid^3} \mathrm{d}v' \tag{1.21}$$

Following Geselowitz [30], the magnetic flux density is calculated through:

$$\vec{B}(\vec{r}) = \vec{B}_\infty(\vec{r}) + \frac{\mu_0}{4\pi} \sum_{j=1}^{N} (\sigma_j^- - \sigma_j^+) \int_{S_j} \Phi(\vec{r}') \vec{n}(\vec{r}) \times \frac{\vec{r} - \vec{r}'}{\mid \vec{r} - \vec{r}' \mid^3} \mathrm{d}\vec{S}_j \tag{1.22}$$

The magnetic flux density in an infinite homogeneous medium is determined by the continuous Biot-Savart law:

$$\vec{B}_\infty(\vec{r}) = \frac{\mu_0}{4\pi} \int_V \vec{J}_i(\vec{r}') \times \frac{\vec{r} - \vec{r}'}{\mid \vec{r} - \vec{r}' \mid^3} \mathrm{d}v' \tag{1.23}$$

To make the solution computationally feasible, the continuous boundaries are discretized into boundary elements, which are assumed to have a constant potential. Since the number of boundary elements is finite, the continuous model turns into a discrete one, which has a matrix representation:

$$\vec{\Phi} = \vec{\Phi}_\infty + \mathbf{Z} \cdot \vec{\Phi}, \qquad \vec{B} = \vec{B}_\infty + \mathbf{T} \cdot \vec{\Phi} \tag{1.24}$$

where $\mathbf{Z}$ and $\mathbf{T}$ are coefficient matrices that moderate the propagation of the potential generated by each boundary element to every other boundary element with respect to the concrete volume conductor model. We can compute $\vec{\Phi}$ as $(1 - \mathbf{Z})^{-1} \cdot \vec{\Phi}_\infty$ through matrix deflation, inversion based on factorization and dedeflation [52]. The solution can then be used to compute $\vec{B}$.

The question arises of which boundaries should be included in the boundary element model. Theoretically, the influence of the volume conductor is strongest at the boundaries between volume compartments with a large difference in conductivity. The most important such boundary is the body surface. Thus, modeling only the body surface and assuming homogeneous conductivity inside significantly improves the accuracy of the propagation model. Numerically, the assumption of homogenity allows us to use the boundary element method. The model can be further improved by defining the boundary of organs with a conductivity different from the conductivity of normal tissue [39, 40]. These are mainly the air-filled lungs with low conductivity and the heart with higher conductivity. The accuracy gain from the body surface is however most drastic [79, 94]. Also, the deeper the source is inside the body, the more critical is an accurate volume conductor model. In practice, the boundaries can be segmented from a magnetic resonance image (MRI) or computer tomogram (CT) of the individual patient.

The finite element method (FEM) additionally allows to model finegrained inhomogeneities and anisotropies of the volume conductor, which has a further positive effect on the reconstruction accuracy [100, 101]. Disadvantages are the computation time and the problem of determining or rather defining the local conductivities and anisotropies of the various body compartments and tissue types correctly.

1.4.5 Inverse Problem

The inverse problem is concerned with determining the parameters of the source model that best fit the measured data. For this task, electrical data derived using electrodes can supplement the magnetic readings. Typically, the solution to this problem is not unique, because the problem is either overdetermined for focal sources or underdetermined in case of distributed sources. Additionally, the solution is highly sensitive to small changes in the signal, such as noise [37].

The resulting source model is often referred to as the equivalent dipole because the physiological source does not actually consist of one or several dipoles. It is possible to either use electric current dipoles in the source model, or magnetic ones. The common choice are the electric dipoles, which have been introduced in the previous sections. Their advantage is a close relation to realistic electrophysiological conditions. Their disadvantage is their complexity. Magnetic dipoles are rarely used today.

In the case of focal sources, non-linear optimization algorithms are applied, which iteratively adjust the model parameters, position, orientation and magnitude of the dipoles, to match the model sensor readings and the real sensor recordings. This is computationally expensive, because a forward solution has to be made in every iteration to obtain the model sensor amplitudes. Various deterministic optimization techniques can be used for this task. The most common deterministic ones are the Levenberg-Marquardt method [63], the Nelder-Mead-Simplex method [78]. Among the non-deterministic ones, simulated annealing, genetic algorithms and evolutionary strategies [19] have been used.

For distributed sources, minimum norm techniques are applied. Common are the L1 and L2 norm [36]. Regularization is used to tackle the problem of underdetermination, such as Tikhonov-Philips regularization [95]. Common variants are the low resolution tomography (LORETA) [73].

1.4.6 Lead Field Theory

The lead field theory was first described for electric leads measuring the electric field [62], but equally applicable to magnetic recordings. Consider a single electric dipole in a linear ohmic volume conductor of inhomogeneous conductivity. A unipolar lead picks up the potential at the body surface. The relationship between the dipole magnitude and the lead potential is linear and may thus be expressed in a 3D transfer vector. A bipolar lead measures the potential difference between two unipolar leads. The combination of both tranfer vectors is called the lead vector of that bipolar lead. The lead vector thus describes the sensitivity of the electrode to a current source at a particular position in space. This vector is dependent on the electrode and dipole positions as well as the volume conductor. If we generalize the lead vector to all dipole positions, we have a continuous vector field, called the lead vector field, or shorter the lead field [62]. The potential difference V_j of the bipolar lead j can thus be expressed as [42]:

$$V_j = \int_V \vec{L}_j^E(\vec{r}) \cdot \vec{J}(\vec{r}) \mathrm{d}v \tag{1.25}$$

This electrode-sensor relationship can also be interpreted reciprocally. If we pass a unit current to the electrical lead, an electric field is produced inside the volume conductor. The current density then has exactly the same form as the lead field. This is why the sensitivity vector field is called "lead field" [62]. If we apply this lead field theory to magnetic sensors, such as magnetometer j, we have a very similar sensitivity relationship [42]:

$$B_j = \int_V \vec{L}_j(\vec{r}) \cdot \vec{J}(\vec{r}) \mathrm{d}v \tag{1.26}$$

For practical applications, we discretize the lead-vector field down to the configurations of the dipoles of our source model. The lead vector field for each sensor, turns into a vector of sensitivity coefficients, one for each dipole. These coefficient vectors are combined to a matrix (one sensor per row), called the leadfield matrix L [37], which describes the linear relation between source dipole current density amplitudes $\vec{d}$ and sensor amplitudes $\vec{s}$:

$$\vec{s} = L \cdot \vec{d}, \qquad \vec{s} \in \mathbb{R}^{\#\ sensors},\ \vec{d} \in \mathbb{R}^{\#\ dipole\ sources} \tag{1.27}$$

The leadfield matrix contains the information about the particular volume conductor, sensor setup and source model. Once the leadfield matrix is known, the forward solution reduces to a matrix-vector multiplication.

1.5 Research Objective

Motivation Magnetic field imaging (MFI) is a technique to record contact free the magnetic field distribution and estimate the underlying source distribution in the heart [57] and the brain [42]. Today, these biomagnetic fields are recorded with SQUIDs [2]. SQUIDs are restricted in their positioning to cryostats, since they require liquid helium (low temperature superconductors) or nitrogen (high temperature superconductors) cooling. Recent advances of optical magnetometers [10, 6, 102] which operate at room temperature make less restrictive sensor positioning feasible. Therefore, the general question arises of how to optimally place the sensors obeying a technical minimum distance due to the sensor size.

Objectives The critical points of this task are the optimization constraints and the efficient goal functions. The first constraint is that the sensor must be outside the body, while the optimal positions are inside the body. Secondly, the sensors have a certain size, which induces the minimum sensor distance constraint. The goal function needs to be evaluated frequently. Performing an inverse solution every time and assessing its accuracy is too time-consuming. Efficient goal functions, for example based on the leadfield matrix (Section 1.4.6) need to be investigated. The primary objectives of this study are:

1. To develop a generic optimization framework for sensor arrangements (position and orientation) that supports an arbitrary search volume and a minimum distance constraint,
2. To implement example components (one optimizer, one constraint handling technique and one goal function) which represent a full optimization strategy,
3. To apply this strategy to the settings of magnetocardiography.

The optimizer component is particle swarm [22, 1] based and the goal function component implements the condition number of the leadfield matrix. Lateron, these components can be further developed, customized and combined with new components that implement the generic interfaces. The secondary objective is to integrate them into the biomedical simulation environment SimBio [24].

Forward and inverse calculations are carried out with SimBio using a boundary element model of the human torso. A theoretical source model of the heart [17] and subsequently a more realistic source model derived from an MCG recording of a test person is used.

Overview This work is directed to two audiences, computer scientists and biomedical engineers, and is structured accordingly. From the computer scientists perspective, Chapters 2 and 3 resemble the problem analysis and algorithmic design including evaluation of alternatives, performance and asymptotic runtime. Chapter 4 describes the object-oriented design of the software and maps the algorithms to particular data structures and classes. For SimBio developers the integration into and changes to the SimBio framework are documented. Chapter 5 demonstrates the achieved goals and generic structure by application to magnetocardiography. Interesting results for biomedical engineering are documented in this chapter.

Original Contribution This study was conducted as Diploma Thesis (German "Diplomarbeit"). This document has been written by Stephan Lau. Any content that does not have a literature reference attached to it, is an original contribution. In particular, Stephan Lau contributed:

1. the constraint optimization problem analysis (Chapter 3),
2. the algorithmic design (all algorithms except Algorithm 1),
3. the time and memory complexity analysis of the algorithms (Chapter 3),
4. the object-oriented design and integration into SimBio (Chapter 4),
5. the implementation in C++ and testing (Chapter 4),
6. the construction of the BEM, the source model and the search volumes (Chapter 5),
7. the acquisition and interpretation of optimization results (Chapter 5).

Selected results have been presented at the 16th International Conference on the Computation of Electromagnetic Fields (COMPUMAG) in June 2007 [56] and will be presented at the 41st Annual Conference of the German Society for Biomedical Engineering (DGBMT) in September 2007 [55].

2 Goal Functions and Optimization Strategies

2.1 Reconstruction Accuracy and Robustness

A goal function for the optimization of sensor arrays needs to quantify the reconstruction accuracy and robustness against noise. The reconstruction accuracy may be expressed experimentally in the distance of the position of a physical dipole to the position of an equivalent dipole reconstructed from measuring the physical dipole [59]. The robustness can be estimated by adding noise to the recording and repeating the reconstruction. However, in biomedical practice we do not know the nature of the sources. In fact, measurable dipole moments are only produced by the superposition of microscopic intracellular dipole moments. So the assumption of the existence of few individual dipoles is a simplification already. We are therefore interested in the stability of the reconstructed source model in presence of noise which can be estimated by repeated simulation [17].

Theoretically, each goal function evaluation of a sensor setup could involve in a set of representative inverse solutions with varying added noise. However, with the current computational resources an inverse solution requires a significant amount of time. Each inverse solution itself is an optimization process, which is implemented as the solution of linear equation system (distributed sources) or the iterative optimization of the goodness of fit of a candidate dipole set to the field measurements (focal sources). For focal sources we need to perform a forward solution to obtain the goodness of fit every time. It is therefore necessary to develop simple approximate measures of the reconstruction accuracy and robustness.

2.2 Lead Field Matrix based Goal Functions

A suitable representation of the coupling between sensors and dipoles, mediated by the volume conductor, is the leadfield matrix, which was introduced in Section 1.4.6. The leadfield matrix represents the linear transformation of a vector of dipole amplitudes to the according vector of sensor amplitudes. While this matrix can be used for fast forward calculations, its structure is more interesting because it represents the sensitivity of each sensor to each dipole.

Condition Number The condition number of the matrix A is a measure of the conditioning of the linear equation system represented by it:

$$\mathbf{A}\mathbf{x} \;=\; \mathbf{b} \tag{2.1}$$

A well-conditioned equation system reacts to a small change in the right hand side **b** with a small change in the result vector **x**. An ill-conditioned equation system instead reacts with a large change in the result vector. If we solve an ill-conditioned system, we can only trust the solution to a certain degree (number of digits) [45]. Consider a disturbance $\mathrm{d}\mathbf{b}$ of **b**. Then the change $\mathrm{d}\mathbf{x}$ in the solution vector can be denoted by

$$(\mathrm{d}\mathbf{x}) \;=\; \mathbf{A}^{-1}(\mathrm{d}\mathbf{b}) \tag{2.2}$$

Applying a matrix norm, which by definition fulfils the inequality $\| \mathbf{A} \cdot \mathbf{B} \| \leq \| \mathbf{A} \| \, \| \mathbf{B} \|$, to both equations, yields

$$\| \mathbf{A} \| \, \| \mathbf{x} \| \geq \| \mathbf{b} \| \tag{2.3}$$

and

$$\| \mathrm{d}\mathbf{x} \| \leq \| \mathbf{A}^{-1} \| \, \| \mathrm{d}\mathbf{b} \| \tag{2.4}$$

which can be combined to define an upper bound of the relative change of the solution vector

$$\frac{\| \mathrm{d}\mathbf{x} \|}{\| \mathbf{x} \|} \;\leq \| \mathbf{A} \| \, \| \mathbf{A}^{-1} \| \; \frac{\| \mathrm{d}\mathbf{b} \|}{\| \mathbf{b} \|} \tag{2.5}$$

The relative change in the right hand side is magnified by at most $\| \mathbf{A} \| \, \| \mathbf{A}^{-1} \|$, which is defined to be the condition number κ

$$\kappa \;= \| \mathbf{A} \| \, \| \mathbf{A}^{-1} \| \tag{2.6}$$

If the L_2 norm is chosen and the matrix is symmetric, the condition number is the ratio of maximum to minimum eigenvalue:

$$\kappa_2 = \| \mathbf{A} \|_2 \, \| \mathbf{A}^{-1} \|_2 = \; \frac{\lambda_{max}}{\lambda_{min}} \tag{2.7}$$

If the matrix is not symmetric, κ can be expressed as ratio between maximum and minimum singular value:

$$\kappa_2 = \frac{\sigma_{max}}{\sigma_{min}} \tag{2.8}$$

The condition number of the identity matrix is one. The computation of the condition number requires more operations than solving the equations by elimination. Efficient approximations of the condition number in cases where certain factorizations of the matrix exist are reviewed in [32]

For our purposes the condition number is suitable as a measure of inverse solution robustness. A low condition number indicates that a small change in a dipole amplitude results in a small change in the sensor readings. The inverse problem condition is more robust, the lower the condition number is. An intuitive interpretation of the condition number is the slope of the sorted singular values. The condition number has been used for optimal sensor sensor selection before [67].

Further Figures of Merit The goal functions applicable to the optimization of sensor setups are commonly referred to as figures of merit, because initially these measures were used to assess and compare the quality of existing or planned sensor arrangement designs. Due to the helium cooling, the SQUIDs needed to be fixed in their arrangement and the cryostat needed to be designed to suit the SQUID array or vice versa. Recently proposed leadfield matrix based figures of merit are lower error bounds [80] and measures based on the projecting of solution space basis vectors of one matrix onto the solution space of another one [81, 28].

2.3 Further Goal Functions

Goal functions that have been used already for body surface potential maps (BSPM) in the 1970's are the root mean square error between a selection of electrical leads and the full BSPM, the mean correlation coefficient and the error to signal power ratio [61, 25]

Several approaches based on Shannon's theory of communication [90] exist. The channel capacity [89] has been theoretically investigated [36]. A second measure, called total information [68, 69, 93], is based on transforming measured data into virtual channels and summing up the power signal to noise ratios without making assumptions of the source currents.

2.4 Particle Swarm Optimization

2.4.1 Advantage of Swarm Intelligence

The task of optimizing sensor arrangements is similar to optimizing dipole arrangements (inverse solution) in that they both modify the linear mapping between dipole and sensor amplitudes. A difference is that the dipole amplitudes are optimized, while the sensors are only optimized in position and orientation. Secondly, a source model commonly consists of few dipoles, while a sensor setup typically has a larger number of sensors to sample the field.

The typical optimization techniques for dipole fitting are the Levenberg-Marquardt method [63, 58] and the Nelder-Mead-Simplex [78] method. Both techniques are strongly gradient oriented and thus prone to local minima in the goal function and dependent on a sensible start solution. While for few dipoles this may still work, it is a critical limitation for a larger set of sensors. Additionally, the sensors have a certain size, so a minimum sensor distance needs to be considered as constraint.

Therefore, in this study particle swarm optimization (PSO) [48, 22] will be adapted and applied to sensor optimization. The advantage of PSO is that not one current solution is iteratively adjusted towards the optimum, but a swarm of solutions, the particles. Each particle is guided not only by the gradient information of itself but of the other particles at other locations as well. This technique is thus much more robust against local minima or disturbances in the goal function.

2.4.2 Standard PSO Algorithm

One of the first authentic PSO algorithms was presented by Kennedy and Eberhart in 1995 [47]. Since then many modifications to the method itself (Section 2.4.4) and customizations to particular problems have been developed. To be able to compare variants and customizations to a non-modified version, a standard PSO algorithm has been defined in 2006 and validated by main researchers in the field. The standard PSO

algorithm is similar to the original one of Kennedy and Eberhart [47], but includes improvements based on recent developments. This standard is freely available [1] and is used in this study as the base algorithm to which modifications are made.

The standard PSO algorithm is formally described in Algorithm 1. First, the positions are initialized randomly within the limits of each dimension. The velocities are initialized by picking a random position and using the half the vector from the particle position to that position. In the first iteration, the information links are established randomly in a way that informants may be chosen more than once. So K is only the maximum possible number of informants. Every time an iteration does not yield a improvement in the goal function value, the information links are re-initialized.

Algorithm 1: Particle swarm optimization (Standard)

Data : SxD matrix $\mathbf{P}$ of positions of S particles in D-dimensional space;
SxD matrix $\mathbf{B}$ of best position so far for all particles; global best particle b;
SxD matrix $\mathbf{V}$ of velocities; number of goal function evaluations so far $nEval$
SxS matrix $\mathbf{L}$ of boolean information link existences

Input : # particles S; maximum # informants per particle K; cognitive coefficients w, c;
Dimension limit vectors $\mathbf{x}_{min} \in \mathbb{R}^D$ and $\mathbf{x}_{max} \in \mathbb{R}^D$; goal function $g()$;
Maximum number of goal function evaluations E; desired goal function value ϵ

Output : Result vector $\mathbf{r} \in \mathbb{R}^D$

1 **foreach** *particle* p *and dimension* d **do** *// initialization*
2 $\quad \mathbf{P}(p,d) \leftarrow rand(\mathbf{x}_{min}(d), \mathbf{x}_{max}(d))$; *// uniformly random*
3 $\quad \mathbf{B}(p,d) \leftarrow \mathbf{P}(p,d)$;
4 $\quad \mathbf{V}(p,d) \leftarrow (rand(\mathbf{x}_{min}(d), \mathbf{x}_{max}(d)) - \mathbf{P}(p,d))/2$; *// non-uniformly random*
5 $b \leftarrow$ particle with minimum goal function value $g(\mathbf{P}(b,:))$;
6 $e \leftarrow g(\mathbf{P}(b,:))$;
7 **while** $e \geq \epsilon$ *and # goal function evaluations* $\leq E$ **do**
8 $\quad$ **if** *current error* e $\geq$ *previous error* **then** *// re-initialize information link topology*
9 $\quad\quad \mathbf{L} \leftarrow 0$;
10 $\quad\quad$ **foreach** *particle* p **do**
11 $\quad\quad\quad \mathbf{L}(p,p) \leftarrow true$; *// each particle informs itself*
12 $\quad\quad\quad$ **for** $k \in [1, K]$ **do**
13 $\quad\quad\quad\quad p' \leftarrow rand(1, S)$;
14 $\quad\quad\quad\quad \mathbf{L}(p,p') \leftarrow true$; *// links are one way only*
15 $\quad$ **foreach** *particle* p **do**
16 $\quad\quad i \leftarrow$ informant particle with minimal goal function value $g(\mathbf{B}(i,:))$;
17 $\quad\quad$ **foreach** *dimension* d **do** *// update velocities and move particles*
18 $\quad\quad\quad \mathbf{V}(p,d) \leftarrow w \cdot \mathbf{V}(p,d) + rand(0,c) \cdot (\mathbf{B}(p,d) - \mathbf{P}(p,d)) + rand(0,c) \cdot (\mathbf{B}(i,d) - \mathbf{P}(p,d))$;
19 $\quad\quad\quad \mathbf{P}(p,d) \leftarrow \mathbf{P}(p,d) + \mathbf{V}(p,d)$;
20 $\quad\quad$ **foreach** *dimension* d **do** *// restore lower and upper limits*
21 $\quad\quad\quad$ **if** $\boldsymbol{P}(p,d) < \boldsymbol{x}_{min}$ **then** $\mathbf{P}(p,d) \leftarrow \mathbf{x}_{min}$; $\mathbf{V}(p,d) \leftarrow 0$;
22 $\quad\quad\quad$ **if** $\boldsymbol{P}(p,d) > \boldsymbol{x}_{max}$ **then** $\mathbf{P}(p,d) \leftarrow \mathbf{x}_{max}$; $\mathbf{V}(p,d) \leftarrow 0$;
23 $\quad\quad$ **if** $g(\boldsymbol{P}(p,:)) < g(\boldsymbol{B}(p,:))$ **then** $B(p,:) \leftarrow \mathbf{P}(p,:)$; *// update individual best*
24 $\quad\quad$ **if** $g(\boldsymbol{P}(p,:)) < g(\boldsymbol{B}(b,:))$ **then** $b \leftarrow p$; *// update global best*
25 $\quad\quad e \leftarrow g(\mathbf{B}(b,:))$; *// update error*
26 $\mathbf{r} \leftarrow \mathbf{B}(b,:)$;

In each iteration all particles move along a certain velocity vector. This vector is composed of the previous velocity weighted with w, the vector from the current position to the individual best one so far, and the vector from the current position to the best position within the informants (Line 18). The last two

components are both weighted with a random number in the intervall $[0, c]$. So the movement is non-deterministic. Note that the new individual best and the new position are available to the particles moving subsequently already.

If a particle passes a dimension limit, its position value in that dimension is put back to the limit and the velocity is set to zero. This assumes that the search volume limit marks the edge of sensible solutions. Thus the swarm should not be directed there.

The default settings for this PSO implementation are summarized in Table 1. The swarm size must be chosen to suit the number of dimensions. Here, the number of informants is very small. Recent research [49, 64] also advocates fully-informed swarms. The parameters w and c adjust the degree of adaption to the current best individual and informant solution as well as the movement energy of the particles. The number of evaluations is chosen to be sufficient for 1000 iterations. A number of repetitions are suggested to eliminate initialization errors.

Symbol	Meaning	Default
S	swarm size	$10 + 2 \cdot \sqrt{D}$
K	number of informants	3
w	first cognitive coefficient	$1/(2 \cdot ln(2))$
c	second cognitive coefficient	$0.5 + ln(2)$
E	maximum number of evaluations	$S \cdot 1000$
R	number of repetitions	50

Table 1: Default parameter values for standard PSO [1]

2.4.3 Modifications to the Standard Algorithm

The task of sensor optimization involves two constraints. First we have a limited search volume in which the sensors are allowed to move. In particular, they are not allowed to move inside the body surface, even though this is where the optimal positions can be expected. Secondly, the sensors have a certain size, which induces a minimum distance of the sensor centers. Depending on the particular application other constraints may apply. Additionally, there is not just one strategy to handle any of these constraints. It is therefore desirable to add a support for arbitrary constraints into the algorithm.

Since we do not know the particular nature of the constraint, we separate the definition of the constraint from the optimization algorithm and assume an interface that provides us the necessary operations. These operations are to obtain a set of parameters that is random within the search space induced by the constraint definition, to check the constraint satisfaction of given parameter set and to fix a parameter set so that if fulfills the constraint. The particular constraint definitions and operations used in this study are described in detail in Section 3.

Initialization of Positions The most common approach is to initialize the positions randomly within the constrained search space. It is also possible of course to make use of any a priori information. A generic way to obtain a nearly random position within an arbitrary search volume is to sample the volume with a fine enough sampling width. This results in a regular grid of positions within the search volume. Each grid point can be thought of as center of a cube. The set cubes approximates the search volume. One can then pick a random grid point and within the cube of this grid point a random position. The particular implementation is deferred to the constraint which is representing the search volume (Line 2 in Algorithm 2). So if the search volume is a geometric object, such as a sphere, a random position may be obtained much simpler, e.g. through a random sperical coordinate.

Initialization of Velocities The velocities however pertain to PSO and not to optimizers in general. Their initialization should therefore be implemented in the PSO algorithm using the constraint interface. The velocities should be in relation to the size of the search volume. To determine the approximate dimensions of the volume, we sample it through 50 random parameter sets from the constraint interface. We obtain an interval for each dimension and pick a random number between + and - half of the interval length (Line 10). In the standard implementation, half of the interval length is used. However, since we underestimate the limits of the search space, we do not need this.

Information Link Topology The information topology was changed so that each particle has exactly K informants, not counting itself (Line 19). This enables us to use a fully informed swarm [49] or to control the degree of interconnection.

Algorithm 2: Modified particle swarm optimization

Data : SxD matrix **P** of positions of S particles in D-dimensional space;
SxD matrix **B** of best position so far for all particles; global best particle b;
SxD matrix **V** of velocities; number of goal function evaluations so far $nEval$;
SxS matrix **L** of boolean information link existences

Input : # particles S; maximum # informants per particle K; cognitive coefficients w, c;
Constraint interface I; goal function $g()$; velocity adjustment factor $\lambda \in [0, 1]$;
Maximum number of goal function evaluations E; desired goal function value ϵ

Output : Result vector $\mathbf{r} \in \mathbb{R}^D$

→ 1 **foreach** *particle* p **do** *// initialization of positions*
→ 2 $\quad$ $\mathbf{P}(p, :) \leftarrow$ random solution obtained from the constraint interface I;

→ 3 $\mathbf{x}_{min} \leftarrow \infty$; $\mathbf{x}_{max} \leftarrow -\infty$;
→ 4 **for** $i = 1, 2, .., 50$ **do** *// get dimensions of search space by sampling*
→ 5 $\quad$ $\mathbf{p} \leftarrow$ random solution obtained from the constraint interface I;
→ 6 $\quad$ **foreach** *dimension* d **do**
→ 7 $\quad\quad$ **if** $\boldsymbol{p}(d) < \boldsymbol{x}_{min}(d)$ **then** $\mathbf{x}_{min}(d) \leftarrow \mathbf{p}(d)$;
→ 8 $\quad\quad$ **if** $\boldsymbol{p}(d) > \boldsymbol{x}_{max}(d)$ **then** $\mathbf{x}_{max}(d) \leftarrow \mathbf{p}(d)$;

→ 9 **foreach** *particle* p *and dimension* d **do** *// initialization of velocities*
→ 10 $\quad$ $\mathbf{V}(p, d) \leftarrow (\mathbf{x}_{max}(d) - \mathbf{x}_{min}(d)) \cdot rand(-0.5, 0.5)$;

11 $b \leftarrow$ particle with minimum goal function value $g(\mathbf{P}(b, :))$;
12 $e \leftarrow g(\mathbf{P}(b, :))$;

13 **while** $e \geq \epsilon$ *and # goal function evaluations* $\leq E$ **do**
14 $\quad$ **if** *current error* e $\geq$ *previous error* **then** *// re-initialize information link topology*
15 $\quad\quad$ $\mathbf{L} \leftarrow 0$;
16 $\quad\quad$ **foreach** *particle* p **do**
17 $\quad\quad\quad$ $\mathbf{L}(p, p) \leftarrow true$; $\quad$ *// each particle informs itself*
18 $\quad\quad\quad$ **for** $k \in [1, K]$ **do**
→ 19 $\quad\quad\quad\quad$ $p' \leftarrow$ random particle that is not an informant yet;
20 $\quad\quad\quad\quad$ $\mathbf{L}(p, p') \leftarrow true$; $\quad$ *// links are one way only*
21 $\quad$ **foreach** *particle* p **do**
22 $\quad\quad$ $i \leftarrow$ informant particle with minimal goal function value $g(\mathbf{B}(i, :))$;
23 $\quad\quad$ **foreach** *dimension* d **do** *// update velocities and move particles*
24 $\quad\quad\quad$ $\mathbf{V}(p, d) \leftarrow w \cdot \mathbf{V}(p, d) + rand(0, c) \cdot (\mathbf{B}(p, d) - \mathbf{P}(p, d)) + rand(0, c) \cdot (\mathbf{B}(i, d) - \mathbf{P}(p, d))$;
25 $\quad\quad\quad$ $\mathbf{P}(p, d) \leftarrow \mathbf{P}(p, d) + \mathbf{V}(p, d)$;
→ 26 $\quad\quad$ **foreach** *particle* p **do** *// restore constraint and adjust velocities*
→ 27 $\quad\quad\quad$ $\mathbf{p}' \leftarrow$ fix constraint in $\mathbf{P}(p, :)$ using interface I;
→ 28 $\quad\quad\quad$ $\mathbf{V}(p, :) \leftarrow \mathbf{V}(p, :) - \lambda \cdot (\mathbf{p}' - \mathbf{P}(p, :))$;
→ 29 $\quad\quad\quad$ $\mathbf{P}(p, :) \leftarrow \mathbf{p}'$;
30 $\quad\quad$ **if** $g(\boldsymbol{P}(p, :)) < g(\boldsymbol{B}(p, :))$ **then** $\mathbf{B}(p, :) \leftarrow \mathbf{P}(p, :)$; *// update individual best*
31 $\quad\quad$ **if** $g(\boldsymbol{P}(p, :)) < g(\boldsymbol{B}(b, :))$ **then** $b \leftarrow p$; *// update global best*
32 $\quad\quad$ $e \leftarrow g(\mathbf{B}(b, :))$; $\quad$ *// update error*
33 $\mathbf{r} \leftarrow \mathbf{B}(b, :))$;

Velocity Adjustment due to Constraint From the perspective of the optimizer the restoration of the constraints in each iteration is a deviation d of the current position in high-dimensional space. The current velocity vector describing the individual direction in which the global optimum is expected should then be adjusted as shown in Figure 15. The new velocity should be the sum of the old velocity and the negative scaled deviation vector (Line 28). The $\lambda \in [0, 1]$ may be chosen heuristically. $\lambda = 0$ or $\lambda = 1$ are also valid alternatives. $\lambda = 1$ assumes that the length of the velocity vector is approximately the distance to the expected global optimum. $\lambda = 0$ assumes that the deviations caused by the constraint are small enough to be ignored, where small enough means that they are compensated by the randomization of the weight c.

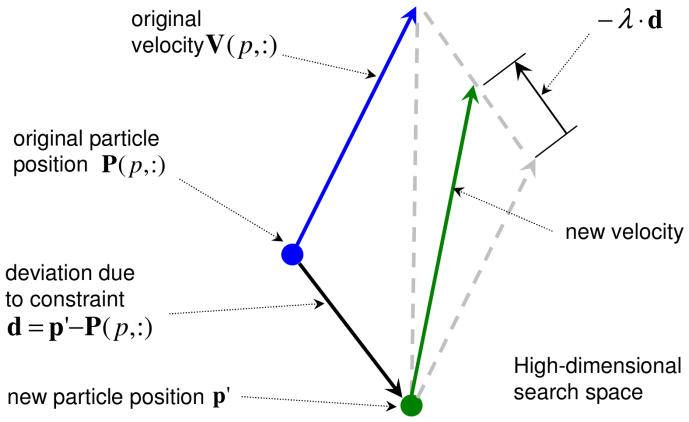

Figure 15: Adjustment of the velocity of the particles after the shift caused by the constraint restoration. The blue dot and arrow indicates the old particle position and velocity in high-dimensional search space. The new velocity (green) is obtained by subtracting the scaled deviation vector. Note that particles encode multiple sensor positions and directions.

In the standard PSO algorithm each dimension of the velocity vector that lies outside the search volume is set to zero. This assumes that the search volume marks the limit of sensible solutions. This is not true in our case. A sensor inside the body, or even the heart muscle, would significantly improve the goal function value. The search volume is a technical constraint, which modifies the optimization problem. Therefore, in our algorithm the velocity pointing outside is kept. This allows other particles to be informed about the direction of the expected optimum.

2.4.4 Other Variants of PSO Optimization

There is a whole range of PSO customizations available for particular problems [22, 13, 97]. Key points of design are the information topologies [49, 64], the memory of the particles [11], the types of information exchanged, the dynamic adaptation of the cognitive coefficients and multiple objectives [14]. Discrete PSO algorithms have been proposed by Clerc et al. [12] using the traveling salesman problem as example. This discretization is however different from our quasi-continuous approach, in that our optimizer does not do a combinatorical search. PSO applications in electromagnetics exist for other problems than sensor optimization [83].

3 Constraint Handling and Search Volumes

3.1 Constraint Types

In the optimization of sensor arrangements two constraints need to be considered. First, the sensors have to be placed outside the object under investigation, e.g. the human body. This is non-trivial, because theoretically and practically the ideal sensor positions are inside and close to the source volume, e.g. the heart [65] or the somatosensory cortex of the brain [20]. A physical search volume needs to be defined, which constrains each triplet $\{x, y, z\}$ of positions of one sensor, respectively. The overall search space is limited by a composite boundary.

The second constraint is the minimum sensor distance m caused by the size of the sensors themselves. This causes the high-dimensional sensor position search space to be occluded with patches of invalid sensor arrangements. First, consider two sensors s_1 and s_2 in one-dimensional space. If $s_1 < s_2$ then s_1 can never jump across s_2 without coming too close to s_2. In 1D space, the minimum distance constraint induces compartments of valid sensor configurations. Now consider the 2D case with 2 sensors: If $x(s_1) < x(s_2)$, then it is always possible to move both sensors continuously obeying minimum distance with the help of the second dimension, so that $x(s_1) > x(s_2)$. With two or more physical dimensions the valid search space induced by the minimum distance constraint consists of only one compartment.

In general, for each physical dimension $d \in [1, D]$ with $D \geq 2$ and each pair of sensors out of n an invalid subspace is induced which extends infinitely in $D \cdot n - 1$ dimensions, since $d(s_2) \in [d(s_1) - m, d(s_1) + m]$. The number of such subspaces is:

$$D \cdot \binom{n}{2} = D \cdot \frac{n!}{(n-2)! \cdot 2!} = \frac{D}{2} \cdot n \cdot (n-1) \in \Theta\left(D \cdot n^2\right) \tag{3.1}$$

For our purposes, $D = 3$ and we have $\Theta\left(n^2\right)$ subspaces. These subspaces overlap each time three or more sensors clash, merging all of them into one subspace. Thus, the invalid search space also consists of only one compartment. Valid and invalid search space are interwoven.

When both constraints are combined, the number of sensors must be chosen less than the maximum number of sensors the search volume can fit. For realistic optimizations, which operate only inside the valid search volume, the number of sensors should be significantly lower than the maximum number of sensors.

3.2 Penalty versus Fixing Strategy

There are two principle ways of handling such constraints. The first and simplest is to adjust the goal function with a penalty. To keep sensors outside the measured object for example, an exponential increase in the goal function towards the boundary of the search space could be added. If the sensor arrangement as a whole is penalized, the information of which sensor(s) violates the constraint is not used. A good setup with a minor violation may be lost. For the search volume constraint is would also be possible to have a separate penalty for each sensor.

With a minimum distance, the case is more complex. A minimum distance violation inherently involves more than one sensor. Infact, one sensor at one position can be valid or invalid depending on the current positions of the other sensors. If we would penalize each sensor individually, we would disregard the interaction with other sensors. For example, if two sensors move towards one optimum, they would both be penalized and would thus both have to move away and the optimum would never be identified.

The penalty approach disregards information about how the constraint is violated and how the arrangement can be made to satisfy the constraint easily. An advantage on the other hand is that boundaries are soft, so that an optimizer may tunnel through a thin barrier in the goal function. The penalty strategy is suitable for problems with few parameters, which have global constraints.

Fix Strategy The second approach of handling the constraints is to fix the minor violations caused by one iteration step directly after that step. The initial solution satisfies the constraint by definition. In each iteration the sensors are modified slightly in position and direction, which causes slight localized constraint violations. Knowing the details about individual violations enables us to fix them with minimal impact on the overall solution. For example, if one sensor is moved across the boundary of the physical search space, only this sensor is moved back onto the boundary. All other sensors remain unchanged and the constraint is satisfied. Additionally, the minor violations can not multiply over successive iterations. Because the goal

function is computed after fixing the current parameter, there is no deviation in the goal function. However, optimizers with memory of previous steps may need to consider the local changes due to fixing. The fixing strategy is suitable for problems with localized constraints and supports large numbers of parameters.

For the task of optimizing sensor arrangements the fixing strategy is the better choice, because the number of parameters is very high and the constraints are localized. For example, an arrangement of 50 sensors has 250 parameter, three for the x, y, and z coordinates and two for ϕ and θ, the spherical angles of the direction, respectively. Therefore, the fixing strategies outlined in this chapter will be used in this study.

3.3 Continuous Case

3.3.1 General Considerations

The continuous case applies to the Levenberg-Marquardt and Nelder-Mead-Simplex technique and to particle swarm optimization. The physical space within the search volume is continuous as well as the two-dimensional space of sensor directions.

3.3.2 Continuous Search Volume Realization

A realistic application, such as magnetocardiography, requires a custom search volume, such as a volume around a human torso. We wish to be able to place the sensors as close as possible to the body. Such a search volume is defined by a triangulation of its surface, which is also a usual representation in a boundary element model of the human body for forward and inverse solutions. The simplest way to produce this search volume is therefore to first apply a dilation operator and to invert boundary element model. Second, this inverted model can be dilated and the difference between to the first step can be used as search volume, again enclosed in triangulated boundary. The typical operations needed in our optimization are the check of whether a given position is inside the volume or not (Algorithm 3) and to obtain the projection on the boundary to any given point (Algorithm 4).

Inside Search Volume Check In order to define inside and outside, we need to know for each triangle which side is facing inside. We use the following rule: A triangle t with the vertex triplet $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ is oriented inside, if its normal vector using the right hand rule $((\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1))$ points inside. This information can be saved in a boolean array during initialization. To determine the orientations efficiently and automatically, the orientation of one triangle needs to be determined and spread recursively to all its neighbor triangles. Under the assumption that the boundary is closed, the first triangle can be found by selecting a random point outside the bounding box and projecting it on the surface. The side of the projection triangle on which the point is is outside. The initial recursive spread requires $\Theta(\#triangles)$ sequential time and memory in the general case. The asymptotic runtime of Algorithm 3 is identical to that of Algorithm 4.

Algorithm 3: Test whether a position is inside a triangulated boundary

Data : map: triangle $t \rightarrow$ vertex triplet $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$;
orientation of each triangle ($((\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1))$ points inside?)
Input : position $\vec{s} \in \mathbb{R}^3$
Output : true/false

```
1 find boundary triangle t ∈ ℕ on which the projection of s⃗ lies (Algorithm 4);
2 determine distance d of s⃗ to the plane defined by t using the right hand rule;
3 if t oriented inside then
4 |   if d ≥ 0 then return true ;    // boundary counts as inside
5 |   else        return false;
6 else
7 |   if d ≤ 0 then return true ;    // boundary counts as inside
8 |   else        return false;
```

Projection on Boundary The projection on the triangulated search volume boundary (Algorithm 4) is the central method of the search volume implementation. It works in two steps. First, the closest vertex of the triangulation to any given position is determined using the runtime-optimized strategy of Algorithm 5. This reduces the set of candidate triangles to the ones adjacent to the closest vertex. Second, the triangle with the largest solid angle as seen from the given position is selected. Under the assumption that the triangles are of same size, this yields the closest triangle. Then it only remains to determine the closest point inside this triangle. The projection of the given position into the plane of the triangle can fall into one out of the seven cases in Figure 16. In case I we keep the projection. In cases II, IV and VI we take the adjacent vertex, and in cases III, V and II we project onto the edge of the triangle.

Algorithm 4: Projection on triangulated search volume boundary

Data : continuous search volume $V \subseteq \mathbb{R}^3$
Input : position $\vec{s} \in \mathbb{R}^3$
Output : projection point $\vec{s}' \in V$ on boundary; triangle number $t' \in \mathbb{N}$

```
find closest boundary grid node b⃗ to s⃗ (Algorithm 5);
maxSolidAngle ← 0 ;   // maxSolidAngle = max. solid angle so far
foreach triangle t with b⃗ as one of its vertices do
    compute solid angle a of t seen from s⃗;
    if | a |> maxSolidAngle then
        t' ← t;
        maxSolidAngle ← a;
determine projection p⃗ of s⃗ on the plane defined by triangle t';
determine closest position s⃗' to p⃗ inside triangle t' (Figure 16);
```

By searching for the closest vertex instead of the closest triangle the performance is increased significantly, but we accept a minor error in a certain rare case. If the search volume is concave and the given position is very close to the middle, a vertex of one side may be closer than any of the vertices of the correct triangle on the other side but further away than the correct triangle itself. Then the given position is projected onto the first side, not the second one. This is very minor, because it can only happen if the position is very close to the middle, in which case it does not matter on which side it is projected.

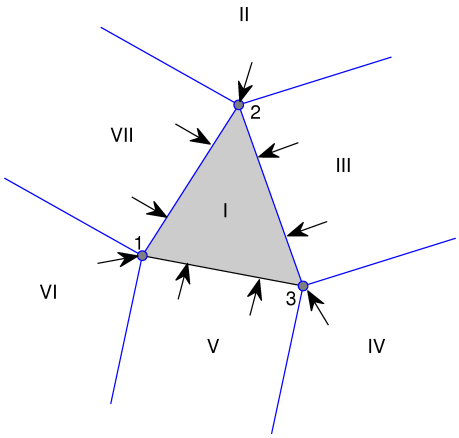

Figure 16: Cases in finding the closest position within a triangle

Efficiency Considerations The problem with a non-analytic search volume is that non-optimized operation on it require us to touch every point or triangle. We therefore need to define an efficient grid data structure and efficient operations on it. The two generic operations we need are to find the closest grid node to any given position and to find all grid nodes within distance range. In order to narrow down the number of grid nodes to be checked, we would ideally like to sort the grid nodes according to some one-dimensional

distance measure. A good approximation of the distance is for example the x-coordinate. If we are looking for grid nodes in the ϵ-environment of a given position $\vec{s}$, it is sufficient to check all grid nodes $\vec{g}$ with $x(\vec{g}) \in [x(\vec{s}) - \epsilon, x(\vec{s}) + \epsilon]$ because for the remaining grid nodes $\vec{g}' \mid \vec{g}' - \vec{s} \mid \geq \mid x(\vec{g}') - x(\vec{s}) \mid > \epsilon$.

The only pitfall is that volume grids, which we use for the discrete search volume later, are regular and thus a large number of grid nodes would have identical x-, y- or z-coordinates, respectively. We would then have to check whole set of nodes with one identical coordinate within the distance range. These node sets would be slices of the search volume, in our case irregularly shaped bands arround the torso. A simple solution is to rotate the coordinate system around the z-axis by a small angle and use the new coordinate x_{rot} as sort value. Under the assumption of a regular grid, the ideal angle is:

$$\alpha = tan^{-1}\left(\frac{1}{\#\ samples\ in\ y\ dimension}\right) \tag{3.2}$$

In the worst case the given position is exactly in the middle between two slices. Without rotation we would have to check both slices. If we rotate by α, half of each of the slices is moved out of the search range. Since we only need the rotated coordinate x_{rot} for sorting, we define our sort dimension as:

$$v([x, y, z]) = x_{rot} = cos(\alpha) \cdot x - sin(\alpha) \cdot y \tag{3.3}$$

If the grid is not regular, we could also use the first principal axis of a PCA of the grid nodes as heuristic sort dimension.

Closest Grid Node This sorting dimension is used to find the closest grid node to a given position in Algorithm 5. The idea is to start at the first grid node with at least the sort value of the position and to search in both directions. We keep the closest node so far and its distance. Once we reached the current minimum distance in both directions we're done. Interestingly, the search range gets smaller each time we find a closer node and may even fall below one of the ends of the already searched range (Lines 17- 18).

Algorithm 5: Fast retrieval of closest grid node

Data : list L of {value v_j, node index i_j} pairs sorted according to v_j
Input : position $\vec{s} \in \mathbb{R}^3$
Output : index $i' \in \mathbb{N}$ of grid node closest to $\vec{s}$

```
1  determine sort value v of s⃗;
2  determine index i of the first element in L with L[i].v ≥ v (Algorithm 6) ;   // start index
3  lower ← min(max(i, 0), |L| − 1) ;   // select first lower candidate
4  upper ← max(min(i + 1, |L| − 1), 0) ;   // select first upper candidate
5  d ← ∞ ;   // minimum distance so far
6  while true do
7      if lower > −1 then
8          if |s⃗ − G[L[lower].i]| < d then
9              d ← |s⃗ − G[L[lower].i]| ;
10             i′ ← L[lower].i;
11         lower ← lower − 1 ;   // move one further
12     if upper < |L| then
13         if |s⃗ − G[L[upper].i]| < d then
14             d ← |s⃗ − G[L[upper].i]| ;
15             i′ ← L[upper].i;
16         upper ← upper + 1 ;   // move one further
17     if (lower = −1 or L[lower].v < (v − d)) and
18       (upper = |L| or L[upper].v > (v + d)) then
19         return ;
```

The cost of initialization is dominated by sorting the grid nodes along the sort dimension. The sort algorithm of the STL used here implements introsort [66]. The introsort algorithm has a worst case sequential

time complexity of $\mathcal{O}(n \log(n))$ for n elements and is at least as fast as quicksort on average. The computation of the sort value and the creation of the sorted list are each accomplished in $\mathcal{O}(n)$ sequential time. If a sufficient number of processors is available, the sorting time can be theoretically reduced to $\mathcal{O}(log(n))$ with $\mathcal{O}(n \log(n))$ operations (Corollary 4.5 in [44, p.173]).

The runtime of Algorithm 5 depends on the concrete structure of the grid and the slices. Let us consider the case of a full regular cubic volume grid. The sequential time for finding the closest node is reduced to checking the two adjacent slices. For n grid nodes in a cube, the number of nodes per slice is $n^{2/3}$. Thus the sequential runtime of Algorithm 5 is $\mathcal{O}\left(n^{2/3}\right)$. The time of the trivial full search is $\mathcal{O}(n)$. For $n = 10^6$ Algorithm 5 is thus $n^{1/3} = 10^2 = 100$ times faster. The rotation improves the concrete runtime, not the asymptotic one.

If a surface grid is considered, such as the torso surface, the number of grid nodes is moderate (20000-30000 torso triangulation nodes with $4\ mm$ side length). To estimate the complexity, we simply the torso to a cube that is aligned with the coordinate system. Each side contains $n/6$ nodes. So if the closest node is on the left or right we find an upper bound of $\mathcal{O}(n)$. However, the real number of node comparisons is still reduced below $n/6$. In the case of top, bottom, front or back side we only need to search a slice in the shape of an empty square. This is very fast, since the cube side length has $\sqrt{n/6}$ nodes only. We thus only have to check $4 \cdot \sqrt{n/6} \in \mathcal{O}(\sqrt{n})$ nodes.

Algorithm 5 may be parallelized well, knowing the grid width w. In this case, it is sufficient to check all grid nodes $\vec{g}$ with $x(\vec{g}) \in \left[x(\vec{s}) - \left(w/2 \cdot \sqrt{3}\right), x(\vec{s}) + \left(w/2 \cdot \sqrt{3}\right)\right]$ because $w/2 \cdot \sqrt{3}$ is the maximum distance of any position inside the grid to the closest grid node. All candidates are independent and can be checked concurrently. If the position is outside the grid, this search will not succeed and we can revert to Algorithm 5.

First Candidate Finding the first element with sort value of at least the given value (Algorithm 6) is an internal operation used by Algorithm 5 and Algorithm 7. It is implemented with logarithmic search in $\mathcal{O}(log(n))$ sequential time with n being the size of the presorted list. With $p \leq n$ processors this operation can be completed in $c \cdot log_{p+1}(n)$ time with an EREW simply by dividing the current search range into $p+1$ equal sections and checking the p separators.

Algorithm 6: Find first entry in sorted list with sort value $v \geq$ a given value

Data : list L of {value v_j, node index i_j} pairs sorted according to v_j
Input : sort value $v \in \mathbb{R}$
Output : index i' in L of first element with $v_{i'} \geq v$

```
1  lowerLimit ← 0 ;    // lower limit of index search range
2  upperLimit ←| L | −1 ;    // upper limit of index search range
3  if v ≤ L[lowerLimit].v then
4      i' ← lowerLimit; return ;    // below valid range
5  if v L[upperLimit].v then
6      i' ← upperLimit + 1; return ;    // above valid range
7  while true do
8      i ← lowerLimit + ⌊(upperLimit − lowerLimit) ÷ 2⌋ ;    // pick the middle
9      if L[i].v < v then
10         lowerLimit ← i + 1 ;    // search in upper half
11     else
12         if L[i − 1].v < v then
13             i' ← i; return ;    // found it
14         else
15             upperLimit ← i − 1 ;    // search in lower half
```

Nodes Within Distance Range The third grid algorithm is the search for all nodes within a certain distance range to a given position (Algorithm 7). This is for example required in Algorithm 6. The idea, similar to the closest node search, is to start at the first grid node with at least the sort value of the position and to search in both directions. The asymptotic time complexity depends on the search range.

Algorithm 7: Find nodes within distance range

Data : list L of {value v_j, node index i_j} pairs sorted according to v_j
Input : position $\vec{s} \in \mathbb{R}^3$; distance range $\{d_{min}, d_{max}\} \in \mathbb{R}^+ \times \mathbb{R}^+$ with $d_{min} \leq d_{max}$
Output : set of grid nodes indices $G' \subseteq [0, \mid G \mid -1]$

```
1  determine sort value v of s⃗;
2  determine index i of the first element in L with L[i].v ≥ v (Algorithm 6) ;   // start index
3  lower ← min(max(i, 0), |L| − 1) ;   // select first lower candidate
4  upper ← max(min(i + 1, |L| − 1), 0) ;   // select first upper candidate
5  while true do
6      if lower > −1 then
7          if d_min ≤| s⃗ − G[L[lower].i] |≤ d_max then G′ ← G′ ∪ {L[lower].i};
8          lower ← lower − 1 ;   // move one further
9      if upper <| L | then
10         if d_min ≤| s⃗ − G[L[upper].i] |≤ d_max then G′ ← G′ ∪ {L[upper].i};
11         upper ← upper + 1 ;   // move one further
12     if (lower = −1 or L[lower].v < (v − d_max)) and
13       (upper =| L | or L[upper].v > (v + d_max)) then
14         return ;
```

3.3.3 Limited Search Volume Constraint

The physical search volume consists of a continuous 3D volume defined by its boundary, which may consist of an inner and an outer part. For realistic applications an arbitrary volume is defined by a triangulation of its surface as it is done in a boundary element model. The constraint of a limited search volume is violated if a sensor is positioned outside. The boundary itself is considered to be inside. To fix this violation, the sensor needs to be moved back inside, but causing minimum deviation. Thus, we need to find the closest position on the boundary and move the sensor to it (Algorithm 8). The projection point on the boundary implicitely computed in Line 2 of Algorithm 8 can be cached and reused in Line 5.

Algorithm 8: Constraint violation fix for continuous search volume

Data : continuous search volume $V \subseteq \mathbb{R}^3$
Input : multiset $S \subseteq \mathbb{R}^3$ of element positions
Output : multiset $S' \subseteq V$ of element positions

```
1  foreach s⃗ ∈ S do
2      if s⃗ ∈ V (Algorithm 3) then
3          S′ ← S′ ∪ {s⃗};
4      else
5          find projection on boundary s⃗′ of s⃗ (Algorithm 4);
6          S′ ← S′ ∪ {s⃗′};
```

An important question is whether and how the elements can move along the boundary. In practice, this is important because the ideal sensor position for MCG is inside the heart muscle, thus inside the body. The optimization process must thus be able to move the sensors on the body surface to some optimum. With the penalty approach the goal function value would increase close to the boundary, so an optimizer would either not let an element cross the boundary or move it back inside, depending on the type of optimizer.

With the fixing strategy outlined above, the optimizer would move the element across the boundary. The fixing would move it back inside, however along the boundary normal instead of the path it came. Thus after fixing, the sensor would be at another position on the boundary, which is closer to the desired position than the previous one. The elements thus move along the boundary. The elements are moved out of the search volume by the optimizer in the direction of the expected optimum and are moved back inside to another position by the constraint. Any barrier in the goal function on the boundary between the exit and entry point is tunneled by this step.

3.3.4 Minimum Sensor Distance Constraint

The magnetometer available for biomagnetic instrumentation have a certain size. The Argos 200 System for example has square 8x8mm sensors [70], the Elekta Neuromag MEG has integrated chips of 21x21mm side length [54]. The currently available optic magnetometers contain cells of cesium vapor of 20mm length [6]. Newer technologies will however allow sensors diameters of well below 1 cm [10]. The minimum distance is thus a necessary constraint and needs to be parameterized to an arbitrary minimum distance.

The task of resolving all distance violations while causing minimum deviation is in itself an optimization problem. However, it is not practical nor necessary to apply a generic optimization technique with its own parameters. Instead, we shall define a simple iterative approach. To understand the problem of minimum distance violations, it is necessary to look at typical scenarios.

Minimum Distance in Infinite Space A simple scenario is that two or few elements follow a gradient and only cross each others path. No fixing is strictly neccessary since we really only want the final solution to satisfy the minimum distance. However, violations can build up across iterations, so we should fix this violation in each iteration. We can however introduce a distance tolerance level to smooth this process. Here, the advantage of fixing over penalty is apparent. We are able to use the information of which element exactly causes the violation and make a local minimum-impact fix.

A second scenario is that elements accumulate in one area. This may be due to the goal function and would then be a recurring problem across iterations. A natural approach here is to choose a representative, e.g. the one with the maximum number of violations, and to move the remaining elements away, e.g. radially (Figure 17). This would improve the situation, but not fix all minimum distance violations. We need to repeat this step several times.

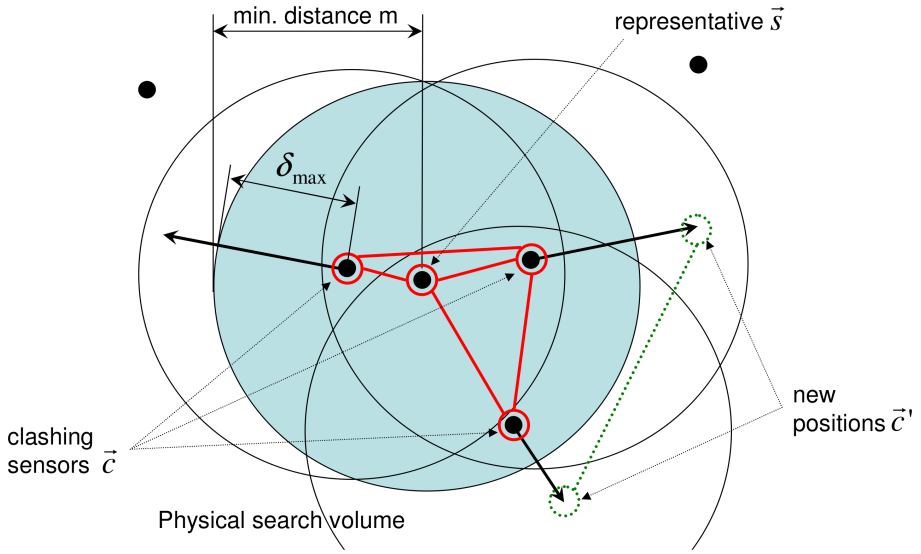

Figure 17: Restoring the minimum distance by radial movement. Black dots indicate sensor positions, red circles and connection lines mark clashes. Large circles indicate the minimum distance of each sensor, the shaded large circle for the representative of the cloud. Exemplary two new sensor positions are indicated by small dotted circles.

This approach is formalized in Algorithm 9. The method stops once all violations are eliminated. The

constraint violation severity is measured by the mean of all violations, each within $(0, m]$. Because the violation severity reduction is strong only in the first iterations, the number of iterations is limited to $\hat{i}$ (Line 2). Later iterations fix micro violations, which is unnecessary and time consuming. The constraint is considered satisfied if the mean violation is below a tolerance threshold of $t \cdot m$ with $t \in [0, 1)$ (Line 14). After the last iteration, a zero-tolerance fix is performed. Theorectially, an oscillation between three or more collinear elements may occur. Such oscillations would disperse relatively quickly by themselves, because the elements are spread appart. To eliminate the possiblility of longer oscillations, a random angular deviation of the otherwise radial movement vector is added in Line 9.

Algorithm 9: Constraint violation fix for minimum distance

Parameter: $m \in \mathbb{R}^+ \setminus \{0\}$ = min. distance; $t \in [0, 1)$ = rel. tolerance;
$\hat{i} \in \mathbb{N} \setminus \{0\}$ = max. #iterations; $\hat{\alpha} \in [0, pi]$ = max. angular dispersion;
$\xi \in [0, 1]$ = influence of max. violation;
Input : multiset $S \subseteq \mathbb{R}^3$ of element positions
Output : set $S' \subseteq \mathbb{R}^3$ of element positions with distances within tolerance

1 $i \leftarrow 0$; *// iteration count*
2 **while** *mean violation in* $S > 0$ *and* $i < \hat{i}$ **do**
3 find one $\vec{s} \in S$ with max. # clashing elements $C \subseteq S \setminus \{\vec{s}\}$;
4 let $\delta_{max} \in [0, m]$ be the max. violation of all $\vec{c} \in C$;
5 $S' \leftarrow S \setminus C$; *// carry over remaining elements*
6 **foreach** $\vec{c} \in C$ **do**
7 **if** $\vec{c} = \vec{s}$ **then** $\vec{d} \leftarrow$ random vector with length 1;
8 **else** $\vec{d} \leftarrow (\vec{c} - \vec{s}) \,/\, |\, \vec{c} - \vec{s}\, |$; *// $\vec{d}$ = normalized direction of movement*
9 add random angular deviation $\leq \hat{\alpha}$ to $\vec{d}$; *// to disperse oscillations, if any*
10 $l \leftarrow \xi \cdot \delta_{max} + (1 - \xi) \cdot (m - |\, \vec{c} - \vec{s}\, |)$; *// l = length of movement vector*
11 $\vec{c}' \leftarrow \vec{c} + l \cdot \vec{d}$; *// move clashing element away*
12 $S' \leftarrow S' \cup \{\vec{c}'\}$;
13 $S \leftarrow S'$; $i \leftarrow i + 1$;
14 **if** *mean violation of* $S' > t \cdot m$ **then** abort;

For the length of the movement vector, two extremes exist. On one hand, we could move all clashing elements to exactly the minimum distance ($\xi = 0$). This would cause minimal deviation, but two clashing positions may come closer to each other. On the other hand, we could move all clashing elements away by the maximum violation δ_{max} within the clashing elements ($\xi = 1$). This would at least maintain distances within the group of clashing positions. But it would cause stronger deviation from the input configuration. In Line 10 we use a combination of both.

A third scenario is that of a densely filled search volume. If the number of elements inside the search volume is close to, or even more than half of, the number of elements that the search volume can fit considering the minimum distance, minimum distance violations are unavoidable. The fixing reduces to spreading elements evenly. Algorithm 9 is suitable for this case as well.

Minimum Distance Inside Search Volume The combination of a limited search volume and a minimum distance constraint creates the additional scenario that fixing one constraint causes violations of the other. For example, if a number of elements accumulate in one area on or close to the boundary, then taking the central element and moving the others away may push some across the boundary. The search volume fix may in turn violate the minimum distance constraint. This can be resolved simply by giving nodes close to the boundary precedence in the selection of the representative. The combined fixing strategy is (Algorithm 10) is identical to Algorithm 9 except that elements with boundary distance of less than m are given precedence in the representative selection (Lines 4-6) and that the limited search volume condition is enforced initially (Line 1) and in each iteration (Line 16).

Algorithm 10: Constraint violation fix for continuous search volume with minimum distance

Parameter: $m \in \mathbb{R}^+ \setminus \{0\}$ = min. distance; $t \in [0, 1)$ = rel. tolerance;
$\hat{i} \in \mathbb{N} \setminus \{0\}$ = max. #iterations; $\hat{\alpha} \in [0, pi]$ = max. angular dispersion;
$\xi \in [0, 1]$ = influence of max. violation;
Input : multiset $S \subseteq \mathbb{R}^3$ of element positions
Output : set $S' \subseteq \mathbb{R}^3$ of element positions with distances within tolerance

1 → fix search volume constraint in S (Algorithm 8);
2 $i \leftarrow 0$; *// iteration count*
3 **while** *mean violation in* $S > 0$ *and* $i < \hat{i}$ **do**
4 → find multiset of elements $B \subseteq S$ with distance to boundary $< m$;
5 → **if** $B \neq \emptyset$ **then** pick $\vec{s} \in B$ with max. # clashing elements $C \subseteq S \setminus \{\vec{s}\}$;
6 → **else** pick $\vec{s} \in S$ with max. # clashing elements $C \subseteq S \setminus \{\vec{s}\}$;
7 let $\delta_{max} \in [0, m]$ be the max. violation of all $\vec{c} \in C$;
8 $S' \leftarrow S \setminus C$; *// carry over remaining elements*
9 **foreach** $\vec{c} \in C$ **do**
10 **if** $\vec{c} = \vec{s}$ **then** $\vec{d} \leftarrow$ random vector with length 1;
11 **else** $\vec{d} \leftarrow (\vec{c} - \vec{s}) \,/\, |\, \vec{c} - \vec{s}\, |$; *// $\vec{d}$ = normalized direction of movement*
12 add random angular deviation $\leq \hat{\alpha}$ to $\vec{d}$; *// to disperse oscillations, if any*
13 $l \leftarrow \xi \cdot \delta_{max} + (1 - \xi) \cdot (m - |\, \vec{c} - \vec{s}\, |)$; *// l = length of movement vector*
14 $\vec{c}' \leftarrow \vec{c} + l \cdot \vec{d}$; *// move clashing element away*
15 $S' \leftarrow S' \cup \{\vec{c}'\}$;
16 → fix search volume constraint in S' (Algorithm 8);
17 $S \leftarrow S'$; $i \leftarrow i + 1$;
18 **if** *mean violation of* $S' > t \cdot m$ **then** abort;

3.4 Discrete Case

3.4.1 General Considerations

The discrete case is necessary for sensor arrangement optimization for two reasons. First, in each iteration of the optimization we need to compute at least one goal function value which is very time-consuming in the continuous case. The goal function computation requires a full forward solution with arbitrary sensor positions. This is significantly accelerated by defining a finite grid of valid sensor positions and directions and precomputing the forward solution for all position-direction combination once. In the concrete case of leadfield matrix based goal functions, each sensor position is represented by one row in the leadfield matrix. Thus the leadfield matrix of a set of sensor positions on the discrete grid can determined simply be concatenating the rows of the respective sensor positions.

The second reason for a discrete search space is that discrete optimization algorithms, such as tabu search, require it. The discrete case thus applies to all of the optimization techniques. The continuous optimizers are run in a quasi-continuous fashion. The distance between grid nodes should therefore be significantly smaller than the minimum distance.

3.4.2 Discrete Search Volume Realization

The discrete search volume is realized as a finite grid of positions. A regular grid can be derived from a continuous search volume by obtaining its bounding box and scanning this bounding box with a defined sampling width. For each candidate position, we determine whether it is inside the continuous search volume (Algorithm 3) and if so add it to the grid. The check whether an arbitrary position is inside reduces to checking whether it is identical to its closest grid node (Algorithm 5). The projection on the boundary operation also reduces to finding the closest grid node. A special case is a single layer of grid nodes around the body surface. This case is also covered with this approach.

3.4.3 Limited Search Volume Constraint

A limited search volume violation can be detected by comparing the current position to its closest grid node (Algorithm 5). Here, the efficient implementation of Algorithm 5 yields a significant performance gain, because this test is done for every position in every iteration of the optimization. To fix a violation, we simply snap back to the grid (Algorithm 11). The deviation caused by the constraint fix depends on the sampling width of the grid. A finer grid yields a smaller deviation but also makes an efficient implementation all the more necessary.

Algorithm 11: Constraint violation fix for discrete search volume

Data : discrete search grid $G \subseteq \mathbb{R}^3$
Input : multiset $S \subseteq \mathbb{R}^3$ of element positions
Output : multiset $S' \subseteq G$ of element positions

1 **foreach** $\vec{s} \in S$ **do**
2 find closest grid node $\vec{s}' \in G$ to $\vec{s}$ (Algorithm 5);
3 $S' \leftarrow S' \cup \{\vec{s}'\}$;

3.4.4 Minimum Sensor Distance Constraint

The problem of minimum sensor distance by itself in Section 3.3.4 remains valid for the discrete case as well. We therefore only need to modify Algorithm 10 and yield Algorithm 12. We enforce the limited discrete search volume condition initially (Line 1) and in each iteration (Line 14). Second, we handle the case that an element that was moved away from the cloud representative is moved back below minimum distance due to the grid width. For every such element, we simply pick the closest grid node that satisfies the minimum distance (Lines 15- 20).

3.4.5 Limited Direction Constraint

A practical representation of a direction vector are spherical coordinates. Since only the orientation is important and not the length, it is sufficient to store the two angles $\phi \in [0, \pi]$ (zenith) and $\theta \in [0, 2\pi]$ (azimuth). A regular grid of angles may be defined by scanning ϕ and θ with 45°, 30°, 15° etc. step width. The resulting list of valid directions needs to be indexed. The north and south pole may get the indices 0 and 1, and the remaining directions can be numbered from north to south continuing in positive θ direction.

The two conversion methods indexToDirection and directionToIndex both only require $\mathcal{O}(1)$ time, because they are analytical. DirectionToIndex additionally brings ϕ and θ back into the valid range and snaps them to the grid. The constraint fix (Algorithm 13) snaps the continuous directions to the grid by reusing both conversion methods. Since all directions are independent this may be parallelized with a speed up $\sim$ # processors on an EREW.

The combination of the constraints for limited search volume, minimum distance and limited directions is simple, because position constraints are decoupled from the direction constraint. They can be executed one after the other. An adjustment of the direction with respect to the movement of the sensor by the position constraints can not generally be made, because the direction represents the vectorial field component to be measured. Algorithm 13 may even be executed concurrently to Algorithm 12 on an EREW.

Algorithm 12: Constraint violation fix for discrete search volume with minimum distance

Parameter: $m \in \mathbb{R}^+ \setminus \{0\}$ = min. distance; $t \in [0,1)$ = rel. tolerance;
$\hat{i} \in \mathbb{N} \setminus \{0\}$ = max. #iterations; $\hat{\alpha} \in [0, pi]$ = max. angular dispersion;
$\xi \in [0,1]$ = influence of max. violation;
Input : multiset $S \subseteq \mathbb{R}^3$ of element positions
Output : set $S' \subseteq \mathbb{R}^3$ of element positions with distances within tolerance

1 fix search volume constraint in S (Algorithm 11);
2 $i \leftarrow 0$; *// iteration count*
3 **while** *mean violation in* $S > 0$ *and* $i < \hat{i}$ **do**
4 pick $\vec{s} \in S$ with max. # clashing elements $C \subseteq S \setminus \{\vec{s}\}$;
5 let $\delta_{max} \in [0, m]$ be the max. violation of all $\vec{c} \in C$;
6 $S' \leftarrow S \setminus C$; $C' \leftarrow \emptyset$;
7 **foreach** $\vec{c} \in C$ **do**
8 **if** $\vec{c} = \vec{s}$ **then** $\vec{d} \leftarrow$ random vector with length 1;
9 **else** $\vec{d} \leftarrow (\vec{c} - \vec{s}) \,/\, |\, \vec{c} - \vec{s}\, |$; *// $\vec{d}$ = normalized direction of movement*
10 add random angular deviation $\leq \hat{\alpha}$ to $\vec{d}$; *// to disperse oscillations, if any*
11 $l \leftarrow \xi \cdot \delta_{max} + (1 - \xi) \cdot (m - |\, \vec{c} - \vec{s}\, |)$; *// l = length of movement vector*
12 $\vec{c}' \leftarrow \vec{c} + l \cdot \vec{d}$; *// move clashing element away*
13 $S' \leftarrow S' \cup \{\vec{c}'\}$;
14 fix search volume constraint in C' (Algorithm 11);
15 **foreach** $\vec{c}' \in C'$ **do**
16 **if** $|\, \vec{c}' - \vec{s}\, | < m$ **then**
17 find closest search volume node $\vec{c}''$ to $\vec{c}'$ with $|\, \vec{c}'' - \vec{s}\, | \geq m$ (Algorithm 7);
18 $S' \leftarrow S' \cup \{\vec{c}''\}$;
19 **else**
20 $S' \leftarrow S' \cup \{\vec{c}'\}$;
21 $S \leftarrow S'$; $i \leftarrow i + 1$;
22 **if** *mean violation of* $S' > t \cdot m$ **then** abort;

Algorithm 13: Constraint violation fix for discrete directions

Parameter: $n \in \mathbb{N} \setminus \{0, 1\}$ = # grid nodes within $[0, \pi]$ inducing grid of directions
$G \subset [0, \pi] \times [0, 2\pi]$
Input : multiset $D \in \mathbb{R}^2$ of $\{\phi, \theta\}$ pairs
Output : multiset $D' \in G$ of $\{\phi, \theta\}$ pairs

1 **foreach** $d \in D$ **do**
2 $d' \leftarrow indexToDirection(directionToIndex(d))$;
3 $D' \leftarrow D' \cup \{d'\}$;

4 Object-Oriented Design and Development Process

4.1 Overview of SimBio

SimBio [24] is a generic distributed simulation environment. Its objective is the improvement of clinical and medical practices by introducing large-scale numerical simulation capabilities to bio-medical problems. It comprises subsystems for the representation of the physical problem, numerical solutions, inverse problem solving, optimization and visualization. Its key feature is the ability to use individual patient data as input for modeling and simulation. Example applications are the localization of electromagnetic sources inside the human brain or bio-mechanical simulations and prosthesis design. In general, SimBio is suitable for non-invasive diagnosis and pre-operative planning. The software framework consists of several server-based subsystems, which are connected to a graphical user interface at the application site through CORBA middleware. The computationally expensive simulation algorithms are designed to utilize parallel hardware.

Inverse Toolbox The subsystem of interest is the inverse toolbox. The objective of the inverse toolbox is to provide a collection of inverse procedures and algorithms for error assessment. The design is intended to be as generic as possible, supporting interoperation with external programs. To date, two types of problems are supported: (1) problems with a continuous parameter space and (2) problems with a discrete parameter space. In the continuous case, fitting single or multiple, fixed, rotating or moving dipoles is implemented through non-linear optimization. In the discrete case, dipole positions are assumed to be fixed. Both, electric and magnetic field recordings are supported.

The inverse toolbox consists of a number of generic classes, such as optimizers and goal functions. Because their direct usage by a client is too complex, three user interface (UIF) layers have been defined. The class `UIF1_c` provides functions for all typical usage scenarios and accesses the toolbox classes directly. `UIF2_c` provides functions with file names for input and output which call `UIF1_c`. Finally, UIF3 is a command line interpreter calling `UIF2_c`. `UIF1_c` instantiates several toolbox objects and an analyzer object (Figure 18), which encapsulates a particular task requested by a user. For example `anAnalyzerInverseDipoleFit_c` creates the necessary objects to do a dipole fit. The analyzer instantiates several more toolbox objects, such as an optimizer and a goal function. Both, `UIF1_c` and the analyzer classes implement the Facade [27, p. 185] design pattern. They provide a simplified interface to a more complex subsystem by delegating tasks to a set of hidden objects. Figure 19 shows an example of a setup of toolbox components to perform a moving dipole fit. The respective UIF1 function creates and initializes the objects with data and links and finally tells the analyzer to run the optimization. The optimizer makes use of the goal function iteratively, which in turn utilizes more components.

System-wide Conventions Data are typically stored in utility classes, which are `utVector_t`, `utMatrix_t` and `utBlock_t`. These classes define standard mathematical vector and matrix operations, such as cross product or matrix multiplication. All three are parameterized in their number format using C++ templates. The coding standard defines the class name prefixes ”`an`” for analysis and ”`ut`” for utilities, as well as postfixes ”`_c`” for class, ”`_t`” for template class and ”`_s`” for struct. Class members are prefixed with ”`m_`”.

4.2 Optimizers

The role of the optimizer classes derived from `anAbstractOptimizer_c` (Figure 20) is to implement different numerical optimization techniques. Their input is a two dimensional set (`utMatrix_c`) of parameters, representing the initial solution. Typically, each row defines one dipole or sensor in terms of position and orientation. But the optimizers are generic, so that they work on any set of parameters. The optimization process minimizes the goal function value of the current solution by modifying it. In order to support any kind of goal function, a protected reference of type `anAbstractGoalFunction_c` is used. In each iteration, the current solution is passed to the goal function object, which returns the goal function value. The output is a parameter matrix of the same dimension as the input representing the result.

Settings and stop criteria are passed in form of a `utVector_t<double>`, which allows to define the same get and set methods in the abstract base class. Additionally, the number of settings of a concrete optimizer may be increased by adding functionality, without needing to change the interface. As in most of

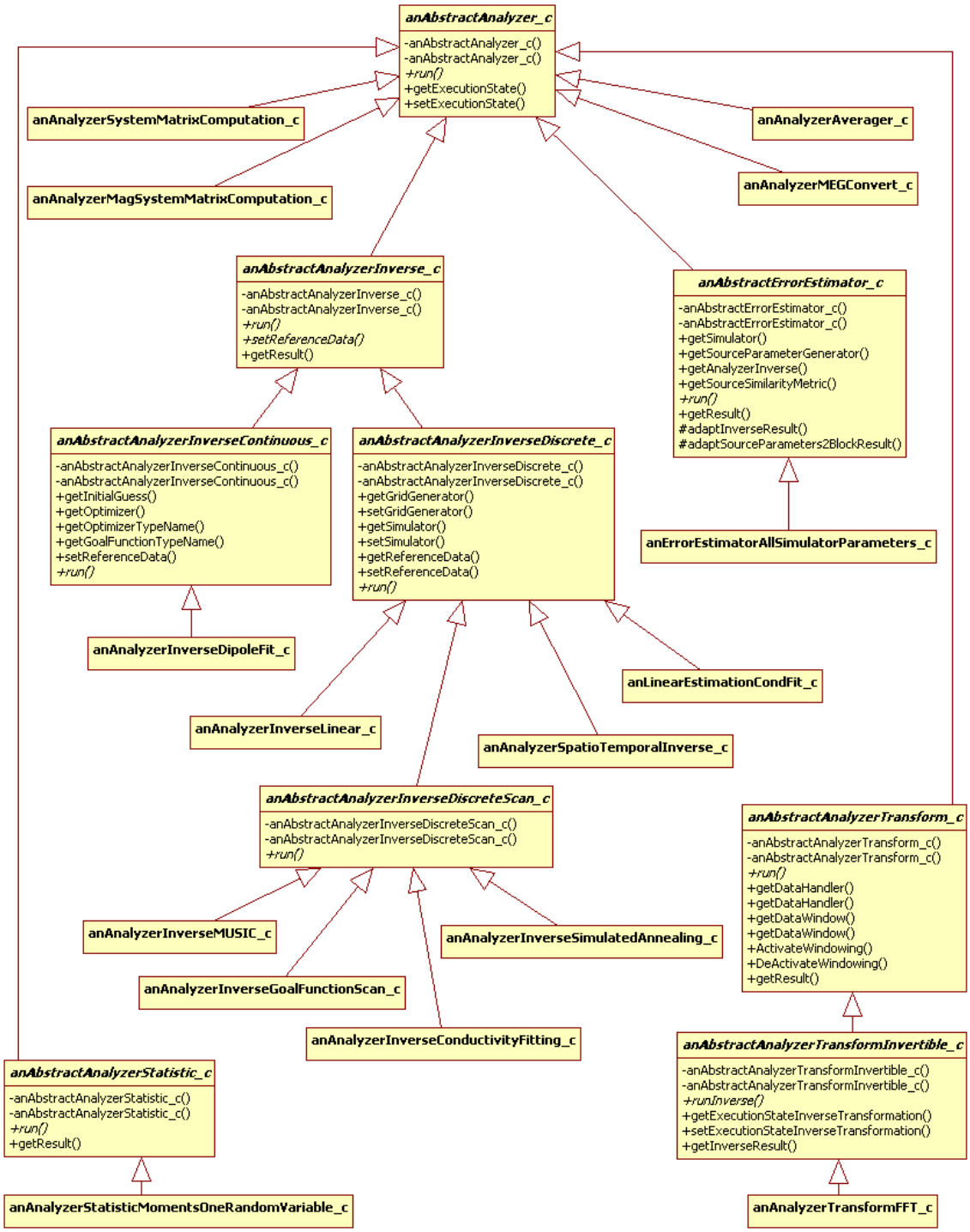

Figure 18: Analyzer abstraction hierarchy (showing operations of abstract classes only)

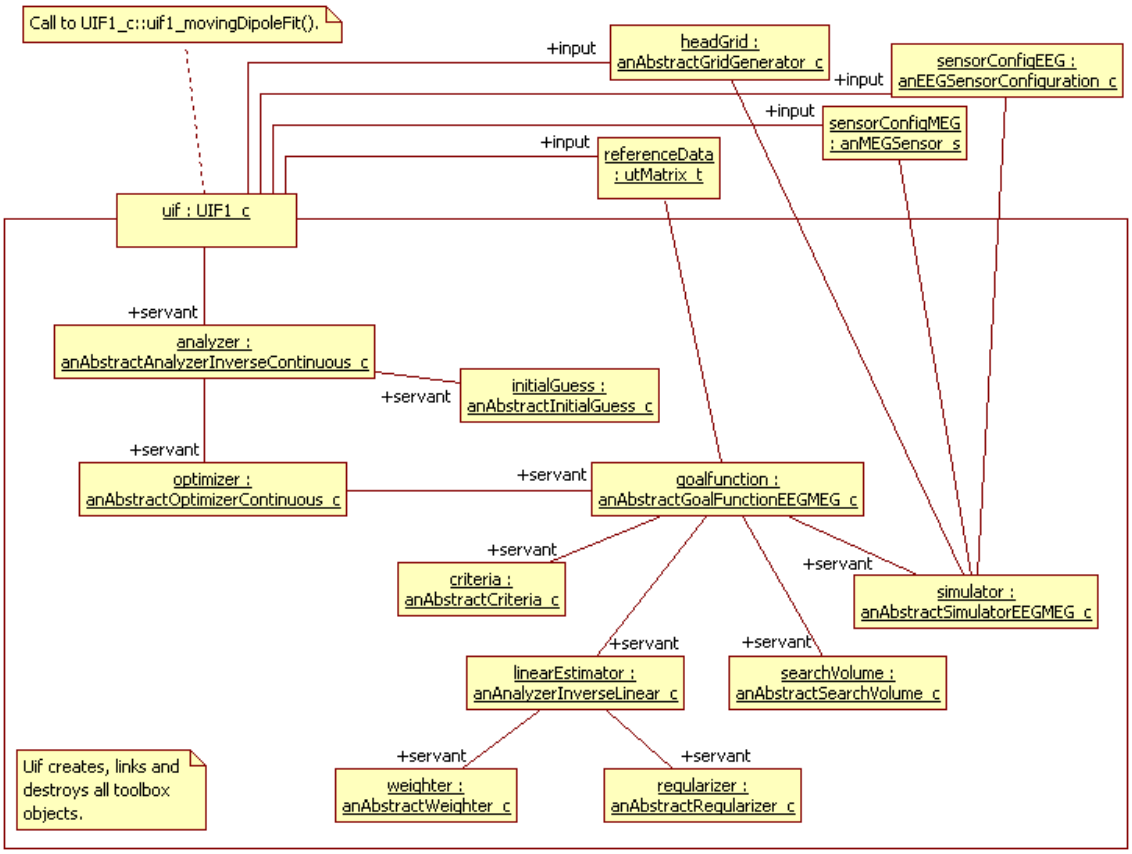

Figure 19: Object diagram of setup for a moving dipole fit using electrical and magnetic sensors.

the SimBio classes, the method `getTypeName(): std::string` is defined, which simplifies output of information and error messages.

Independent Constraint The definition of the optimization problem in general consists of two parts: the goal function and the definition of the, possibly constrained, search space. The pre-existing SimBio implementation only supported one type of constraint: a limited physical search volume, typically a sphere fitted to model the human brain. This constraint was hardcoded into `anAbstractGoalFunctionEEGMEG_c` through a penalty strategy using an `anAbstractSearchVolume_c` reference. This of course limited the use of this constraint to this particular application.

The new class `anAbstractConstraint_c` (Figure 20) is the independent generic representation of a constrained search space definition. The optimizer can use the constraint interface to check and restore the validity of the constraint without knowing its particular structure. The usage of a constraint consists of two steps: (1) call `isSatisfied()` and/or `fix()` on the initial guess to enforce a valid initial solution and (2) call `fix()` after each iteration step to obtain a close valid solution. This design decouples the problem definition from the optimization approach and also the constraint definition from the goal function. Thus any optimizer can minimize any goal function inside any constrained search space. In particular, we can also fit dipoles, thus performing an inverse solution, with the new PSO optimizer. Additionally, application-specific goal functions and constraint definitions or generic optimization techniques can be added without recompiling the other two components, respectively.

Default Constraint In order to support the unconstrained search space and to be fully compatible with the various pre-existing client classes of `anAbstractOptimizer_c`, the reference `m_constraint` is initialized to an instance of `anConstraintEmpty_c` (Figure 22) by default if no constraint is passed in the constructor. This is implemented by the very simple static factory method `createDefaultConstraint()` in `anAbstractOptimizer_c`. Thus, the pre-existing optimizers `anOptimizerMarquardt_c` and `an-`

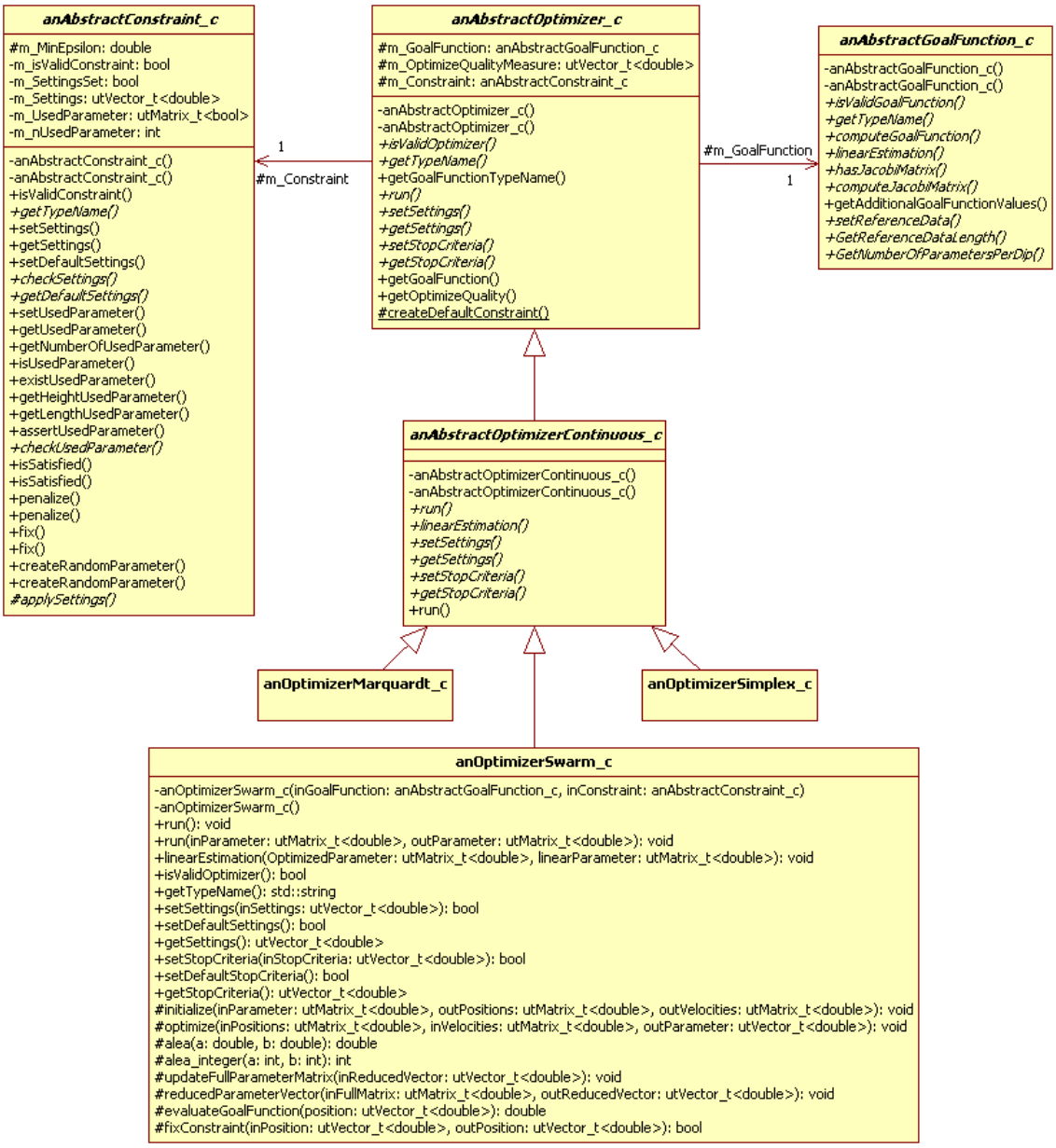

Figure 20: Optimizer abstraction hierarchy (showing operations and attributes of abstract classes and operations of `anOptimizerSwarm_c`)

`OptimizerSimplex_c` do not need to be touched to run as they did till now. They only need to be instrumented to make use of the new constraint.

Particle Swarm Optimizer The new class `anOptimizerSwarm_c` implements the particle swarm optimization outlined in Section 2.4. The publicly available standard PSO [1] algorithm was restructured in the following way: C arrays were replaced with `utMatrix_t` and `utVector_t` objects. The interface of `anAbstractOptimizerContinuous_c` was implemented. Parameters and stop criteria are managed only through the set and get method. Many global variables were put in their proper scope. The two dimensional parameter set used in SimBio is linearized internally and conversion methods were introduced for interaction with the goal function and the constraint. The linearization includes support for arbitrary definitions of unused parameter. Unused parameter are not altered by the optimizer, but are used to compute the goal function and fix the constraint (see Section 4.3).

The initial solutions of the particles are determined using the method `createRandomParameter()`

of the constraint. The random initialization is thus delegated to the constraint, whose responsibility it is. After each iteration the method `fix()` of the constraint is called to turn the current solution into a close valid solution. Since this moves some particles inside the search space, the respective velocities may need to be adjusted accordingly. Three strategies have been implemented, which can be chosen through a setting. In the current implementation, these strategies are pragmatically chosen by a setting. Much more generic would be the application of the Strategy [27, p. 315] design pattern. A concrete strategy could be a small class derived from an abstract velocity strategy base class. Then strategies could be added without changing the optimizer code and the definition of the valid settings range. A disadvantage would be the computational overhead during runtime.

4.3 Constraints

The role of a constraint class is to represent the high-dimensional search space induced by a particular combination of basic constraints and to provide the methods necessary to integrate it into the optimization process. The four central interface methods (Figure 21) are `isSatisfied()` to check whether a parameter set satisfies the constraint, `fix()` to modify it so that it satisfies the constraint, `penalize()` to compute a penalty to add to the goal function and `createRandomParameter()` to obtain a parameter set that is random within the possibly complex search space and satisfies the constraint. The methods `fix()` and `penalize()` are alternative constraint handling techniques, which both pertain to the definition of the constraint. The method `createRandomParameter()` is necessary to generate initial guesses. It is also closely tied to `anAbstractConstraint_c`, because it requires full access to the definition of the constraint.

Figure 21: Constraint interface sections

Not in all constraint cases all interface methods will be simple to implement. And not in all application cases, all interface methods will be strictly necessary. Therefore, default implementations, so called hook methods, are provided in the abstract base class, throwing and exception with an appropriate error message. This allows incremental development of new constraint classes. For example, for sensor optimization we have many sensors, so `penalty()` does not work well, so we do not implement it Another developer may wish to use the same constraints for 2-3 dipoles, e.g. to separate thalamic and cortical activities in the brain. (S)he can implement `penalty()` any time later without needing to change the interface or recompile any optimizers.

Multiple Virtual Inheritance The nature of application-specific constraints is that they can be decomposed into a set of basic constraints. Each of the basic constraints has its own data structures and methods.

If two constraints are combined, data structures and methods of both should be reused. The natural way is to inherit from both of the basic constraints and to reimplement only the interface methods that realize the combination of both constraints. This can be done repeatedly to combine more than two constraints. For example, the class `anConstraintVolMinDistCont_c` (Figure 22), which defines the minimum distance within a continuous search volume, inherits from `anConstraintVolumeContinuous_c` and `anConstraintMinDistCont_c`. Protected internal minimum distance methods, such as `fixPosition()` which actually moves the elements away from one cloud representative (Lines 6- 12 of Algorithm 9), are reused without any changes. Others, such as `nextPositionToFix()` are reimplemented in the derived class, because elements close to the volume boundary need to be preferred to others. The method `fix()` of `anConstraintVolumeContinuous_c` is even reused as is by calling it with the proper scope operator inside `fix()` of `anConstraintVolMinDistCont_c`. The combination of the resulting class with `anConstraintDirectionContinuous_c` is even simpler, because both parent constraints only touch distinct parameter sets: positions and directions. The method `fix()` of `anConstraintVolMinDistDirCont_c` only calls both base class implementations one after the other.

The second requirement is that all constraints inherit from `anAbstractConstraint_c` (Figure 22) to be exchangeable amongst each other and compatible to any optimizer. This is achieved by diamond-type inheritance [92, p. 399]. If we would choose non-virtual inheritance, two or more `anAbstractConstraint_c`-segments would be allocated in memory for each instance of the derived class. However, we want the settings and the used parameter definition to exist only once per constraint object. We therefore choose virtual multiple inheritance.

Settings Access Template The multiple virtual inheritance requires us to think about the settings vector. If two constraints are combined to one new derived class with only one settings vector, we need to concatenate the settings of both base classes in the getSettings method and we need to check the validity of all settings, ideally by reusing base class code. Additionally, we need to prevent that constraint algorithms access the settings vector directly, because once we concatenate the settings, they are not in the same element of the settings vector any more.

The solution is to encapsulate the settings vector and to provide only controlled access to it, which is shown in Figure 21. We first make the `m_Settings` private so that no derived class code can access it directly. We then apply and extend the template method [27, p. 325] design pattern. The method `setSettings()` is implemented non-virtual in `anAbstractConstraint_c` as a template. It first calls the method `checkSettings()` which is pure virtual, so that the derived class developer has to implement it. If the settings are valid, the template method calls the pure virtual method `applySettings()`, which copies the settings to respective named class attributes. Only these attributes are accessible to the derived class code. With this design a validity check is enforced and no direct access to the unnamed settings is possible. The template method is parameterized with the two virtual methods. The same technique is used for the non-virtual `setDefaultSettings()`. The method getSettings is implemented non-virtual to enforce a hard copy of the settings vector.

Now, in the case of multiple virtual inheritance, each class with two base classes only needs to implement `checkSettings()` and `applySettings()` (Figure 22) in the way that both base class versions are called sequentially on subsets of the input settings vector using the scope operator. New settings of the derived class can be appended at the end of the settings vector and checked and applied as third step. This results in complete reuse of base class functionality.

Encapsulation and Support of Used Parameter Definition The definition of used parameters is stored in a `utMatrix_t<bool>` and defines for each parameter, cartesian coordinates of position and spherical coordinates of direction, whether it is variable (used) or constant (unused). A particular constraint may impose additional restrictions on the used parameter definition. For example, in `anConstraintVolumeContinuous_c` we only want to allow to make whole position triplets to be either used or unused. To enforce this we again apply the template method [27, p. 325]. The method `setUsedParameter()` (Figure 21) is implemented non-virtual, which calls the virtual `checkUsedParameter()` to make the custom check.

The used parameter definition pertains to the definition of the constraint, because it directly restricts the search space by eliminating complete dimensions. However, the optimizer depends on this definition to be able to optimize only the used parameter. But we do not want a second copy to exist in the optimizer. So we make the used parameter definition private in `anAbstractConstraint`ς to encapsulate it and provide efficient access methods to it.

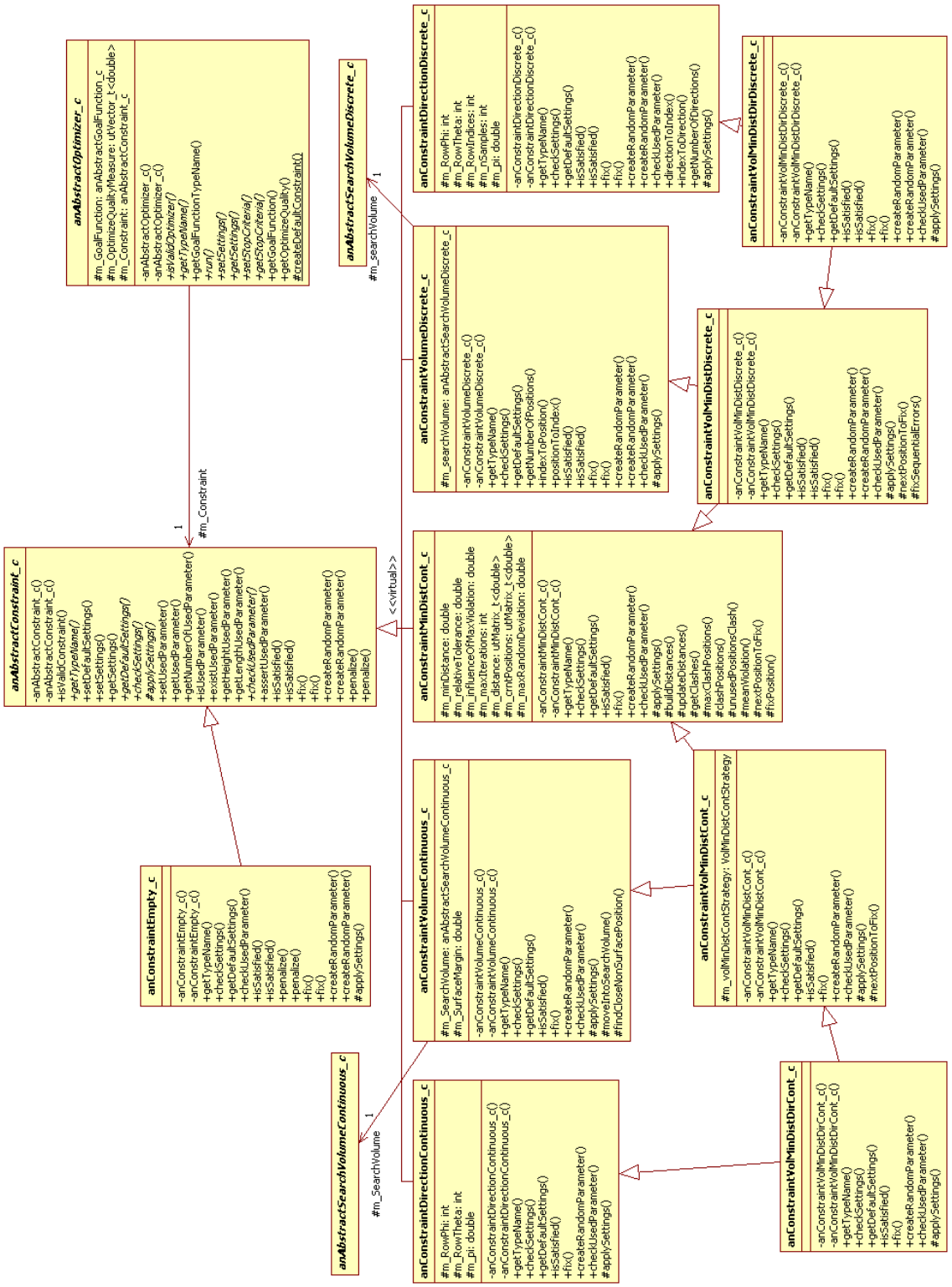

Figure 22: Constraint abstraction hierarchy (omitted attributes of `anAbstractConstraint_c` and all members of both search volume classes)

The developer of a particular constraint may decide to not allow unused parameters at first through `checkUsedParameter()`. (S)he may then develop support for particular sets of unused parameter and allow them by modifying `checkUsedParameter()`. Thus, constraints can be developed incrementally. The used parameter support turns out not to be very difficult. For search volume constraints, we only allow whole positions to be switched off and simply do not alter them within `fix()` and `createRandomParameter()`. For minimum distance constraints, we prefer unused positions in the selection of the cloud representative, which is not moved. And in direction constraints we only need to ignore the unused directions.

The support of a used parameter definition in the constraints actually extends the application area of SimBio. So far only a continuous parameter space with all parameters used and a discrete paramete space with all positions fixed was supported. Now we can switch of any selection of positions and directions in the continuous and discrete case. For example, we may fix some a priori sensors or dipoles and fit the remaining ones. Or we may fix all sensors or dipoles and only optimize their direction. We can also fix all directions and optimize the positions only. This may be of theoretical interest, since we can either choose the same direction for all sensors or dipoles or a balanced number of sensors/dipoles for each spatial dimension. Furthermore, the used parameter definition is not bound to dipoles or sensors any more, but can be customized to any type of optimization constraint.

Shared Parameter Responsibility The combination of several constraints operating on the same parameter matrix requires us to think about which constraint is responsible for which subset of the matrix. Of course, the search volume constraints manage the cartesian position coordinates, e.g. rows one to three. The direction constraints manage the directions, e.g. rows four and five. But when we call `createRandomParameter()` of `anConstraintVolMinDistDirCont_c`, we wish to pass the subtask to base class implementations. Therefore, `createRandomParameter()` of `anConstraintDirectionContinuous_c` is passed a matrix of initial parameters already processed by `anConstraintVolumeContinuous_c` and overrides only the direction parameters in row four and five. To be generic, `createRandomParameter()` of `anConstraintVolumeContinuous_c` is also only responsible for position parameters and leaves direction values as they are. Thus, every constraint has a defined set of parameters, it is responsible for. For the particular use with sensors or dipoles, we define for every constraint the rows which it is responsible for and what the content of the rows is. We can parameterize the row indices used, simply through settings. For example, `anConstraintDirectionContinuous_c` (Figure 22) has the members `m_RowPhi` and `m_RowTheta` which hold the respective row indices. Consequently, `createRandomParameter()` of `anConstraintEmpty_c` returns the parameter matrix as is.

Overloading Methods for Index Support The discrete constraints have a finite number of valid configurations, which can be numbered sequentially. For example, `anConstraintVolumeDiscrete_c` represents a finite set of positions, which need to be stored sequentially in memory anyways. The class `anConstraintDirectionDiscrete_c` holds a finite set of directions. A discrete optimizer, such as tabu search, may wish to operate on indices rather than the parameter values. Additionally, a discrete goal function, such as `anGoalFunctionConditionNumber_c` (Figure 23) holds a finite number of goal function values for a finite set of element configurations. If this goal function is passed parameter values, it needs to convert them to indices of valid goal function sampling points every time. If matching indices would be used, a discrete optimizer could pass indices directly instead of values.

Because both modes are alternatives describing identical information, it is natural to overload the methods `isSatisfied()`, `fix()`, `penalize()` and `createRandomParameter()` with input and output of type `utMatrix_t<int>` instead of `utMatrix_t<double>` (Figure 21). Both versions, may be implemented separately or one may call the other depending on the nature of the constraint. Note that in the index mode we still use a matrix, not a vector, because of the shared parameter responsibilities. The volume constraint has position indices and the direction constraint has indices as well. Both parts do not know about each other. In our particular application case, we have an index matrix with two rows. For other applications, different numbers of index rows are supported by the design. We however need to make explicit definitions which row contains what by settings. We define hook methods in `anAbstractConstraint_c`, which throw an exception with an appropriate error message. The constraint developer can then add index support and/or value support incrementally depending on the requirements at hand. In our case, only the discrete branch of the constraints has both alternatives implemented (Figure 22), while the continuous branch has only value support.

Mapping of Algorithms to Classes and Methods The fix for a limited continuous search volume of Algorithm 8 is implemented in `anConstraintVolumeContinuous_c::fix()`. The minimum distance in an infinite volume of Algorithm 9 is implemented in `anConstraintMinDistCont_c`. The algorithm is decomposed into a number of methods to support code reuse in derived classes. The method `fix()` implements the outer loop. The representative choice in Line 3 is implemented generically in `nextPositionToFix()` which calls `maxClashPosition()` in the concrete implementation. Lines 6- 12 are implemented in `fixPosition()`. The augmentations of Algorithm 10 in Lines 1 and 16 are included in the reimplemented `fix()` of `anConstraintVolMinDistCont_c` and Lines 4- 6 are integrated by reimplementing `nextPositionToFix()`. Algorithm 11 is implemented by `fix()` in the class `anConstraintVolumeDiscrete_c`. Algorithm 12 is represented by `fix()` of the class `anConstraintVolMinDistDiscr_c`. Lines 15- 20 are implemented by the new method `fixSequentialErrors()`, which is called within `fix()`. The discrete direction fix of Algorithm 13 is implemented in `fix` of `anConstraintDirectionDiscrete_c`.

4.4 Goal Functions

The role of `anAbstractGoalFunction_c` (Figure 23) is to declare the interface of goal functions. The central method is `computeGoalFunction()` which computes the goal function value of a particular parameter set. The computation of the Jacobi matrix of partial derivatives also pertains to the goal function. It is however only used by the Levenberg-Marquardt optimization technique and is therefore optional and `hasJacobiMatrix()` is provided to check whether it is supported.

To be able to test the optimizer classes under controlled conditions, thus without the new goal functions, we implemented Rastrigin's function in `anGoalFunctionRastrigin()` as a test goal function. Rastrigin's function is a typical benchmark function for optimization techniques with the following formula:

$$r(\vec{x}) = 10 \cdot d + x_1^2 + x_2^2 + .. + x_d^2 - 10 \cdot (cos(2\pi x_1) + cos(2\pi x_2) + .. + cos(2\pi x_d)) \qquad (\vec{x} \in \mathbb{R}^d) \quad (4.1)$$

Its special properties are that it only has one global minimum, but many regularly arranged local minima and maxima. Close to the global minimum the difference between local minima and the global minimum is very small. But it increases quadratically with increasing distance.

For the optimization of sensors based on measures of the leadfield matrix, we designed and implemented `anAbstractGoalFunctionLeadfield_c` (Figure 23). Its role is to provide the functionality to manage the leadfield matrix and to convert position-direction configurations given as parameter values or indices into global indices that map directly to one row of the leadfield matrix. Since the leadfield matrix implies a fixed global indexing sequence, it also provides the method `getSensors()` to obtain the full list of sensor configurations in the sequence of the leadfield matrix rows.

Since an index matrix passed to the goal function by the optimizer should use the same indexing as the leadfield matrix, `anAbstractGoalFunctionLeadfield_c` reuses the indexing of the respective constraint. Now, since we are by definition using a leadfield matrix, the set of positions and orientations must be discrete. For the optimizer to work on this search space, it requires a constraint that inherits from `anConstraintVolumeDiscrete_c` and `anConstraintDirectionDiscrete_c`. But we do not know which concrete class this would be. So we store two pointers to the constraint object, one of type `anConstraintVolumeDiscrete_c` and one of type `anConstraintDirectionDiscrete_c`. The constraint class may for example be a `anConstraintVolMinDistDirDiscrete_c`. But it may also be any `anConstraintVol***DirDiscrete_c`. The optimizer knows the same object only through a reference of type `anAbstractConstraint_c`.

The concrete leadfield matrix based goal functions, such as `anGoalFunctionConditionNumber_c`, can now be created simply by deriving a class which accesses the leadfield matrix provided by the base class functionality and implements `computeGoalFunction()` (Figure 23).

4.5 Search Volumes

The search volumes (Figure 24) are an integral part of the volume constraints. Until now SimBio only supported continuous search volumes, whose interface was defined in `anAbstractSearchVolume_c`. The interface sections particular to continuous search volumes have been split off into a derived class `anAbstractSearchVolumeContinuous_c` and `anAbstractSearchVolumeDiscrete_c` has been added,

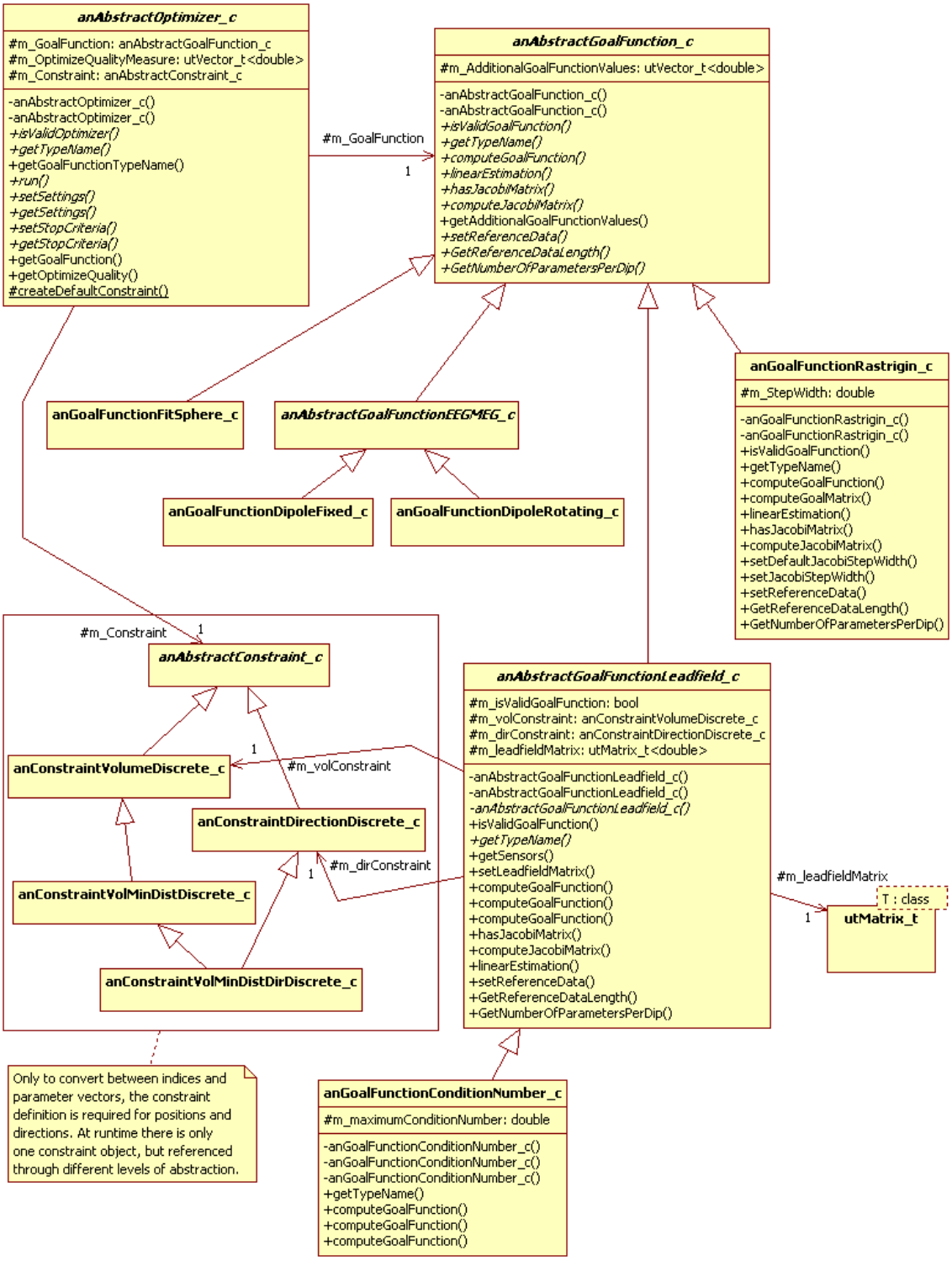

Figure 23: Goal function hierarchy (omitted members of pre-existing classes, constraints and `utMatrix_t`)

which extends the interface with index based methods (Figure 24). All previous references to `anAbstractSearchVolume_c` only need to be replaced with `anAbstractSearchVolumeContinuous_c`. The methods `randomInsidePosition()` was added to the whole hierarchy and `projectionOnSurface()` to the continuous branch. Both have a concrete use in `createRandomParameter()` and `fix()` of the volume dependent constraints.

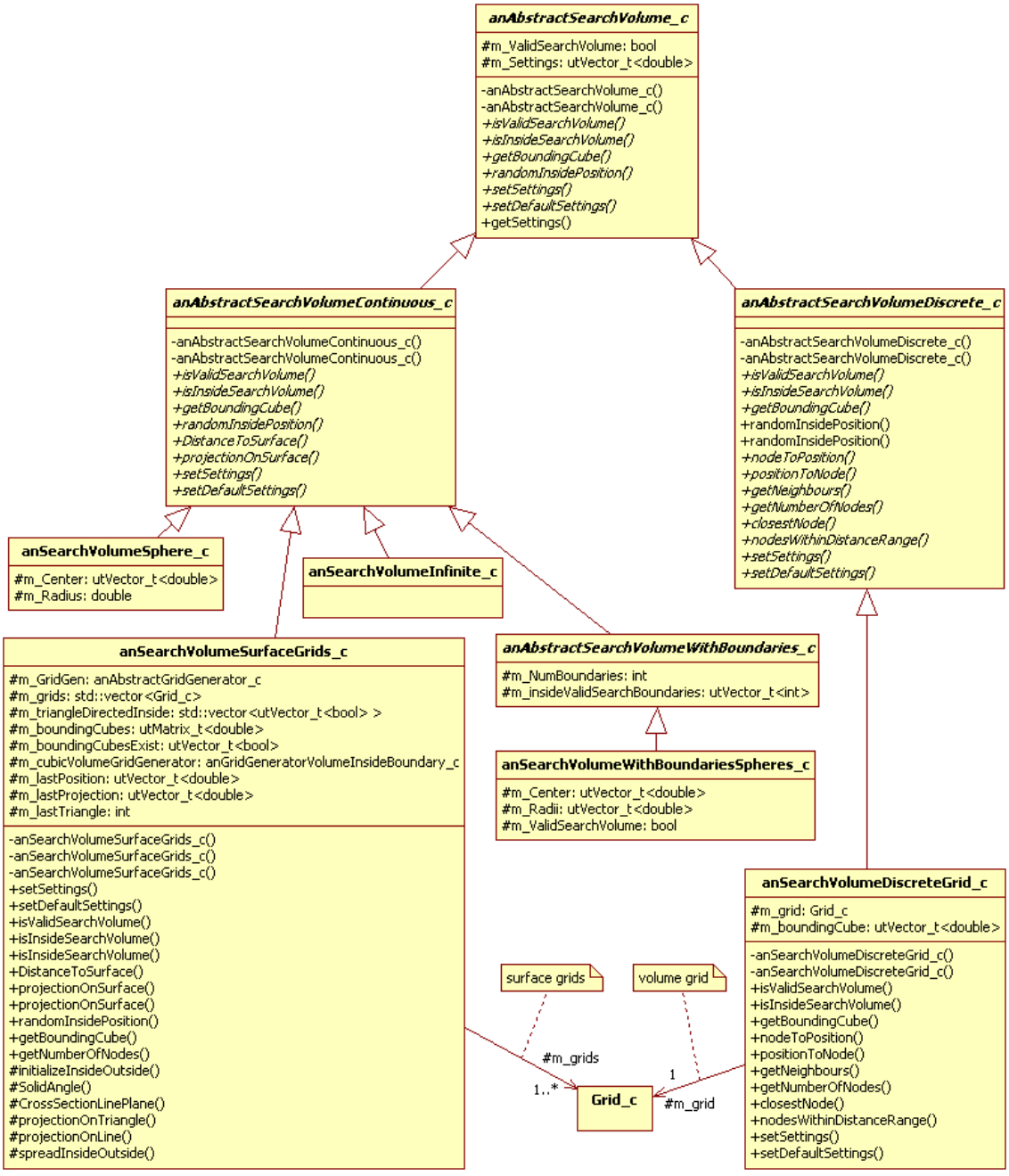

Figure 24: Search volume abstraction hierarchy (omitted operations of pre-existing classes and attributes of `Grid_c`)

The two new classes are `anSearchVolumeSurfaceGrids_c` and `anSearchVolumeGrid_c`. The first represents a set of surface grids, just like the ones a BEM model consists of. One surface can be selected as current search volume. Algorithm 3 of Section 3.3.2 is implemented in `isInsideSearchVolume()` and Algorithm 4 in `projectionOnSurface()`.

The class `anSearchVolumeGrid_c` represents a discrete search volume, which can be thought of as

a grid, basically. One design option would be to inherit from `Grid_c` as well. However, the pre-existing grid generator concept which decouples the representation of a grid from the construction process is not compatible with that. `anSearchVolumeGrid_c` is therefore a proxy class as outlined by the Proxy [27, p. 207] design pattern. The search volume class provides controlled access to a hidden Grid, it is a so called protection proxy. Additional methods, such as `isInsideSearchVolume()`, are implemented, while other non-related methods of `Grid_c` are not accessible. An advantage is that the method names are translated into the search volume context. This facilitates reuse and reduces errors.

4.6 Grid and Grid Generators

The grid generators (Figure 25) in SimBio are an interesting design, because they implement a simplified version of the Builder [27, p. 97] design pattern. Grids appear in many places in the SimBio design. They may represent the surface grid of a BEM model, or a volume grid of a FEM, or a search volume for sensors or dipoles. The construction of the particular grid is different every time and is complex. Therefore, the construction of the grid is separated from its representation. The abstract base class `anAbstractGrid-Generator_c` provides the interface of the builder classes, namely a method to execute the building process and a method to retrieve the product. The derived classes implement different construction processes. For our purposes, `anBEMGridGeneratorASA_c` is relevant, because it creates a triangulated BEM model surface by reading positions and triangles from an input file. This surface representation is neccessary for the continuous search volume constraint. `anGridGeneratorVolumeInsideBoundary_c` can be reused to obtain a regular volume grid from a continuous search volume definition, which is itself a triangulated surface. This provides a simple method to derive the discrete search volume for our discrete constraints.

The class `Grid_c` is our representation of the continuous and discrete search volume and is thus where the fast operations defined in Section 3.3.2 belong. Algorithm 5 is implemented in `findClosestNode()`, Algorithm 6 is implemented in `sortedIndexOfFirstNodeWithAtLeastX()` and Algorithm 7 is implemented in `findNodesWithinDistanceRange()`. The sorted list is implemented by an STL vector or C++ structs containing node, sort value pairs.

4.7 Software Process Model

An incremental development process [7, 91] was applied for the new components and the modifications of existing structures outlined in this chapter. The requirements in terms of constraint algorithms, compatibilities between optimizers, constraints and goal functions, types of optimizers and goal functions, types of search volumes, etc. were defined fairly completely in the beginning. The constraint hierarchy design was an early strategic decision [7], which provided the skeleton for the incremental development.

Incrementally classes were added. The increments were a new optimizer class, a new constraint and so on. Because the execution of an optimization requires an optimizer, a goal function and an constraint to cooperate, the finished increments were run with placeholder components. For example, Rastrigin's function was used as placeholder goal function and existing analytical search volumes, such as a sphere, were used as placeholder for the search volume defined by a triangulated search volume. Successively, placeholders were replaced by finished components which could be described as a bootstrap process. In this way the new component and its interaction with the system could be tested after each increment and experiences could be fed into the design early. High priority components, such as `anConstraintMinDistCont_c` which is the most critical part of the constraints, could be delivered early and thus received more testing.

The individual implementation, performance optimization and testing stages are not documented separately, because the focus of this study is not solely the software development, but also on the application and verification of the algorithms in the biomedical setting. These stages did, however, pose a number of concrete implementation issues, such as the limited numeric precision in continuous calculations. For example, if we compute a vector between two positions which are very close to each other in Algorithm 9, the resulting movement vector may have a strong deviation in spatial direction due to the limited precision. The integration into SimBio required support for marking specific parameter as unused and only optimizing the rest while still keeping the unused parameter for the goal function evaluation, etc. Please refer to the commented C++ source code for full implementation details.

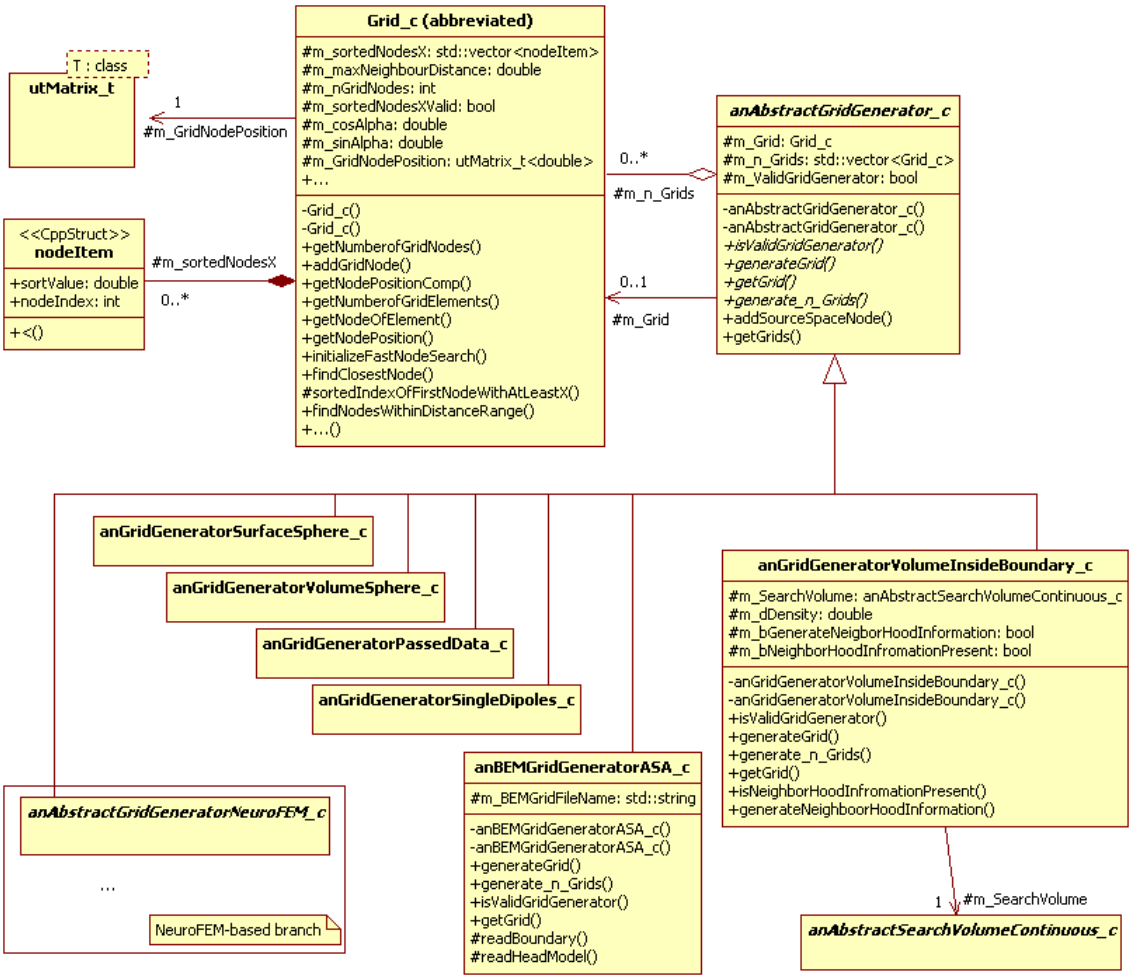

Figure 25: Grid class and the grid generator hierarchy (omitted attributes and operations of pre-existing, unrelated grid generators)

5 Application to Magnetocardiography

5.1 Previous Sensor Optimization Approaches in Magnetocardiography

Efforts to determine the number of sensors necessary to capture the electric field of the myocardium and their positioning have been made already in the 1970's. Lux et al. [61] collected body surface potential maps (BSPM) of 132 test persons and evaluated subsets of these leads in terms of the root mean square error between limited leads and the total lead map, the mean correlation coefficient and the error to signal power ratio [61]. They concluded that 30-35 selected leads yield low enough error values. However, they could not find a unique lead arrangement with this property, but set of different ones. A further principle component analysis of BSPMs [60] uncovered 12 principle components of the electric field captured by BSPMs. Barr et al. [4] found a 24 out of 150 leads to be sufficient. A recent investigation [25] also concludes with 32 leads being sufficient.

Magnetic sensor optimization in cardiology has come into focus more recently. Nalbach and Dössel [67] used the condition number of the leadfield matrix as optimization criterion for both ECG and MCG. In their study, a FE model of a torso with lungs and heart was generated. 990 radial magnetometers were arranged around the torso in a tube shape. The combinatorial optimization consisted simply in successively eliminating sensors that do not increase the CN. They determined setups of 32, 64 and 99 sensors and compared them to commercial sensor setups. They found that the optimization of magnetic sensors and electric ones individually can increase the reconstruction robustness. As a compromise to the cryostat technology they suggest to place a second patch of sensors in a second cryostat at the back of the torso.

Kim et al. [50] reviewed the sampling theory to determine a setup of their tangential magnetometers, which fulfills the Nyquist- theorem. Their finding is that for tangential sensors 99% of the spatial spectral power at distance z from the source in Z-direction is below $1/2z$. They thus conclude that a sampling width of z is sufficient. They approximate z with $4\ cm$, since the heart lies at least $3\ cm$ from the torso surface and the cryostat requires another $1\ cm$. The resulting setup is a regular grid with $4\ cm$ width.

A number of studies exist, which propose criteria for assessing given setups. These criteria may be used as goal functions in the future. For example, Di Rienzo et al. [17, 3] investigated the gain in reconstruction robustness by considering all three vectorial components of the field in each sensor position instead of just one. As evaluation criteria they used the RMS error, the mean correlation coefficient and lower error bounds [80].

A more theoretic optimization of sensor positions is proposed by Rouve et al. [85]. They assume a multipole source model and optimize the sensor positions by minimizing the condition number of the matrix of spherical harmonics influences on the sensors. The optimization approach is a genetic algorithm. The sensor search volume is a sphere around the source and the number of sensors used in their theoretic example is 5 (10 parameters).

5.2 Experimental Setup

5.2.1 Clinical Data

The clinical application of magnetocardiography would be the diagnosis of various functional pathologies of the heart. In order to optimize a sensor setup for this task, MCG recordings of patients with these pathologies should be used to guide the optimization. The acquisition of a representative set of patient recordings and the derivation of a complex cardiac field model is beyond the scope of this study. We would rather like to present a proof of concept, using one recording of a test person with a non-trivial cardiac excitation pattern.

The test person is male and was 72 years old at the day of the recording. He had coronary artery disease with a 50% stenosis of the R. interventricularis anterior and a 70% stenosis of the R. circumflexus. The test person has a sinus rhythm, but the excitation pattern in the ECG and MCG shows a prolonged (>100 ms) QRS complex with two peaks indicating some kind of right ventricular excitation delay or a possible incomplete right bundle branch block (Figure 26, Figure 27). This causes the excitation front to move across the myocardium non-uniformly.

An MCG at rest of 5 minutes duration was made with the Argos200 System introduced earlier. Subsequently, an MRI (T1-weighted) of the complete torso was made with 1 mm resolution. In both procedures the patient positioning unit (PPU) was placed on the test person. The PPU consist of three coils arranged in a right-angle triangle on a non-magnetic base. These landmarks can be identified in the MCG and the MRI and thus both datasets can be mapped geometrically.

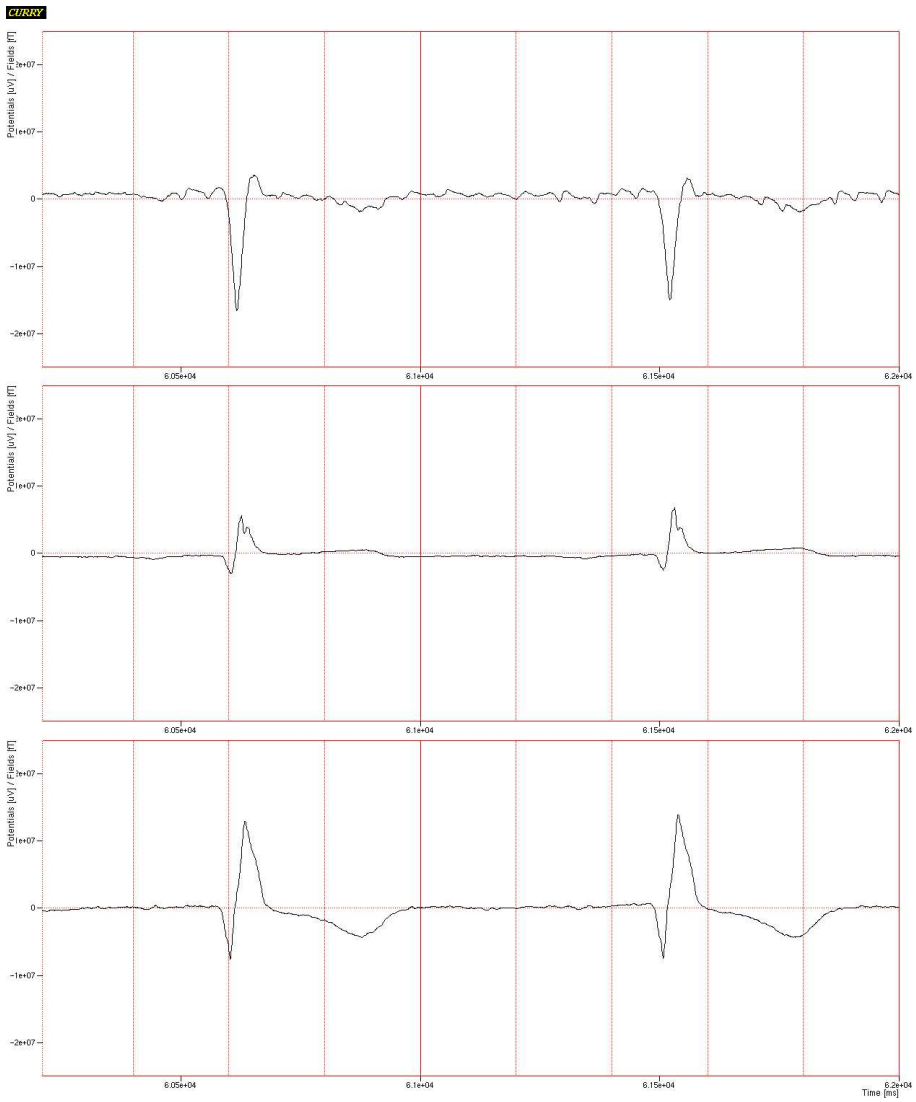

Figure 26: Electric Frank leads in orientations X (top), Y (middle) and Z (bottom)

5.2.2 Data Preprocessing

Prior to data analysis the three vectorial components measured per triplet were transformed into the components B_x, B_y and B_z in the center point of the cube formed by the rectangular sensors. The sensor-plane is the xy-plane and the z-dimension is the normal pointing away from the test person. The heart beat times were determined by template matching and averaged channelwise.

5.2.3 Volume Conductor Model

From the T1-weighted MRI (1 mm resolution) of the test person the torso surface, both lungs and the blood volume inside the heart and the aortic arch were segmented using Curry Version 4.6 (NeuroScan Compumedics, El Paso, USA). The surfaces of these four compartments we triangulated with a triangle side length of 6 mm for the blood mass (1413 nodes), 10 mm for both lungs (889 + 843 nodes) and 14 mm for the torso (2600 nodes). This resolution has been shown to be a good compromise between computational speed and reconstruction accuracy [40]. We assumed a homogeneous conductivity of 0.2 S/m for the torso including the heart muscle, 0.04 S/m for both lungs and 0.6 S/m for the blood mass [40].

To be able to compare our results to previous investigations [17], we also include this preexisting volume conductor, source model and sensor search space (Figure 29) in our study. This model is used for initial theoretic inquiries, while the new recording and volume conductor is used for realistic optimization.

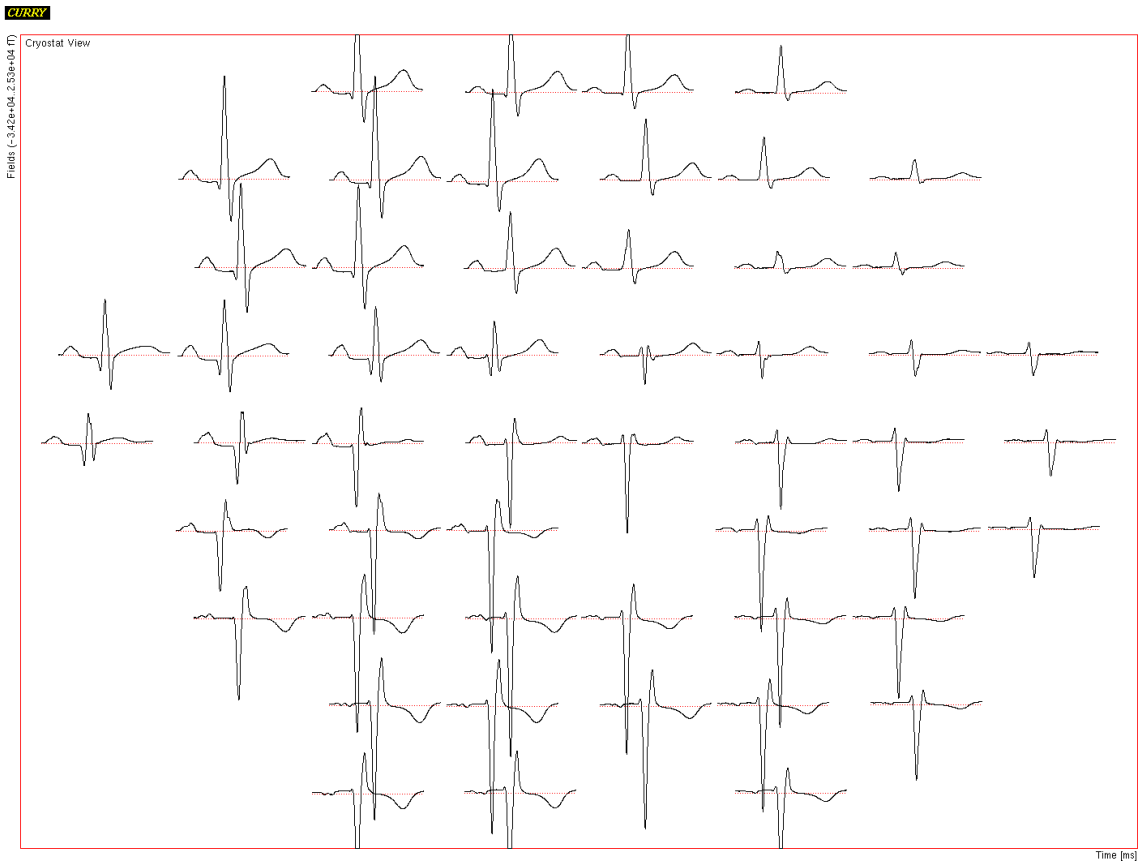

Figure 27: Position Plot of the B_z component of the average heart beat of the test person

Even though both volume conductor models consist of five compartments, we only use the torso and both lungs, because currently SimBio only supports three compartments. The torso and lung surfaces have the strongest impact due to the large differences in conductivity. The ventricles are useful for determining the dipole positions of the source model.

5.2.4 Source Model of the Cardiac Field

The propagation of the excitation in the myocardium is complex. Durrer et al. [18] in 1970 produced a detailed isochronic representation. (Figure 30). In particular, we can see that the action potential front is not solely moving tangentially to the myocardium but also radially. Which may give rise to radially arranged dipoles, such as in Figure 29.

Because the magnetic field of the heart is rather a superposition of distributed dipole moments, a distributed source model is a sensible choice. This is particularly true, if we are trying to accommodate irregular excitation patterns, e.g. caused by weakly conducting infarction scars. A number of dipoles is spread evenly across the myocardium. Because the left ventricle is the dominating field source, often only the left ventricle is used [17]. Each dipole represents the dipole moments in its vicinity. Thus, the distributed field is discretized.

Apart from models with large numbers of dipoles, several different approaches of placing the dipoles exist. Purcel et al. [79] suggested a 7 dipole model, where one dipole is placed at (1) the sinus node and left atrium, (2) the His-bundle, (3) the septum, (4) the mid-posterior left ventricle, (5) the mid-anterior right ventricle, (6) the right atrium and (7) the apex. This model was extended by Hren et al. [41] with 10 more dipoles arranged regularly around the atrioventricular ring [26]. The purpose was to model the different types of the Wolf-Parkinson-White syndrome. More recently, Di Rienzo et al. [17] applied a 13 dipole model (Figure 29) and a 39 dipole model. Twelve dipole positions are arranged regularly in three rings around the left ventricle and one is placed at the apex. To obtain the 39 dipoles, three orthogonal dipoles are placed at each of the positions.

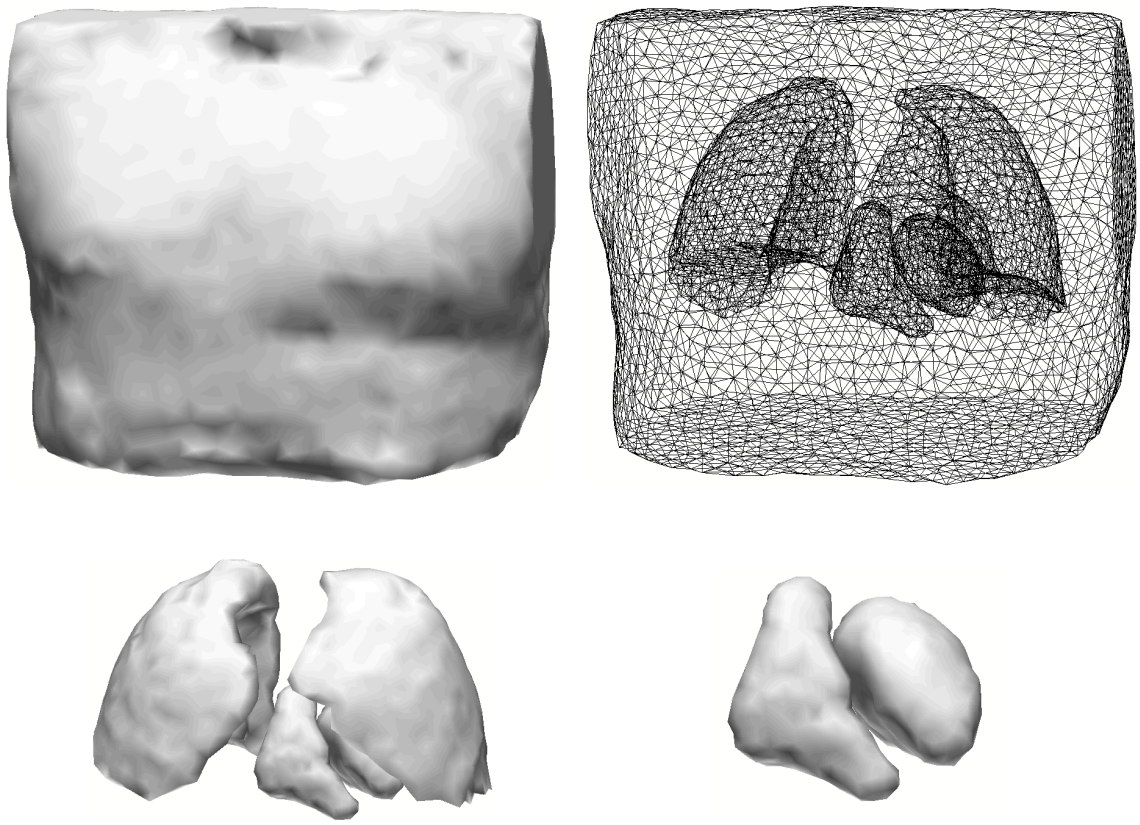

Figure 28: Compartments of the boundary element model from outside to inside: torso surface (top, left), boundary element model (top, right), lungs and blood masses (bottom, left) and enlarged right and left ventricular blood mass (bottom, right)

Figure 29: Sensor area in front of boundary element model of torso, lungs and ventricular blood mass. The source model consists of 13 dipoles [17].

In this study we adopt the 13 dipole model. We set the dipole positions according to the same scheme. However, we do not assume radial dipoles. Instead we perform a rotating dipole fit (minimum norm with L-curve regularization and sLORETA [74, 72] noise normalization using CURRY 4.6) over the averaged heart beat (PQRST) using all three components ($B_x,\ B_y,\ B_z$) of the vectorial recording. The resulting model explained the variance of the recording to more than 99.5%. The resulting dipole orientations (Figure 31) to a large degree tangential, but also have radial components. Figure 32 shows the dipole amplitudes at the

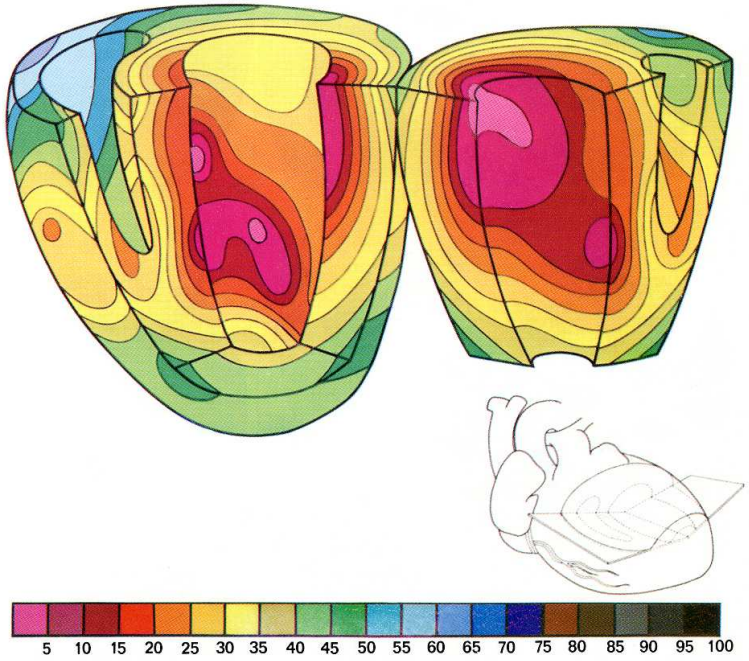

Figure 30: Three-dimensional isochronic ($5\ ms$) representation of the activation of a human heart [18, p. 903]

first R-peak. The strongest dipole amplitudes are in the septal and posterior region.

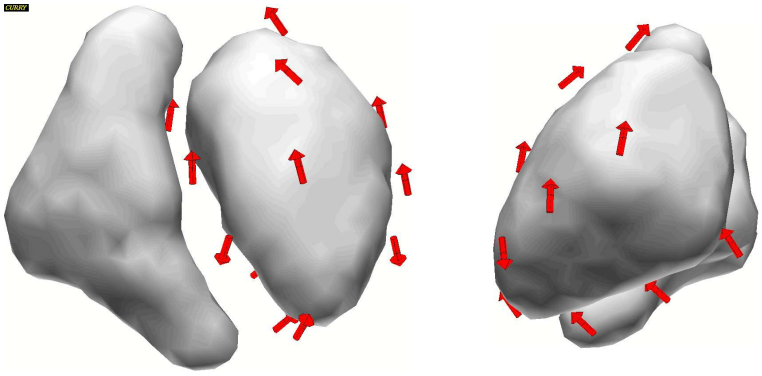

Figure 31: Source model consisting of 13 dipoles (uniform length shown) around the left ventricle (orientations fitted with minimum norm, L-curve regularization and sLORETA noise normalization)

In subsequent sections, we will denote the combination of the BE model and radial dipoles from the literature [17] with "model A" and the combination of the newly developed BE model with the fitted dipoles with "model B".

5.2.5 Sensor Type

For our optimizations we assume a sensor with a spherical outer bound of diameter $1\ cm$. This is approximately the size of current integrated SQUIDs chips (e.g. $8x8\ mm$ in the Argos200 system) and well above the size that current developments of optical magnetometers are targeted at [10]. Research-type SQUID systems can reach a coil-diameter of less than $1\ mm$ [53], which allows sensor distances of a few millimeters.

The sensor is assumed to pick up the field component in the normal direction to the coil area. For the field simulations in SimBio we use a single winding and a coil area of $(0.5\ cm)^2 * \pi = 0.7854\ cm^2$.

For the initial investigations with the pre-existing source model and BE model, we used a sensor size of $2\ cm$. This sensor size accommodates existing optical magnetometers [6].

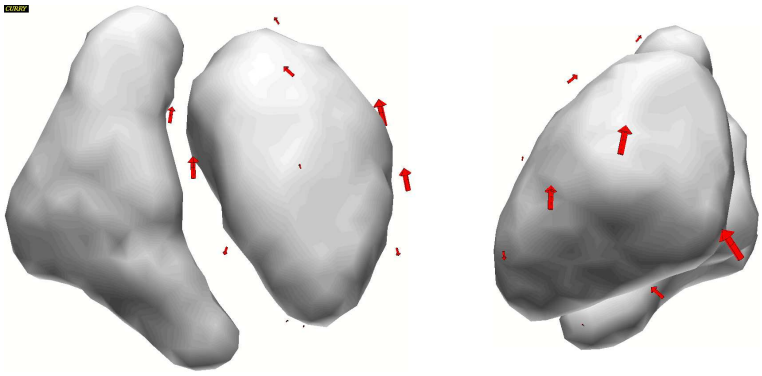

Figure 32: Current density distribution at the first R-peak using the 13 dipole model of Figure 31

5.2.6 Search Volumes

In this study three search volumes are tested. For the initial investigations with the radial dipole model we use the same square sensor area in front of the torso as in [17] (Figure 29). This square is discretized to 85x85 regular positions ($2.5\ mm$ distance) and and the directional space is discretized with a width of 30°, resulting in 62 orientations per position.

Secondly, we create a search volume within the pick-up sensor area of the Argos200 system during the recording (Figure 33). This search volume fills the bottom of the cryostat with a radius of $12.5\ cm$. The area is discretized to a regular grid with $2\ mm$ sampling width. The directional space is discretized with 36° (42 orientations per position). The full leadfield matrix thus has about half a million lines. This search volume enables us to make a cryostat bound optimization. The resulting sensor setup can thus be realized with SQUIDS.

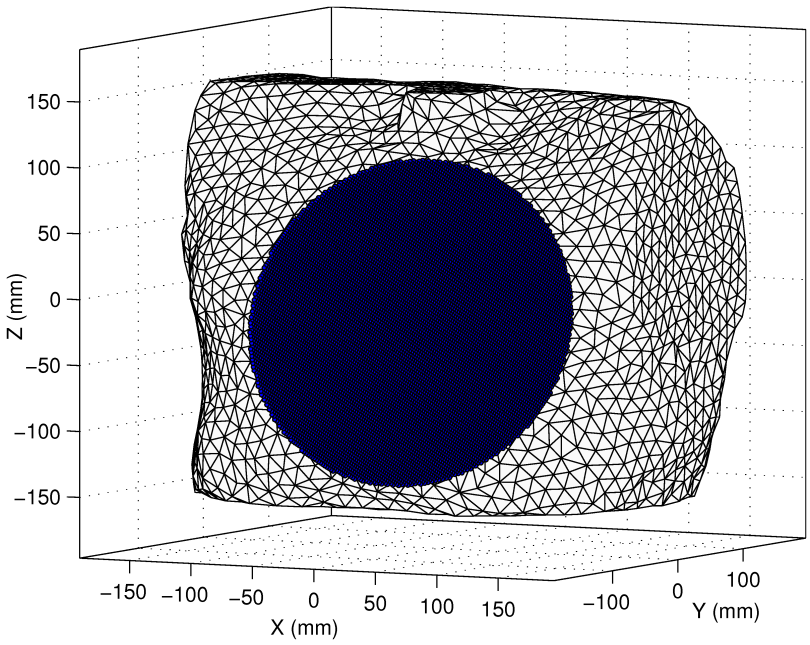

Figure 33: Circular sensor search area (12281 regular positions) with diameter $25\ cm$ positioned over the heart

The third search volume is derived from the torso surface, looking like a vest (Figure 34). Positions are the nodes of the triangulation ($4\ mm$ side length) of the dilated ($14\ mm$ = triangle side length of the torso triangulation) torso surface. The directional space is discretized with 30° width (62 orientations per position). The full leadfield matrix in this case has roughly 1.2 million lines. This search volume enables us to show the gain by removing the cryostat limitation and allowing free placement of sensors.

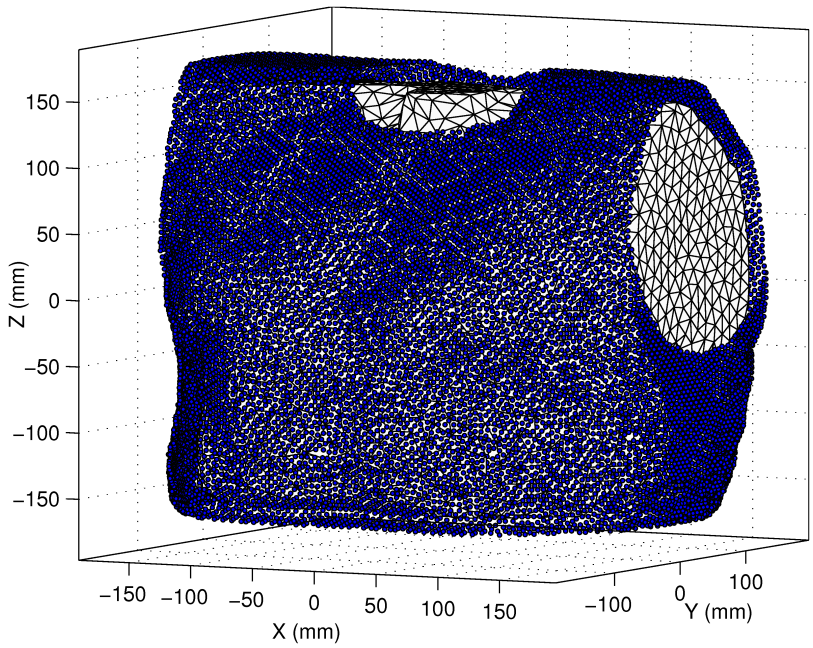

Figure 34: Sensor search volume (19759 positions) around the torso surface omitting connections to arms, neck and lower body. Positions are the nodes of a triangulation ($4\ mm$ side length) of the dilated ($14\ mm$) torso surface.

5.2.7 PSO and Constraint Settings

For the simulations described in subsequent sections the PSO settings in Table 2 and the constraint settings in Table 3 have been used unless stated otherwise. The number of sensors is denoted with N. Each sensor has five parameters, its three cartesian coordinates and two spherical angles for the orientation. So the dimension of the optimization problem is $5 \cdot N$. A nearly fully informed swarm is used. The number of iterations, 3000 in this case, is set via the number of goal function evaluations. Since in each iteration each particle consumes one evaluation, we need to multiply the number of iterations with the number of particles to yield the number of evaluations. We limit the resources for one optimization to 1 million goal function evaluations. Thus, the optimization is repeated as many times as the resources allow.

Symbol	Meaning	Value
S	swarm size	$10 + 2 \cdot \sqrt{5 \cdot N}$
K	number of informants	$\lfloor S \cdot 0.95 \rfloor$
w	first cognitive coefficient	1/3
c	second cognitive coefficient	2
λ	velocity adjustment factor	0
E	maximum number of evaluations per execution	$S \cdot 3000$
R	number of repetitions	$\lfloor 10^6 / E \rfloor$

Table 2: PSO parameter settings for sensor optimization in Algorithm 2

5.3 Optimization within a Square Sensor Plane

The first set of optimizations was run with model A, the sensor area with square border (Figure 29) from the literature [17] and a minimum distance of 2 cm. We used $S = 2\sqrt{5N}$ particles and 1500 iterations, λ was set to 0.5. The optimization was run for a range of numbers of sensors (Figure 35).

To validate the results of the PSO optimization, we applied a discrete tabu search (TS) optimizer (described in [56, 55]) to an adapted version of the same problem (green line in Figure 35). We discretized the same sensor area with a grid width of 2 cm (11x11) to implicitely enforce the minimum distance.

Symbol	Meaning	Value
m	minimum distance	1 cm
ξ	influence of max. violation	0.05 (5%)
$\hat{\alpha}$	max. random angular deviation	$0.02 \cdot 360° = 7.2°$
$\hat{i}$	max. number of iterations	50
t	relative tolerance to mean min. distance violation	0.02 (2%)

Table 3: Constraint parameter settings for sensor optimization in Algorithm 12

The directional space was discretized with a width of 45° (26 orientations). The TS optimizer then does a combinatorial search on the given set of sensor configurations.

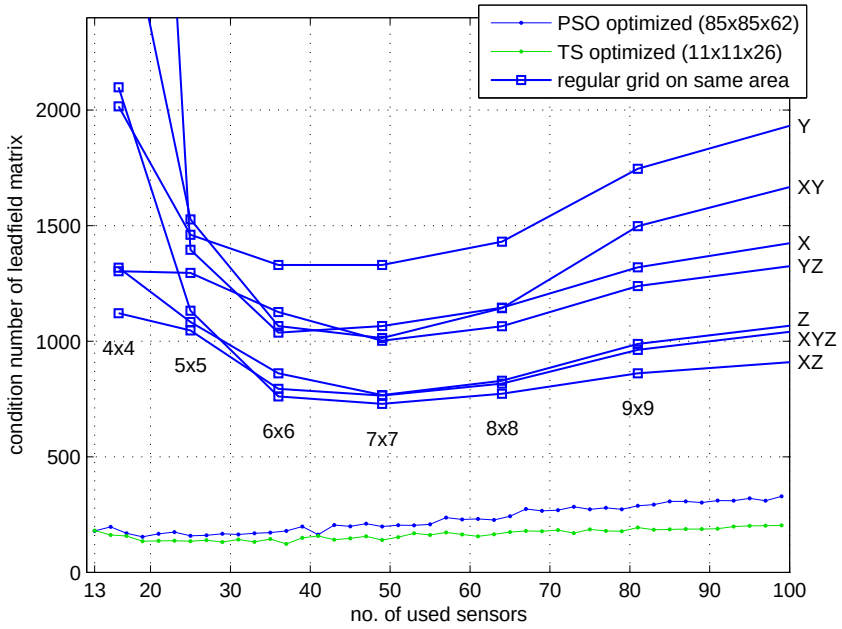

Figure 35: Condition Numbers of optimized and regular grid setups within a square area in front of model A (X = all sensors aligned with positive X-axis, etc.) for a range of numbers of sensors

Regular and optimized setups The CN of the optimized setups is significantly lower than the CNs of regular grids on the same surface with the same number of aligned sensors. In these noise-free simulations, the optimal number of sensors is around 20 - 30 for both TS and PSO. The optimal number of sensors for the regular grids is between 36 (6x6) and 49 (7x7). The strongest gain from regular to optimized setups is achieved for low numbers of sensors. This is expected because the lower the number of sensors available the higher the information gain when optimal sensor positions are used [56].

Between the sensor orientations of the regular grids are significant differences. As expected the Z-direction sensors have a better CN than X- and Y-direction. When Z-direction sensors are combined with X or X and Y sensors, even lower CNs are obtained. This conforms to findings in the literature [3].

PSO versus TS results TS and PSO produce very similar CNs for less than 45 sensors. For denser sensor setups, TS performs a bit better. The higher CNs for PSO at higher numbers of sensors can be explained by the difficulties PSO encounters in moving sensors in a densely populated search volume. Moreover, slight differences between the two optimization approaches can be explained by the fact that the direction discretization was different for PSO (30°) and TS (45°). Thus, the in reality continuous optimal sensor directions could be better explained by one or the other discretization.

Optimized Sensor Positioning The sensors of optimized setups tend to be placed in areas of strong magnetic field gradient. TS and PSO optimized setups show similar positions and orientations of sensors (Figure 36) [56]. Many of the optimal sensor positions are close to the boundary of the search volume. This indicates that the search volume might not have been large enough.

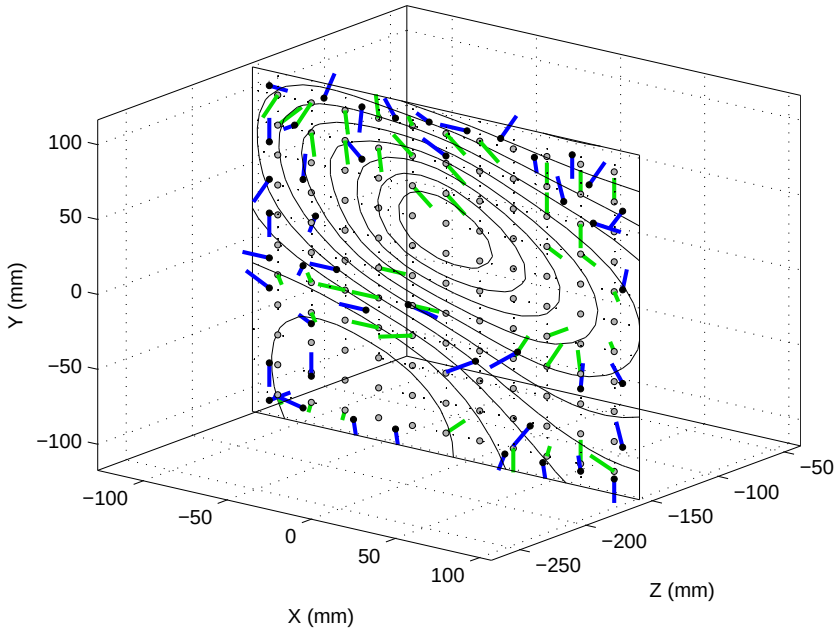

Figure 36: PSO (blue bars) optimized setup of 45 sensors in square sensor area and magnetic field map of the X component (thin solid lines). The sensor grid is positioned centrally in front of the torso (Figure 29).

5.4 Optimization within a Cryostat-bound Sensor Plane

The second set of optimizations was run with model A and the circular search area (Figure 33) modeling the bottom of the cryostat of the Argos200 system used for the original recording. The optimization result would still be compatible with the current SQUID technology.

Condition Number Magnitude The absolute CNs of the optimized setups (green in Figure 37) with less than 45 sensors have a magnitude ($x \cdot 10^2$ instead of $x \cdot 10^3$ for regular setups) similar to the CNs of optimized setups within the square search area in front of model B (blue). This indicates that the results obtained in both simulations pertain to the magnetocardiographic problem, rather than the individual volume conductor model and source model.

The CNs of the circular area are however consistently a bit lower (green in Figure 37) than the CNs of the square area. This is expected since the minimum distance is only $1\ cm$ instead of $2\ cm$. In areas of strong gradient, which offer most information, the field can be sampled with half the sampling width of the previous simulation. Additionally, the sensor area is the same as that of the Argos200 system during the recording. The sensor area is arranged as close to the heart as possible and parallel to the body surface.

The observation that the CN for the second search volume increases much less than that of the first one, can also be attributed to the smaller minimum distance. For the same number of sensors there are much less minimum distance violations. So the sensors can move relatively freely through the search volume. This also means that if there are more sensors that necesssary to capture the field, the redundant ones can move to positions that have a minimum impact on the goal function value.

Required Number of Sensors Considering that the cardiac field only has a certain degree of complexity, there must be an according number of sensors which is sufficient to capture it. The optimized setup of 35

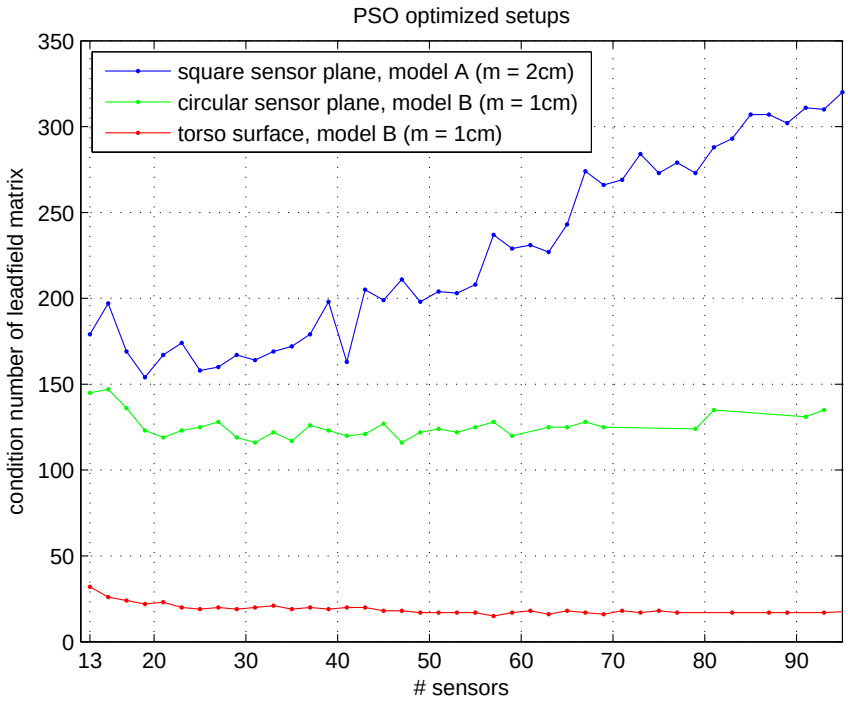

Figure 37: Condition numbers of optimized setups of all three search volumes for a range of numbers of sensors

sensors in Figure 38 for example shows a fairly distributed pattern with several groupings of sensors at the edge of the search space. This may indicate that there are better positions outside the search volume (the cryostat) or that these sensors are redundant. In terms of the CN (green in Figure 37), the best setup would be achieved with about 21 sensors. This setup (top right of Figure 39) has a distributed pattern with no grouping at the edge. Optimized setups with less, e.g. 13 (top left of Figure 39), sensors have an increased CN because they can not capture the field robustly. Above 21 sensors the CN only increases marginally, but the individual setups (Figure 39) are characterized by groupings of sensor on the edge of the search volume.

5.5 Optimization on the Torso Surface

For the third set of simulations we allowed the sensors to move freely on the torso surface (Figure 34). The result of this optimization can be realized with optical magnetometers much more easily than with SQUIDs. The number of repeated executions was set to 10 to reduce the computation time.

Lifting the cryostat-restriction results in a second remarkable drop of the condition number by another order or magnitude ($x \cdot 10^1$) to values around 17 (Figure 37). The CN decreases stronger from 13 to about 21. Above 21 there are only minor improvements in terms of CN by adding more sensors. The CN reaches its lowest and stable values of around 17 for more than 60 sensors. The minimal setup with 13 sensors (top in Figure 40) shows several lateral sensors, two superior ones, few left posterior ones and several inferior ones. Interestingly, these sensors are for the most part outside the flat sensor area of a cryostat. The essentially robust setup with 21 sensors (center in Figure 40) has additional posterior and inferior sensors, but a similar distribution. The 31 sensor setup adds right posterior sensors covering the whole back and several right inferior sensors. In the 49 sensor setup (top in Figure 41) we can observe the formation of a band of sensors remarkably similar to V1-V9 (Figure 4). Additionally a group of sensors accumulates at the inferior back which continues for higher number sensors. For 63 and 93 sensors (Figure 41) the torso front and back are samples more densely, while placing more sensors in the left lateral band, similar to V1-V9.

In contrast to the flat sensor area, the number of sensors that are useful for sampling the field is higher. Thus, redundant sensors only accumulate for higher number of sensors.

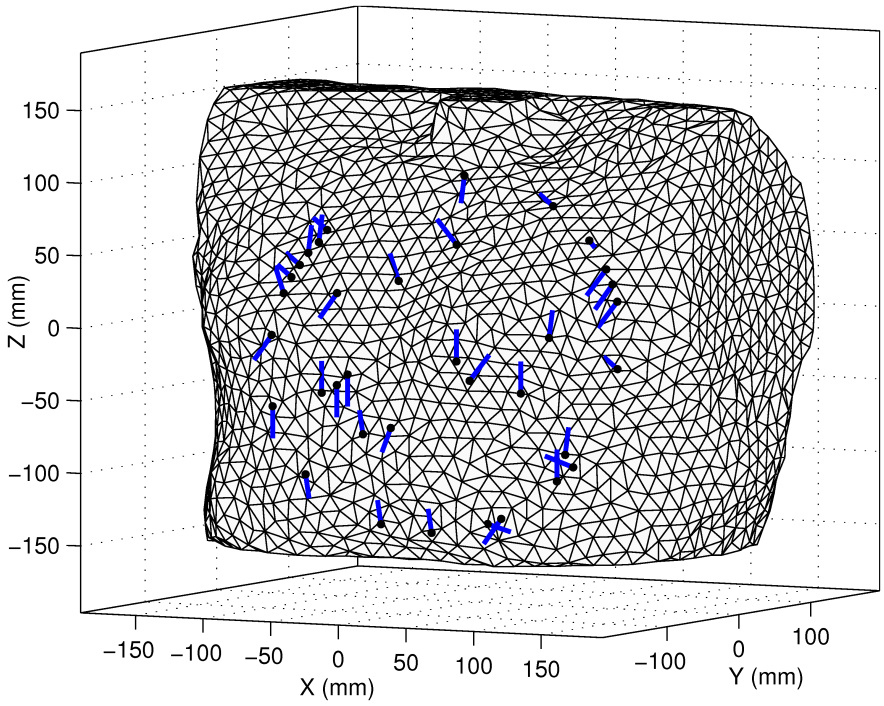

Figure 38: PSO optimized set of 35 sensors (out of 12281 regular positions) within a circular sensor area (diameter $25\ cm$) positioned over the heart (area identical to Argos200 sensor plane during MCG recording)

5.6 Comparison to Previous Findings

Number of Sensors The number of sensors of 20-30 found in this study are comparable to the findings of Lux et al. [61] (30-35 sensors), Barr et al. [4] (24 sensors) and Finlay et al. [25] (32 sensors). This is particularly interesting, since our evaluation criterion is the condition number of the leadfield matrix. The leadfield matrix expresses whether a small change in dipole amplitude also causes a small change is sensor amplitudes or not, disregarding the actual dipoles amplitudes. We are thus optimizing the robustness of the reconstruction, whereas the mentioned studies compared concrete field maps at particular time steps of the heart beat, which implicitely considers the dipole amplitudes.

Position and Orientation of Sensors The positions of the sensors show similarities to the Wilson leads V1-V9 through a band of sensors extending from the left postero-lateral area to the superior anterior torso surface. This also implies that the flat cryostat-bound sensor arrays may not sample the field at the proper positions. Additionally, the torso surface is sampled more densely, the more sensors are used. In agreement to previous findings [67] the left posterior torso surface is an important location for sensor placement as well.

The orientations of the sensors vary within the positions. When several sensors are placed close to each other, they tend to point in rather orthogonal directions. This means that particular and multiple vectorial components of the field are relevant to the robustness of the reconstruction. This is in agreement to previous findings [17, 3], which advocate for measurement of all three vectorial components.

Sensor Search Space This study extends existing sensor search spaces [67] in two ways. Firstly, the complete detailed torso surface is considered. A fine grid of possible sensor positions is used. Secondly, the orientation of the sensor is parameterized and optimized concurrently with the positions. This is important since existing findings [17, 3] indicate that vectorial measurements contain more information.

Optimization Technique The finely grained, and thus flexible, high-dimensional search space suggests the use of a gradient-based method that moves the sensors through the search space quasi-continuously, instead of performing a combinatorial search which disregards spatial closeness relations and local goal

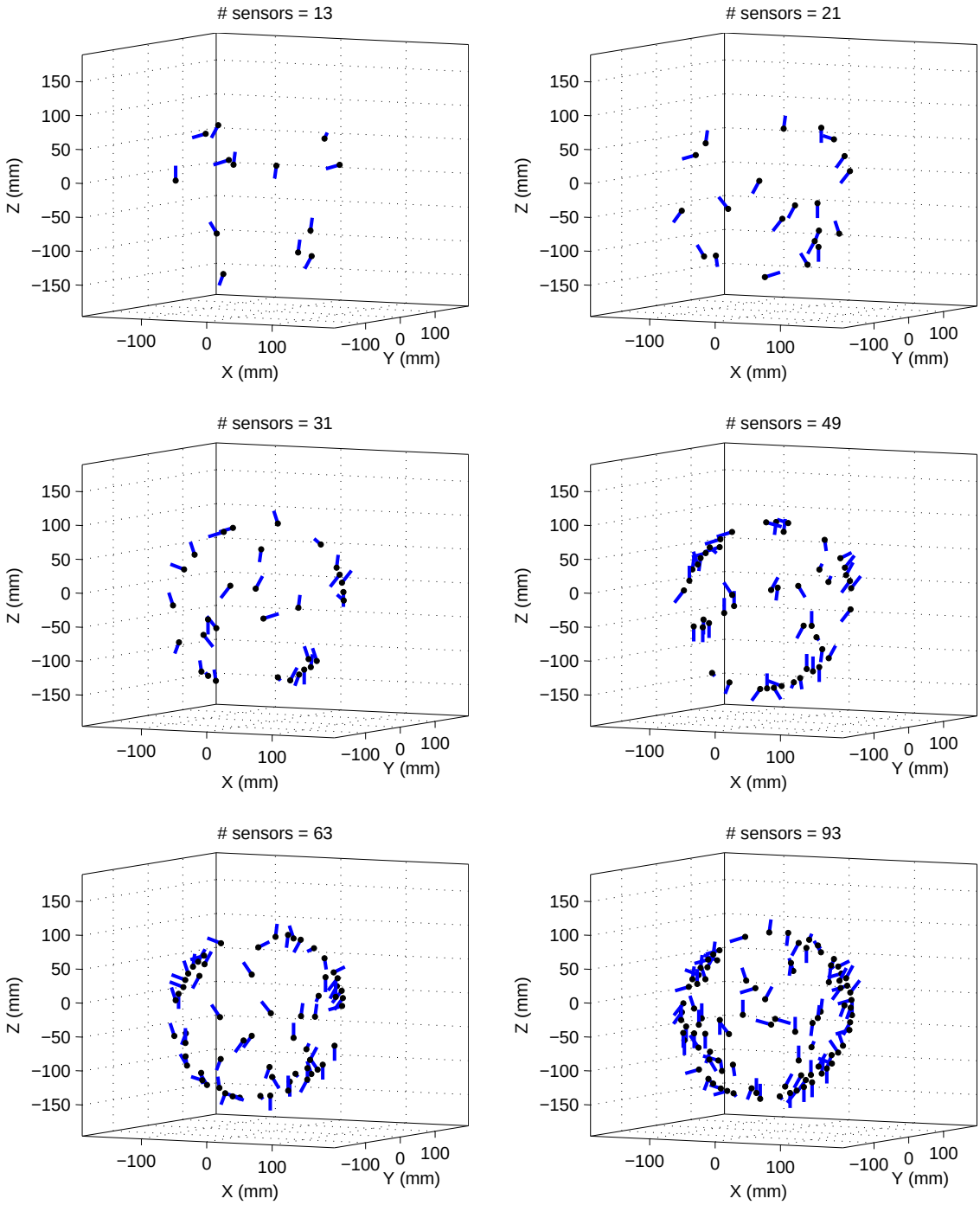

Figure 39: PSO optimized setups of varying # sensors within circular sensor area seen from the same view angle as in Figure 38

function gradients. The constraint PSO implementation presented in this study is such a method. Genetic algorithms [85] and tabu search [56, 31] are alternative approaches, which should be investigated further and compared.

Sampling Density The sampling density in the optimized setups varies at different locations, according to the influence of the dipoles. For the setups with 21 sensors, the distances to the respective closest neighbour have two groups (Figure 42). The dominating distances are around $25\ mm$ ($30\ mm$ for torso surface), these

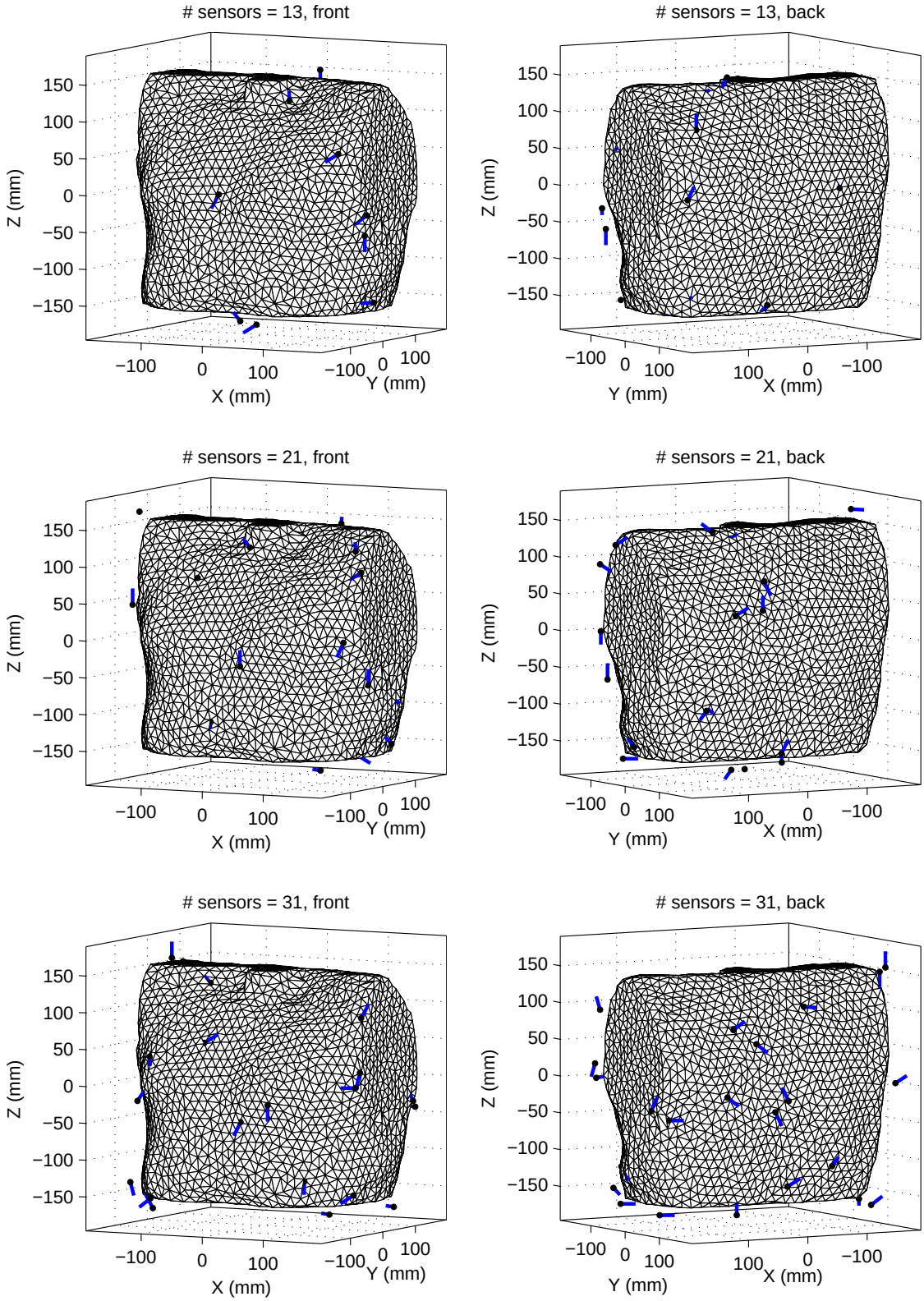

Figure 40: PSO optimized setups of 13, 21 and 31 sensors on torso surface

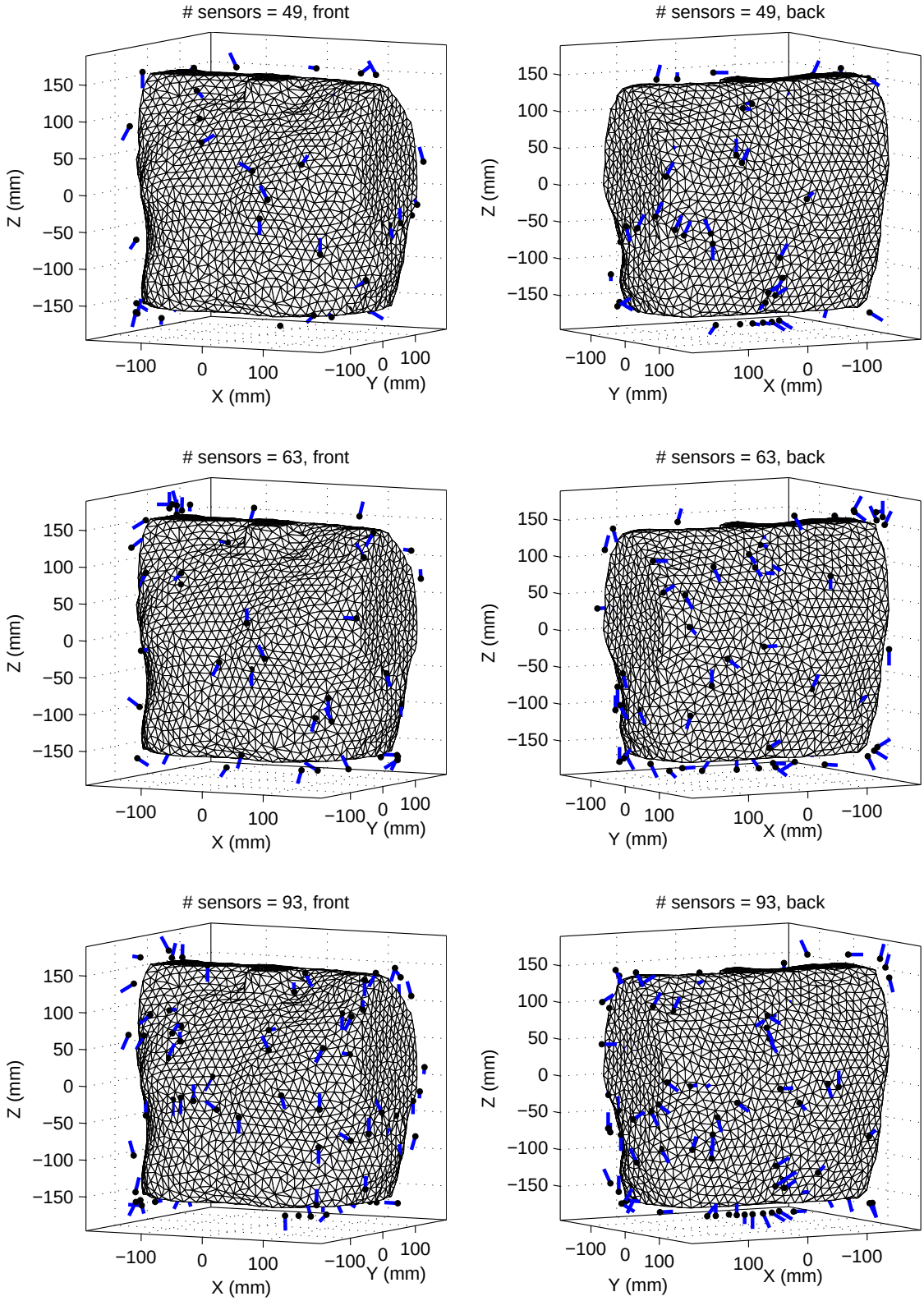

Figure 41: PSO optimized setups of 49, 63 and 93 sensors on torso surface

are the closely sampled areas. The second group are larger distances around $60\ mm$ ($100\ mm$ for torso surface). These represent the more coarsely samples areas. This observation implies that a minimum sensor distance of $20\ mm$, such as the one we assumed when optimizing on model A, would disturb the optimized setup. A minimum distance of $10\ mm$ has a much weaker impact, since the sensor do not tend to be placed that close. The first group of sensor distances can be interpreted in two ways. These shorter distances may describe groups of sensors with different orientations, indicating field information in multiple vectorial components at this location. A second reason for the existence of the first group may be the necessary sampling width. A sampling width of $25 - 30\ cm$ fits to the sampling width of $40\ mm$ for purely tangential setups suggested by Kim et al. [50]. The sensors can be expected to lie closer than $40\ mm$ to each other, because they are picking up separate vectorial components.

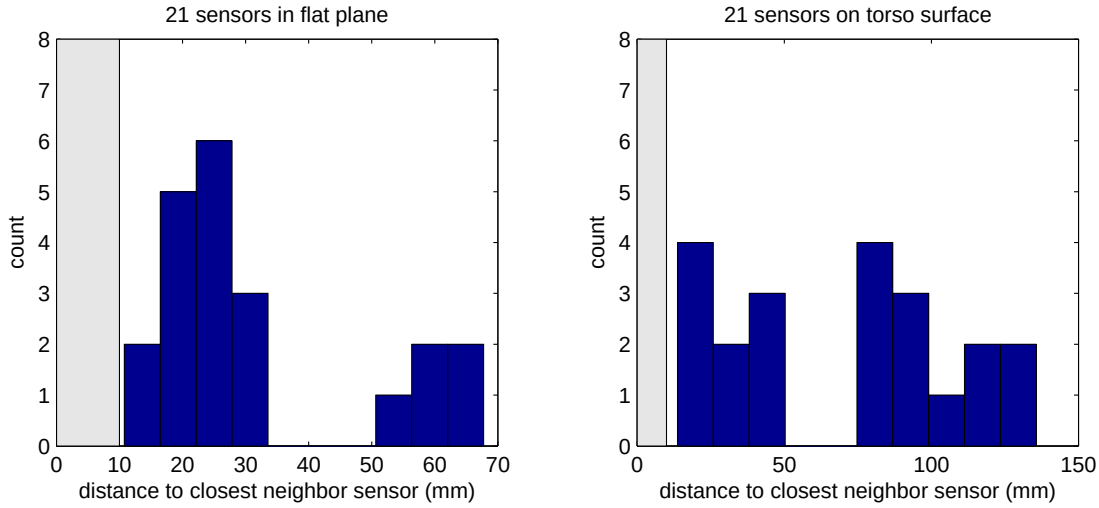

Figure 42: Histogram of the distances to the closest neighboring sensor in the optimized 21 sensor flat setup (top right in Figure 39) and in the 21 sensor torso surface setup (center in Figure 40). Minimum distance of $10\ mm$ is marked in gray.

5.7 Performance of PSO Optimizer and Constraints

The results show that the PSO algorithm is effective for this particular optimization problem. The modifications made due to the constraints showed to be non-critical when the number of sensors is significantly smaller ($< 40\%$) than the number of sensors, the search space can fit. For the circular sensor area and the torso for example we used a velocity adjustment of $\lambda = 0$. The influence of the maximum violation ξ was set to a very small amount (5%) as well. For densely filled sensor search volumes, the constraint fix increasingly interacts with the optimization. This is of cause neccessary to enforce the constraint during the optimization process.

The minimum distance fix was frequently required. Still the tolerance $t = 2\%$ of minimum distance violations was never exceeded, which means that every minimum distance fix was successful. The random angular deviation during the minimum distance fix however did reduce the number of iterations required to fix the constraint. The search volume constraint was required before each evaluation and within the minimum distance fix. Therefore, the performance optimized grid node retrieval algorithms were essential help to make the optimization feasible. Thus, the concept of restoring small violations in each iteration to prevent them from building up was effective.

6 Conclusion

The primary objective of this study was to develop a generic optimization framework for the arrangement of sensors for biomagnetic investigations such as in cardiomagnetism. This goal was achieved by encapsulating the components (optimizer, goal function, constraint, search volume, BEM, etc.) required to perform the optimization and defining the interfaces between these components formally in abstract base classes of the object-oriented design. The concrete components, such as the PSO optimizer and the CN goal function only implement these interfaces and are thus exchangable and compatible to all future components. These interfaces are integrated into the biomedical simulation environment SimBio to maximize reuse.

The second objective of producing example components has been reached by implementing and customizing a particle swarm optimizer, a goal function computing the condition number and a generic hierarchy of basic constraints that can be and have been combined to more complex constraints. The customization of the PSO algorithm to quasi-continuous problems as well as the algorithmic and object-oriented design of the composite constraint handling are original contributions to the field of constraint optimization.

The effectiveness of the optimization strategy, resembled by the example components, has been demonstrated on a realistic magnetocardiographic sensor arrangement problem. The capabilities of the framework in terms of optimization flexibility and robustness and customizability of the search volume significantly extend previous studies in the literature. The condition number dropped by one order of magnitude from regular flat sensor setups ($x \cdot 10^3$) to optimized flat setups ($x \cdot 10^2$) and by another order of magnitude by considering the torso surface ($x \cdot 10^1$). Future work should reconfirm this finding in a larger set of patients.

For the design of biomagnetic measurement systems that means the future availability of small, non-cryostat-bound optical magnetometers will allow a significant increase of the robustness of field reconstruction. At the same time a reduction of the number of sensors is possible without sacrificing this robustness.

Building on top of the generic framework a range of studies can be conducted. Goal functions can be compared to each other, as well as optimization and constraint handling techniques and application specific search spaces. The optimization interfaces are abstract in that they do not make any assumption on what is optimized. Therefore, the framework can be used to optimize other parameterized problems, such as dipole arrangements (inverse solution). The feature of locking arbitrary parameters during optimization can be exploited to introduce a priori information. A subset of sensors may be predefined or only the directions of some or all sensors may be optimized (rotating fit).

Acknowledgments

The author wishes to thank his supervisors Prof. Dr.-Ing. Jens Haueisen and Prof. Dr. Ernst-Günter Schukat-Talamazzini for their support throughout the course of this diploma thesis. Prof. Haueisen has provided invaluable experience in the field of biomagnetism. I wish to thank Roland Eichardt from the Institute for Biomedical Engineering and Informatics (TU Ilmenau) for our interesting work on the tabu search algorithm [56, 55], which we compared the PSO algorithm to. I am grateful to Prof. Dr. Luca Di Rienzo from the Department of Electrical Engineering (Politechnic U. Milan, Italy) for his advice on figures of merit and the discussion of my work.

I also wish to thank Dr. Thomas R. Knösche, a SimBio developer, from the Max Planck Institute of Human Cognitive and Brain Sciences (Leipzig) for commenting on the object-oriented software design. I am grateful to Mario Liehr from the Biomagnetic Center (Jena) for his expert advice and for providing different sensor geometries. I wish to thank Dr.-Med. Matthias Görnig from the Cardiology Department of the University Hospital (Jena) for his clinical advice. I am also grateful to Dr. Hannes Nowak from the Biomagnetic Center (Jena) for providing images of real systems and sensors.

Finally, I thank the Biomagnetic Center (Jena) and the Institute for Biomedical Engineering and Informatics (TU Ilmenau) for providing the computing resources necessary for the simulations.

References

[1] "Particle swarm optimization standard 2006," Website: http://www.particleswarm.info, accessed 1. June 2007.

[2] W. Andrä and H. Nowak, *Magnetism in Medicine*. Weinheim: Wiley-VCH, 2006.

[3] C. M. Arturi, L. D. Rienzo, and J. Haueisen, "Information content in single-component versus three-component cardiomagnetic fields," *IEEE Trans. Magn.*, vol. 40, pp. 631–634, 2004.

[4] R. C. Barr, M. S. Spach, and S. Herman-Giddens, "Selection of the number and position of measuring locations for electrocardiography," *IEEE Transactions on Biomedical Engineering*, vol. 18, pp. 125–138, 1971.

[5] G. Bison, R. Wynands, and A. Weis, "A laser-pumped magnetometer for the mapping of human cardiomagnetic fields," *Applied Physics B - Laser and Optics*, vol. 76, pp. 325–328, 2003.

[6] ——, "Optimization and performance of an optical cardiomagnetometer," *J. Opt. Soc. Am. B*, vol. 22, pp. 77–87, 2005.

[7] G. Booch, *Object-oriented analysis and design: with applications*. Redwood: Benjamin/Cummings, 1994.

[8] J. Bork, H. D. Hahlbohm, and R. Klein, *The 8-layered magnetically shielded room of the PTB: Design and construction*. Helsinki: Helsinki University of Technology, 2001, pp. 970–973.

[9] H. Brauer, T. Knösche, and F. Zanow, "Inverse biomagnetic field calculation using the boundary element method," in *Proceedings IGTE-Symposium*, Graz, Austria, 1992, pp. 127–132.

[10] D. Budker and M. Romalis, "Optical magnetometry," *Nature*, vol. 3, pp. 227–234, 2007.

[11] G. Ciuprina, D. Ioan, and I. Munteanu, "Use of intelligent-particle swarm optimization in electromagnetics," *IEEE Transactions on Magnetics*, vol. 38, pp. 1037–1040, 2002.

[12] M. Clerc, *Discrete particle swarm optimization*. New York: Springer, 2004, pp. 219–240.

[13] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58–73, 2002.

[14] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 256–279, 2004.

[15] B. N. Cuffin and D. B. Geselowitz, "Computer model studies of the magnetocardiogram," *Annals of Biomedical Engineering*, vol. 5, pp. 164–178, 1977.

[16] A. Despopoulos and S. Silbernagl, *Color atlas of physiology*. New York: Thieme Medical Publishers, 2003.

[17] L. Di Rienzo, J. Haueisen, and C. M. Arturi, "Three component magnetic field data: Impact on minimum norm solutions in a biomedical application," *COMPEL: Int. J. for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 24, pp. 869–881, 2005.

[18] D. Durrer, R. T. Vandam, G. E. Freud, M. J. Janse, F. L. Meijler, and R. C. Arzbaech, "Totatl excitation of the isolated human heart," *Circulation*, vol. 41, pp. 899–912, 1970.

[19] R. Eichardt, J. Haueisen, T. R. Knösche, and E. G. Schukat-Talamazzini, "Reconstruction of multiple neuromagnetic sources using augmented evolution strategies - a comparative study," *IEEE Trans. Biomedical Engineering*, 2007, submitted.

[20] M. Eiselt, F. Giessler, D. Platzek, J. Haueisen, U. Zwiener, and J. Rother, "Inhomogeneous propagation of cortical spreading depression - detection by electro- and magnetoencephalography in rats," *Brain Research*, vol. 1028, pp. 83–91, 2004.

[21] Elsevier, *Roche Lexikon Medizin*. Munich, Germany: Elsevier, 2003.

[22] A. P. Engelbrecht, *Fundamentals of swarm intelligence*. Chichester: Wiley, 2005.

[23] R. L. Fagaly, "Superconducting quantum interference device instruments and applications," *Review of scientific instruments*, vol. 77, 2006.

[24] J. Fingberg, G. Berti, U. Hartmann, A. Basermann, F. Zimmermann, C. H. Wolters, A. Anwander, A. McCarthy, and S. Woods, "Bio-numerical simulations with SimBio," *NEC Res Dev*, vol. 44, pp. 140–145, Jan. 2003.

[25] D. D. Finlay, C. D. Nugent, M. P. Donnelly, R. L. Lux, P. J. McCullagh, and N. D. Black, "Selection of optimal recording sited for limited lead body surface potential mapping: A sequential selection based approach," *BMC Medical Informatics and Decision Making*, vol. 6, 2006.

[26] J. J. Gallagher, L. C. Pritchett, W. C. Sealy, J. Kassel, and A. G. Wallace, "The pre-excitation syndromes," *Progress in Cardiovascular Diseases*, vol. 20, pp. 285–327, 1978.

[27] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns - elements of reuseable object-oriented software*. Amsterdam: Addison-Wesley Longman, 1995.

[28] T. Gargano, "Comparison criteria for sensor arrays in magnetostatic inverse problems," Diploma Thesis, Milan, Italy, 2006.

[29] D. B. Geselowitz, "On bioelectric potentials in an inhomogeneous volume conductor," *Biophysical Journal*, vol. 7, pp. 1–11, 1967.

[30] D. B. Geselowitz and W. T. Miller, "Extracorporeal magnetic fields generated by internal bioelectric sources," *IEEE Transactions on Magnetics*, vol. 9, pp. 392–398, 1973.

[31] F. Glover and M. Laguna, *Tabu Search*. Oxford: Kluwer Academic, 2001.

[32] G. H. Golub and C. F. V. Loan, *Matrix computations*. Baltimore: John Hopkins University Press, 1996.

[33] B. Grimm, J. Haueisen, M. Huotilainen, S. Lange, P. VanLeeuwen, T. Menendez, M. J. Peters, E. Schleussner, and U. Schneider, "Recommended standards for fetal magnetocardiography," *Pace, Pacing and Clinical Electrophysiology*, vol. 26, pp. 2121–2126, 2003.

[34] F. Grynszpan and D. B. Geselowitz, "Model studies of the magnetocardiogram," *Biophysical Journal*, vol. 13, pp. 911–925, 1973.

[35] A. C. Guyton, *Textbook of medical physiology*. Philadelphia: Saunders, 1991.

[36] M. S. Haemaelaeinen and R. J. Ilmoniemi, "Interpreting measured magnetic fields of the brain: Estimates of current distributions," Low Temperature Laboratory, Helsinki University of Technology, Tech. Rep. TKK-F-A559, 1984.

[37] J. Haueisen, *Numerische Berechnung und Analyse biomagnetischer Felder*. Ilmenau, Germany: Wiss.-Verlag Ilmenau, 2004.

[38] ——, personal communication, June 2007.

[39] J. Haueisen, A. Bottner, M. Funke, H. Brauer, and H. Nowak, "The influence of boundary element discretization on the forward and inverse solution problem in electroencephalography and magnetoencephalography," *Biomedical Engineering (Biomedizinische Technik)*, vol. 42, pp. 240–248, 1997.

[40] J. Haueisen, J. Schreiber, H. Brauer, and T. R. Knösche, "Dependence of the inverse solution accuracy in magnetocardiography on the boundary-element discretization," *IEEE Transactions on Magnetics*, vol. 38, pp. 1045–1048, 2002.

[41] R. Hren, X. Zhang, and G. Stroink, "Comparison between electrocardiographic and magnetocardiographic inverse solutions using the boundary element method," *Medical & Biological Engineering & Computing*, vol. 34, pp. 110–114, 1996.

[42] M. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila, and O. Lounasmaa, "Magnetoencephalography - theory, instrumentation and applications to noninvasive studies of the working human brain," *Reviews of Modern Physics*, vol. 65, pp. 413–597, 1993.

[43] J. D. Jackson, *Classical Electrodynamics*. New York: Wiley, 1999.

[44] J. Jaja, *An introduction to parallel algorithms*. Reading, Mass.: Addison-Wesley, 1995.

[45] A. Jennings and J. J. McKeown, *Matrix computation*. Chichester: Wiley, 1992.

[46] T. Katila, *Instrumentation for biomedical applications*. Berlin: Walter de Gruyter, 1981.

[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks*, vol. 4, Piscataway, NJ, USA, 1995, pp. 1942–1948.

[48] J. Kennedy, R. C. Eberhart, and S. Yuhui, *Swarm intelligence*. San Francisco: Morgan Kaufmann, 2001.

[49] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 36, pp. 515–519, 2006.

[50] Kim, Y. H. Lee, H. Kwon, J. M. Kim, I. S. Kim, and Y. K. Park, "Optimal sensor distribution for measuring the tangential field components in mcg," *Neurology and Clinical Neurophysiology*, vol. 60, 2004.

[51] K. Kim, W.-K. Lee, I. Kim, and H. S. Moon, "Atomic vector gradiometer system using cesium vapor cells for magnetocardiography: perspective on practical application," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, pp. 458–462, 2007.

[52] T. Knösche, "Magnetocardigraphische feldmodellierung und quellenlokalisation mit hilfe der randelementemethode," Diplomarbeit, Ilmenau, Germany, 1992.

[53] K. Kobayashi and Y. Uchikawa, "Development of a high spatial resolution squid magnetometer for biomagnetic measurement," *IEEE Transactions on Magnetics*, vol. 39, pp. 3378–3380, 2003.

[54] P. P. Laine, M. S. Hämäläinen, and A. I. Ahonen, *Combination of magnetometers and planar gradiometers in an MEG system*. Sendai: Tohoku University Press, 1999, pp. 47–50.

[55] S. Lau, R. Eichardt, L. D. Rienzo, and J. Haueisen, "Particle swarm vs. tabu search optimization of magnetic sensor systems for magnetocardiography," in *Proc. 41st Annual Conference of the German Society for Biomedical Engineering (DGBMT)*, Aachen, Germany, Sept. 2007, submitted.

[56] ——, "Tabu search optimization of magnetic sensor systems for magnetocardiography," in *Proc. 16th Int. C. on Computation of Electromagnetic Fields (COMPUMAG)*, Aachen, Germany, June 2007.

[57] U. Leder, J. Haueisen, M. Huck, and H. Nowak, "Non-invasive imaging of arrhythmogenic left-ventricular myocardium after infarction," *LANCET*, vol. 352, pp. 1825–1825, 1998.

[58] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[59] M. Liehr, J. Haueisen, M. Görnig, P. Seidel, J. Nenonen, and T. Katila, "Vortex shaped current sources in a physical torso phantom," *Annals of Biomedical Engineering*, vol. 33, pp. 240–247, 2005.

[60] R. L. Lux, A. K. Evans, M. J. Burgess, and J. A. Abildskov, "Redundancy reduction for improved display and analysis of body-surface potential maps 1. spatial compression," *Circulation Research*, vol. 49, pp. 186–196, 1981.

[61] R. L. Lux, C. R. Smith, R. F. Wyatt, and J. A. Abildskov, "Limited lead selection for estimation of body-surface potential maps in electrocardiography," *IEEE Transactions on Biomedical Engineering*, vol. 25, pp. 270–276, 1978.

[62] J. Malmivuo and R. Plonsey, *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*. New York: Oxford Univ. Press, 1995.

[63] D. W. Marquardt, "An algorithm for least squares estimation of nonlinear parameters," *SIAM J. Appl. Math.*, vol. 11, pp. 431–441, 1963.

[64] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 204–210, 2004.

[65] B. Messnarz, M. Seger, R. Modre, G. Fischer, F. Hanser, and B. Tilg, "A comparison of non-invasive reconstruction of epicardial vs. transmembrane potentials in consideration of the null space," *IEEE Transactions on Biomedical Engineering*, vol. 51, pp. 1609–1618, 2004.

[66] D. R. Musser, "Introspective sorting and selection algorithms," *Software Practice and Experience*, vol. 27, p. 983, 1997.

[67] M. Nalbach and O. Doessel, "Comparison of sensor arrangements of MCG and ECG with respect to information content," *Physica C*, vol. 372-376, pp. 254–258, 2002.

[68] J. Nenonen, M. Kajola, J. Simola, and A. Ahonen, "Total information of multichannel MEG sensor arrays," in *Proc. of the C. on Biomagnetism (BIOMAG) 2004*, Boston, MA, USA, Aug. 2004, pp. 630–631.

[69] ——, "Total information extracted from MEG measurements," in *Proc. of the C. on Biomagnetism BIOMAG 2006*, Vancouver, Canada, Aug. 2006.

[70] H. Nowak, U. Leder, M. Goernig, J. Haueisen, S. N. Erné, and A. Trebeschi, "Multichannel-vectormagnetocardiography: a new biomedical engineering approach," in *37th Annual Conference of the German, Austrian and Swiss Societies for Biomedical Engineering (BMT)*, St. Virgil/Salzburg, 2003.

[71] H. Nowak, E. Strähmel, F. Giessler, G. Rinneberg, and J. Haueisen, "Sensitive magnetic sensors without cooling in biomedical engineering," *Biomedical Engineering (Biomedizinische Technik)*, vol. 48, pp. 2–10, 2003.

[72] A. D. Pascual-Marqui, "Standardized low resolution brain electromagnetic tomography (sLORETA): Technical details," *Methods and Findings in Experimental and Clinical Pharmacology*, vol. 24, Suppl. D, pp. 5–12, 2002.

[73] R. D. Pascual-Marqui, "Low resolution brain electromagnetic tomography," *Brain Topography*, vol. 7, p. 180, 1994.

[74] R. D. Pascual-Marqui, C. M. Michel, and D. Lehmann, "Low-resolution electromagnetic tomography - a new method for localizing electrical activity in the brain," *Internation Journal of Psychophysiology*, vol. 18, pp. 49–65, 1994.

[75] R. Plonsey, "Capability and limitations of electrocardiography and magnetocardiography," *IEEE Transactions on Biomedical Engineering*, vol. 19, p. 239, 1972.

[76] ——, "The nature of sources of bioelectric and biomagnetic fields," *Biophysical Journal*, vol. 39, pp. 309–312, 1982.

[77] R. Plonsey and D. B. Heppner, "Considerations of quasi-stationarity in electrophysiological systems," *Bulletin of Mathematical Biophysics*, vol. 29, pp. 657–664, 1967.

[78] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical recipes: The art of scientific computing*. Cambridge, UK: Cambridge University Press, 1993.

[79] C. J. Purcell, G. Stroink, and B. M. Horacek, "Effect of torso boundaries on electric potential and magnetic field of a dipole," *IEEE Transactions on Biomedical Engineering*, vol. 35, pp. 671–678, 1988.

[80] L. D. Rienzo and J. Haueisen, "Theoretical lower error bound for comparative evaluation of sensor arrays in magnetostatic linear inverse problems," *IEEE Trans. Magn.*, vol. 42, pp. 3669–3673, 2006.

[81] ——, "Numerical comparison of sensor arrays for magnetostatic linear inverse problems based on a projection method," *COMPEL*, vol. 26, pp. 356–367, 2007.

[82] D. Robbes, "Highly sensitive magnetometers - a review," *Sensors and Actuators A*, vol. 129, pp. 86–93, 2006.

[83] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, pp. 397–407, 2004.

[84] S. E. Robinson, M. B. Burbank, A. A. Haid, P. R. Kubik, I. Sekachev, B. Taylor, M. Tillotson, J. Vrba, G. Wong, C. Lowery, H. Eswaran, D. Wilson, P. Murphy, and H. Preißl, *A biomagnetic instrumentation for human reproductive assessment*. Helsinki: Helsinki University of Technology, 2001, pp. 919–922.

[85] L. Rouve, L. Schmerber, O. Chadebec, and A. Foggia, "Optimal magnetic sensor location for spherical harmonics identification applied to radiated electrical devices," *IEEE Trans. Magn.*, vol. 42, pp. 1167–1170, 2006.

[86] M. Saarinen, P. Siltanen, P. J. Karp, and T. E. Katila, "Normal magnetocardiogram - morphology," *Annals of Clinical Research*, vol. 10, pp. 1–43 Suppl. 21, 1978.

[87] A. Schnabel, M. Burghoff, S. Hartwig, F. Petsche, U. Steinhoff, D. Drung, and H. Koch, "A sensor configuration for a 304 SQUID vector magnetometer," *Neurology and Clinical Neurophysiology*, vol. 70, 2004.

[88] U. Schneider, E. Schleussner, J. Haueisen, H. Nowak, and H. J. Seewald, "Signal analysis of auditory evoked cortical fields in fetal magnetoencephalography," *Brain Topography*, vol. 14, 2001.

[89] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, pp. 10–21, 1949.

[90] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. University of Illinois Press, 1948.

[91] I. Sommerville, *Software Engineering*, 7th ed. Boston: Pearson, 2004.

[92] B. Stroustrup, *The C++ programming language*. Boston: Addison-Wesley, 2006.

[93] S. Taulu and M. Kajola, "Presentation of electromagnetic multichannel data: The signal space separation method," *J. of Applied Physics*, vol. 97, p. 124905, 2005.

[94] U. Tenner, "Source modeling in cardiomagnetism: A physical torso phantom for biomagnetic and bioelectric heart field measurements," Ph.D. dissertation, Faculty of Electrical Engineering and Information Technology, Technical University Ilmenau, Ilmenau, Germany, 2000.

[95] A. Tikhonov and V. Arsenim, *Solutions of ill-posed problems*. New York: Wiley, 1977.

[96] S. Tumanski, "Induction coil sensors - a review," *Measurement, Science and Technology*, vol. 18, pp. R31–R46, 2007.

[97] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 225–239, 2004.

[98] J. Vrba, "Multichannel SQUID biomagnetic systems," in *Applications of Superconductivity*, H. Weinstock, Ed. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000, pp. 61–138.

[99] J. Vrba and S. E. Robinson, "SQUID sensor array configurations for magnetoencephalography applications," *Supercond. Sci. Technol.*, vol. 15, pp. R51–R89, 2002.

[100] C. Wolters, *Influence of tissue conductivity inhomogeneity and anisotropy on EEG/MEG based source localization in the human brain*, ser. MPI Series in Cognitive Neuroscience, Max-Planck-Institute of Cognitive Neuroscience Leipzig, Germany, 2003.

[101] C. H. Wolters, A. Anwander, X. Tricoche, D. Weinstein, M. A. Koch, and R. S. MacLeod, "Influence of tissue conductivity anisotropy on EEG/MEG field and return current computation in a realistic head model: A simulation and visualization study using high-resolution finite element modeling," *NeuroImage*, vol. 30, pp. 813–826, 2006.

[102] H. Xia, A. B. A. Baranga, D. Hoffman, and M. V. Romalis, "Magnetoencephalography with an atomic magnetometer," *Applied Physics Letters*, vol. 89, p. 211104, 2006.

List of Figures

List of Tables

List of Algorithms

Declaration

I declare that this submission is my own work and that I used only the stated sources and aids during its preparation.

Jena, June 29, 2007

Stephan Lau